MIPITS
An Agent based Intelligent Tutoring System

Egons Lavendelis and Janis Grundspenkis
Department of Systems Theory and Design, Riga Technical University, 1 Kalku street, Riga, Latvia

Keywords: Intelligent tutoring systems, Multi-agent systems, Holonic agents.

Abstract: During the last decades many Intelligent Tutoring Systems (ITS) are developed to add adaptivity and intelligence to the e-learning systems. Intelligent agents and multi-agent systems are widely used to implement intelligent mechanisms for ITSs due to their characteristics. The paper presents an agent based ITS for the course "Fundamentals of Artificial Intelligence" named MIPITS. The MIPITS system is based on the holonic multi-agent architecture for ITS development. The system offers learning materials, provides practical problems and gives feedback to the learner about his/her solution evaluating his/her knowledge. The goal of the system is to realize individualized practical problem solving, which is not possible in the classroom due to the large number of students in the course. Thus, the main focus of the system is on problem solving. The system offers three types of problems: tests, state space search problems and two-player games algorithm problems. The MIPITS system is open: new types of problems can be added just by including appropriate agents in the system. The problems are adapted to the learner's knowledge level and preferences about difficulty, size and practicality of problems.

1 INTRODUCTION

Nowadays, the availability of education must be increased to successfully develop knowledge society. Many learners having different knowledge levels and learning styles must be taught together. The traditional tutoring process is not effective in simultaneously teaching many different learners. There is a need for easier available and more individualised tutoring process that adapts to every learner. Additionally, people need to study new things after graduating, because many technologies change very rapidly. Thus, the lifelong education becomes very important.

Different e-learning technologies are used to teach large numbers of students, facilitate availability of education and lifelong learning. Learning management systems like Blackboard (http://www.blackboard.com/) and Moodle (http://moodle.org/) are among the most popular ones. These systems are available from any place with an Internet connection and at any time, thus learners can choose when and where to study the course.

E-learning systems mainly offer learning materials and different kinds of tests to evaluate learners' knowledge. The majority of them use the same materials and tests for all learners. Thus, traditional e-learning systems can not adapt to any specific characteristics of the learner and therefore can not realize individualized tutoring. Moreover, e-learning systems usually are not capable to generate any learning material or test using domain knowledge. The teacher must create all learning materials and tests that are used in the course.

To eliminate the abovementioned drawbacks of e-learning systems, Intelligent Tutoring Systems (ITS) are developed. ITSs imitate the teacher realizing individualized tutoring, using domain and pedagogical knowledge as well as knowledge about the learner. ITSs to a certain extent can adapt learning materials, generate tasks and problems from domain knowledge, evaluate learner's knowledge and provide informative feedback. So, ITSs add adaptivity to above mentioned benefits of e-learning systems (Brusilovsky and Peylo, 2003). During the last 40 years since the first ITS named SCHOLAR and teaching geography (Carbonelli, 1970), large number of ITS have been developed. Well-known examples of ITSs are FLUTE (Devedzic et al., 2000), ABITS (Capuano et al., 2000), Passive Voice Tutor (Virvou and Tsiriga, 2001), Slide Tutor (Crowley and Medvedeva, 2005) and Ines (Hospers et al., 2003).

Lavendelis E. and Grundspenkis J. (2010).
MIPITS - An Agent based Intelligent Tutoring System.
In *Proceedings of the 2nd International Conference on Agents and Artificial Intelligence - Agents*, pages 5-13
DOI: 10.5220/0002707200050013

The abovementioned examples show, that ITSs mainly are dedicated to specific courses. The paper proposes the ITS for the course "Fundamentals of Artificial Intelligence" named MIPITS. The MIPITS system offers learning materials and problems to the learner, evaluates learner's knowledge in each topic and provides feedback to the learner. The system adapts problems to the learner's knowledge level and preferences, described below.

The remainder of the paper is organized as follows. The Section 2 contains general description of the developed system. The architecture of the system is given in the Section 3. The tutoring scenario implemented in the system is described in the Section 4. The Section 5 concludes the paper and gives brief outline of the future work.

2 THE MIPITS SYSTEM

The MIPITS system is developed for the course "Fundamentals of Artificial Intelligence" simultaneously taught to more than 200 students at Riga Technical University. The course contains topics about different algorithms used in artificial intelligence like search algorithms and algorithms for two-player games. Important part of learning such algorithms is practice. However, any guidance and feedback during the practice is limited due to the large number of students. Additionally, it is almost impossible to prepare unique problems and tasks for all students manually. The aim of the MIPITS system is to solve the issues of problem generation, limited guidance and feedback during the practice with different algorithms taught in the course. Moreover, students attending the course have very different knowledge level and learning styles. At the same time, no individualized tutoring can be done in the classroom due to the large number of students. Thus the ITS can improve the tutoring process by adapting to the learner's knowledge level and learning style.

The MIPITS system is intended as an addition to the traditional classroom tutoring. Firstly, the learner attends lectures in the classroom. Later he/she has an opportunity to repeat the topics taught in the classroom and practice in the problem solving using the system. However, it is possible to use the system without attending the classes, because it covers all necessary activities to learn the basics of the corresponding topics. In each topic the MIPITS system offers the learning material, and the problem to be solved by the learner. In the MIPITS system the problem is any task, test or problem used to evaluate the learner's knowledge. After the learner has finished the problem the system evaluates his/her solution and gives appropriate feedback.

The main focus of the MIPITS system is on problem solving. The system provides unique problems that are appropriate to the knowledge level and preferences of the individual learner. Initial version of the system is developed for first three modules of the course - „Introduction", „Uninformed Search" and „Informed Search" (Luger, 2005). Thus, the system is capable to offer the corresponding types of problems:

- Different types of tests: single choice tests, multiple choice tests and tests, where a learner has to write the answer by him/herself. Figures and state spaces can be added to the question.
- Search algorithm problems, where a learner has to do a state space search using the specified algorithm and lists OPEN and CLOSED (Luger, 2005).
- Two-player game problems, where a learner has to apply the MINIMAX algorithm or Alpha-Beta pruning to the given state space (Luger, 2005).

Other types of problems can be added to the system. When the learner requests a task the system finds the most appropriate problem to the learner's knowledge level and preferences among problems of all types that fit the topic and gives it to the learner.

3 THE ARCHITECTURE OF THE MIPITS SYSTEM

ITSs mainly are built as modular systems consisting of four traditional modules: the tutoring module, the expert module, the student diagnosis module and the communication module (Smith, 1998). During the last decade intelligent agents are widely used to implement traditional modules (Grundspenkis and Anohina, 2005). Well-known examples of agent based ITSs are ITS for enhancing e-learning (Gascuena and Fernández-Caballero, 2005), ABITS (Capuano et al., 2000) and Ines (Hospers et al., 2003).

The MIPITS system is developed using open holonic multi agent architecture for ITS development described in (Lavendelis and Grundspenkis, 2008). The architecture consists of the higher level agents that implement the abovementioned modules. All higher level agents can be implemented as holons (Fischer, 2003). Each holon consists of one head agent and a number of body agents. The head of the holon is responsible for communication outside the holon and coordination

of all body agents. Open and closed holons are distinguished. Open holons consist of the head and a certain type of body agents, however, the number and exact instances of body agents are not known during the design of the system and can be changed during the maintenance and runtime of the system so modifying the system's functionality. The body agents have to register their services at the directory facilitator agent. Heads of open holons use the directory facilitator agent to find actual body agents in each open holon. Closed holons consist of agent instances that are specified during the design and can not be changed during the runtime of the system. The development of agent based ITSs using the open holonic architecture is supported with the MASITS methodology (Lavendelis and Grundspenkis, 2009a) and the MASITS tool (Lavendelis and Grundspenkis, 2009b) which are used to develop the MIPITS system. According to the MASITS methodology, the system is implemented in the JADE platform (http://jade.tilab.com/). Further implementation details are omitted due to the scope of the paper.

The architecture of the system is shown in Figure 1. Heads of open holons are denoted with gray colour. The developed system consists of the following higher level agents. The communication module is implemented as an *interface agent* that carries out all interactions with the learner. It is responsible for the following tasks:

- Collecting the registration information about the learner and his/her preferences and carrying out the registration process.
- Log in process, including the validation of learner's log in data.
- Perceiving learner's requests and starting the processes in the system by forwarding learner's requests, actions and data to the appropriate agents.
- Giving all information to a learner, including learning materials, all types of problems and feedback.

The interface agent is the only agent interacting with a learner. Thus, it is the head of the higher level holon.

The tutoring module is implemented as the teaching strategy agent, the problem generation agent, the curriculum agent and the feedback generation agent. The *teaching strategy agent* is responsible for provision of the learning material in each topic. The *curriculum agent* is responsible for creation of the curriculum during the registration of a learner in the system. The *problem generation agent* is responsible for generation of all types of problems used in the system and adaptation of these problems to the knowledge level and preferences of the learner.

The expert module is implemented as the *expert agent*, which is responsible for solving all types of problems.

The student diagnosis module is implemented as the student modelling agent and the knowledge evaluation agent. The *student modelling agent* is responsible for creating, updating and providing the student model upon request of any other agent. The initial student model is created during the registration process. It is modified by reacting on the different actions reported by other agents. The student model contains:

- The personal data of a learner that are collected during the registration process.
- The preferences of a learner that are collected during the registration process. The following preferences are collected: the preferred (initial) difficulty of problems, the preferred practicality of problems and the preferred size of problems described below.
- The curriculum. It is created for a learner during the registration process. Additionally, each topic has its status denoting what activities a learner has completed in the topic. The status has the following possible values: "initial", "started", "finished theoretical part", and "finished".
- All problems given to a learner and the results of all knowledge evaluations based on his/her solutions of the given problems.

The knowledge evaluation agent has to find the learner's mistakes in his/her solution by comparing it to the expert's solution. It must be able to evaluate solutions of all types of problems.

According to the MASITS methodology, to implement different types of problems and allow adding new problems all higher level agents dealing with problems are implemented as open holons. Thus, the problem generation agent, the expert agent, the knowledge evaluation agent and the interface agent are implemented as open holons. The problem generation holon consists of body agents that generate one type of problems. So, it consists of the following body agents: the test generation agent, the search problem generation agent and the two-player games problem generation agent. Similarly, body agents of the expert holon are capable to solve problems of the certain type. Knowledge evaluation body agents compare system's and learner's solutions of the given problem. Each interface body agent is capable to manage user interface of one type of problems. The heads of open holons are only capable to find the appropriate body agent and forward results received from them.

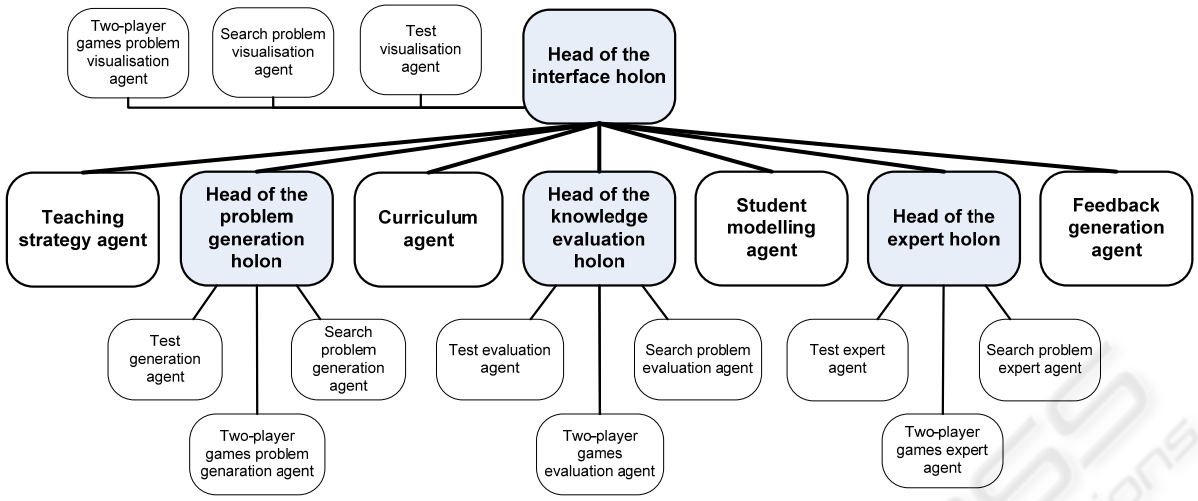

Figure 1: Architecture of the MIPITS system.

The open architecture of the MIPITS system makes it extendable to teach new topics of the course or even some other courses by including new types of problems and appropriate materials. There is no need to change code of existing agents to include new types of problems. It can be done by adding appropriate body agents to the open holons.

4 THE TUTORING SCENARIO OF THE MIPITS SYSTEM

To adapt problems to learner's characteristics a learner must be identified. Thus, the first activity a learner has to do is to *register in the system*. For this purpose a learner fills a form containing his/her personal data and his/her preferences. After a learner has submitted the registration form, the interface agent collects data from the form, checks the data, inserts user data into the database and sends the data to the student modelling agent. The student modelling agent creates the initial student model based on learner's preferences and requests the curriculum agent to create the curriculum for a learner. After receiving the curriculum from the curriculum agent the student modelling agent completes the initial student model by adding the curriculum and sends it to the interface agent, who opens the main window with the curriculum and information about the first module. Interactions among agents are implemented using simple messages. Predicates from the domain ontology are used to express message contents. Messages sent during the registration process are shown in Figure 2.

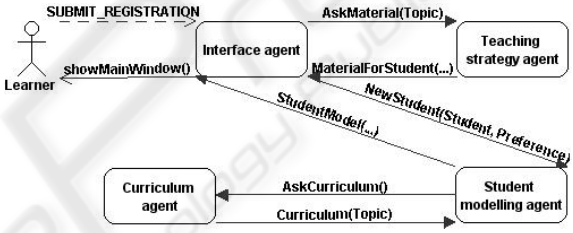

Figure 2: Interactions done during the registration.

Each time a registered learner logs in to the system the learning process is restarted at the topic that was open when a learner quit the system last time. To do it, first, learner's user data are validated by the interface agent and sent to the student modelling agent. Second, the student modelling agent reads the student model from the database and sends it to the user interface agent. Third, the user interface agent requests the teaching strategy agent to provide the material in the current topic. Finally, when the interface agent receives the material, the main window of the system containing the curriculum and the learning material in current topic is opened. Interactions done during the login process are shown in Figure 3.

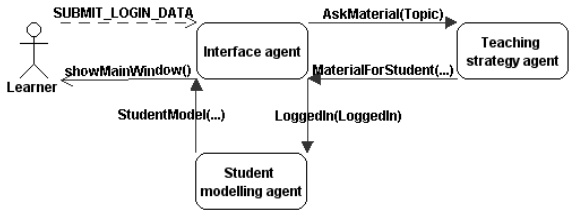

Figure 3: Interactions done during the login.

The curriculum of the course consists of modules that, in their turn, consist of topics. To support teaching of a topic the MIPITS system performs the following scenario consisting of three steps. When a learner chooses to start learning a topic, the system starts the *theoretical step*. During this step a learner studies a theoretical material. After finishing it a learner requests a test. The system switches to the *problem solving step*. During this step a learner has to solve some problems in the current topic. After finishing, a learner submits his/her solutions. The system moves to the *knowledge evaluation step*. As a result of this step a learner receives an evaluation of his/her knowledge in the current topic and constructive feedback about errors made in each problem. When the knowledge evaluation step is over, a learner can choose to start learning a new topic.

After finishing all topics of the module a learner has to pass the final test of the module that may contain problems from all topics included in this module. During the final testing of the module all actions of the problem solving and knowledge evaluation steps are done.

4.1 The Theoretical Step

The goal of the theoretical step is to give a learning material to a learner allowing him/her to repeat theory of the topic that has been given in the classroom. The step is carried out using the following scenario. When a learner chooses the topic to start learning, the interface agent requests the teaching strategy agent to generate a learning material in the chosen topic. The teaching strategy agent finds appropriate learning material and sends it to the interface agent. The interface agent shows a learning material in the user interface of the system. Additionally, the teaching strategy agent notifies the student modelling agent that a learning material in the current topic has been given to a learner. The student modelling agent changes the student module by modifying the status of the topic from “initial” to “started”. Messages sent among agents during the theoretical step are shown in Figure 4.

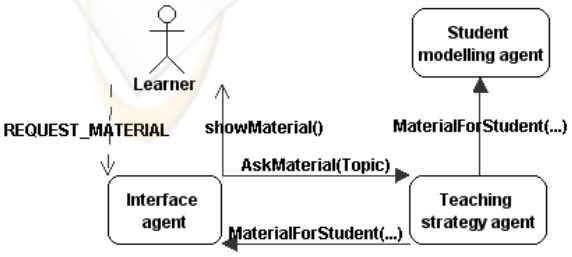

Figure 4: Interactions done during the theoretical step.

4.2 The Problem Solving Step

The goal of the problem solving step is to give a learner an opportunity to practice in different types of problems. The knowledge evaluation step is based on learner’s solutions in the problem solving step. The problem solving step starts when a learner submits that he has studied a material. The interface agent requests the problem generation agent to generate the problem in the current topic. The request is processed by the head of the problem generation holon, using the following algorithm (see Figure 5). Firstly, the head queries the student modelling agent to get full student model and the directory facilitator to find the body agents of the problem generation holon. If there are no problem generation body agents registered to the directory facilitator, the system error is generated. Otherwise, after receiving replies from the student modelling agent and the directory facilitator all body agents are queried to generate a problem in the current topic that is appropriate to learner’s characteristics. Each problem generation body agent either generates the most appropriate problem to learner’s characteristics and sends it to the head of the holon or sends failure to the head of the holon if it can not generate a problem in the current topic.

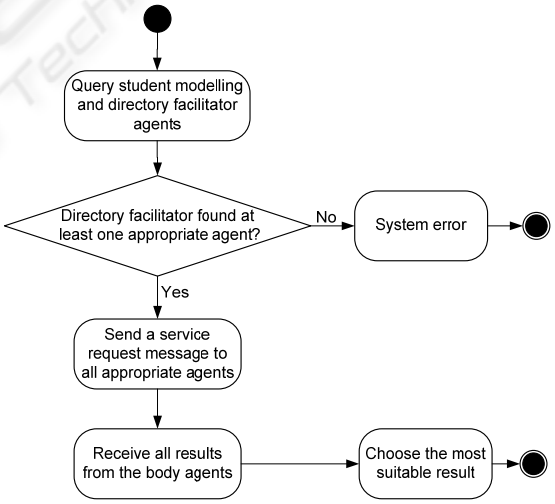

Figure 5: Algorithm for the head of the problem generation holon.

After receiving all problems the head of the holon has to choose the most appropriate one to the learner’s characteristics. The following criteria are used to choose the most appropriate problem:

- Difficulty of the problem. The difficulty of the problem must match the preferable level of

difficulty as close as possible, because the problem should not be too complex for a learner (unsolvable) nor too easy (not interesting).

- The size of the problem. A learner is allowed to choose whether his/her knowledge evaluation will be done with small and concrete problems or large and time consuming problems.
- The practicality of the problem. A learner is allowed to choose between more practical and more theoretical problems to match preferences of more practically and more theoretically oriented learners.
- Frequency of the type of problem. Different combinations of a learner's characteristics may lead to the situation, that only one type of problems is used. However, knowledge evaluation using only one type of problems becomes too monotony. Thus, the frequency of the type of problems should be minimized.

The first action to choose the most appropriate task for learner's characteristics is calculation of the preferable values of all criteria:

- The difficulty of the problem. During the registration process a learner evaluates his/her knowledge level by specifying initial difficulty in the scale from 1 (the easiest problems) to 5(most difficult problems). This is only a subjective learner's estimate that can be inaccurate. Thus, the system calculates the preferred difficulty using the initial difficulty and learner's results in previous knowledge evaluations. Moreover, the more a learner has been working with the system, the more valuable is knowledge evaluation by the system and the less valuable is the initial difficulty. The preferred difficulty is calculated as follows:

$$dif_{pref} = \frac{init*dif_{init} + max*l}{init+max}, \text{ where} \quad (1)$$

init – coefficient denoting how much points for problem solving are equivalent to the initial difficulty. The value of the coefficient is determined empirically and is 50. For comparison, one question in the test is 2 to 4 points worth.
dif_{init} – initial difficulty.
max – maximal number of points that can be scored in the problems solved by a learner.
l – the level of difficulty corresponding to learner's results in the problems he/she has solved. To calculate the level, firstly, a learner's result in percents is calculated. The level is determined using empirical function shown in Table 1.

Table 1: Calculation of the level of difficulty corresponding to learner's results.

Results (%)	Level of difficulty
0-34	1
35-49	2
50-64	3
65-80	4
81-100	5

- The preferred size and practicality of the problem are chosen by the learner during the registration. These criteria are measured in the scale from 1 (small/more theoretical problems) to 3 (large/more practical problems).
- Frequency of the type of the problem is 0 if a learner has not solved any problem yet, otherwise it is calculated as follows:

$$f_i = \frac{n_i}{n}, \text{ where} \quad (2)$$

f_i – frequency of the i-th type of the problem;
n_i – number of problems of the i-th type given to a learner;
n – total number of problems given to a learner.

Each problem received from the problem generation agent contains the values of all criteria. So, after calculating the preferable values of all criteria the difference between preferable and real values is minimised. The appropriateness is calculated as follows:

$$A = -(|dif_{pref} - dif_r| * c_d + |s_{pref} - s_r| * c_s + |pr_{pref} - pr_r| * c_p + f_t * c_f), \text{ where} \quad (3)$$

dif_{pref} – the preferred difficulty of the task;
dif_r – the real difficulty of the task;
c_d – the weight of the difficulty;
s_{pref} – the preferred size of the problem;
s_r – the real size of the problem;
c_s – the weight of the size;
pr_{pref} – the preferred practicality;
pr_r – the real practicality of the problem;
c_p – the weight of the practicality;
f_t – the frequency of the type of the problem;
c_f – the weight of the frequency.

Values of weights in the formula are determined empirically and are the following: c_d=2, c_s=3, c_p=3, c_f=6. With these weights all criteria have significant impact on the appropriateness.

After finding the problem with the highest appropriateness it is sent to three agents:

- To the interface agent, that is responsible for giving it to a learner.
- To the expert agent, that is responsible for finding the correct solution of the problem. The expert agent solves the problem at the same time that a learner is solving it to save time when learner submits his/her solution.
- To the student modelling agent that changes the status of the topic from "started" to "finished theoretical part" in the student model.

Heads of the interface holon and the expert holon are not capable to accomplish the tasks that they are responsible for. Thus, they have to use body agents of their holons. After receiving the problem the heads of the holons are using the same algorithms, generalisation of which is shown in Figure 6. Firstly, the head of the holon uses the directory facilitator to find the appropriate body agent. If such agent is found, the problem is forwarded to the body agent, otherwise system error is generated. The body agent does its job (respectively, gives the problem to a learner or solves it). The body agent of the expert holon sends the solution to the head of the holon and forwards it to the head of the knowledge evaluation holon, which saves the solution to use it in the knowledge evaluation process. All messages sent among agents to give a problem to the learner are shown in Figure 7.

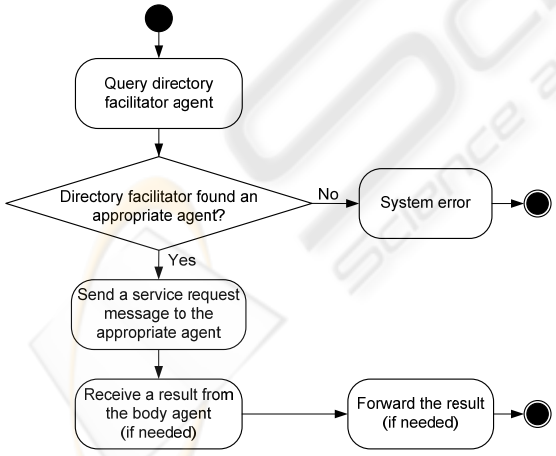

Figure 6: Typical algorithm for the heads of open holons.

As a result of the agents' actions in the problem solving step, the problem is given to the learner using the main window of the system (see Figure 8). The interface of the system is in Latvian, which is the language of the course. The window consists of two main parts: the curriculum denoted with 1 and the main panel denoted with 2. The curriculum consists of all modules and topics taught in the course. The main panel changes its contents depending on the step. It contains materials in the theoretical step and problems in problem solving and knowledge evaluation steps. During the problem solving step the appropriate interface body agent is responsible for creating the panel and managing all activities in it. The screenshot of the system shown in Figure 8 contains the state space search problem. The panel of the problem consists of the following parts: the statement of the problem, the state space denoted with 3 and tools for modifications of data to do the search denoted with 4.

4.3 The Knowledge Evaluation Step

The goal of the knowledge evaluation step is to evaluate learner's solution created in the previous step and give him/her the feedback about the solution. A learner starts the step by submitting his/her solution of the problem. Firstly, the interface agent sends learner's solution to the knowledge evaluation agent. The head of the knowledge evaluation agent uses the algorithm shown in Figure 6. The body agent compares system's and learner's solutions finding learner's mistakes and evaluating the solution. The head of the knowledge evaluation holon forwards the evaluation to the student modelling agent and the feedback agent. The student modelling agent records the knowledge evaluation in the student model and changes the status of the topic to "finished". The feedback agent creates the textual feedback about the result, like "You scored 19 points from 20! Great result!". Additionally, it creates textual information about the learner's mistakes, like "You made a mistake determining the search goal during the last step of the algorithm". After the feedback is prepared it is sent to the interface agent, which gives it to a learner. Interactions among agents done in this step are shown in Figure 9.

All steps of the tutoring scenario are implemented in the way that any new type of problems can be added to the system without modifying the already existing agents. It can be done by adding new agents to the open holons. A single body agent has to be added to each of four open holons: the problem generation holon, the expert holon, the knowledge evaluation holon and the interface agent holon. Newly added body agents have to register themselves to the directory facilitator agent in order the heads of the holons can find them. Additionally, the domain ontology must be refined to include the new classes of problems and their solutions.

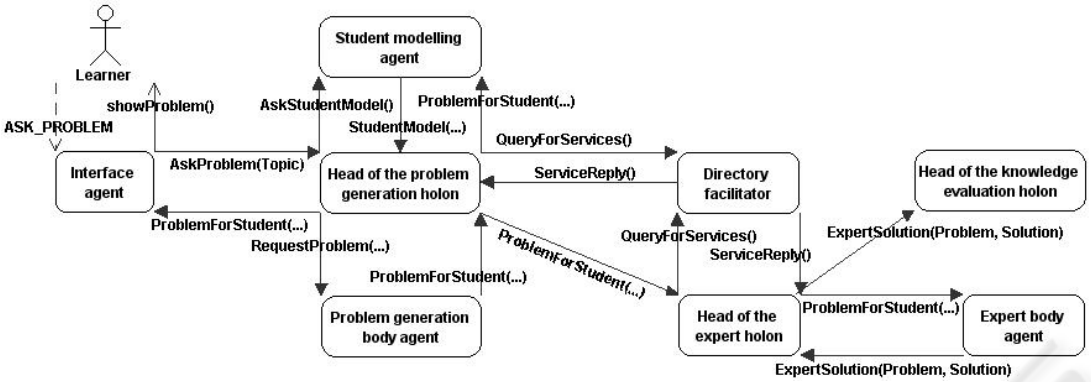

Figure 7: Interactions done during the problem solving.

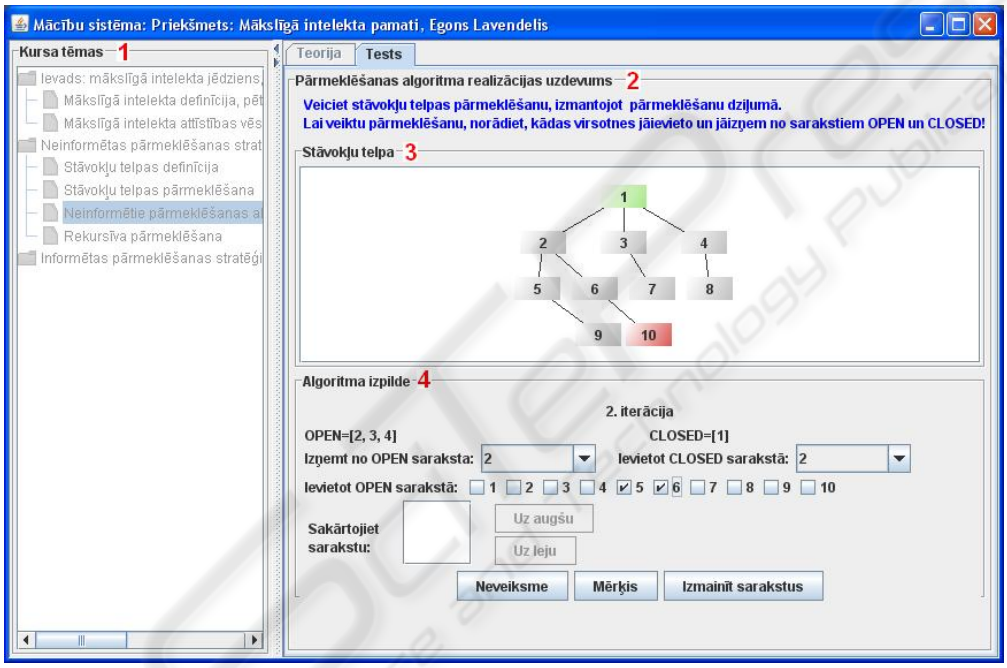

Figure 8: The interface of the MIPITS system during the problem solving step.

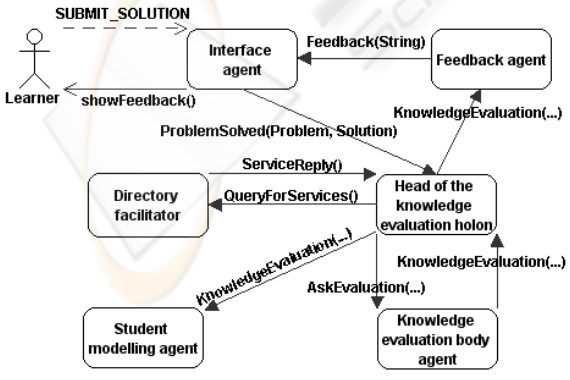

Figure 9: Interactions done during the knowledge evaluation.

5 CONCLUSIONS AND FUTURE WORK

An agent based ITS that adapts the problems to the learner's knowledge level and preferences is proposed. The adaptation of the problems is done by minimizing the difference between the preferred and real values of problem's difficulty, practicality and size. Experiments with the system showed that learners received problems that matched their preferences closer comparing with any problem that could be given to all learners.

The usage of agents and, in particular, holonic agents allow to increase the modularity of the ITS.

Each agent in the MIPITS system is responsible for concrete and separate tasks.

Additionally, agents allow creating open ITSs. The proposed system is an example how an open ITS can be implemented using the open holonic multi-agent architecture for ITS development. The system can be modified by adding or removing types of problems used in the system. The system can be used as an example to create other open ITSs that allow modifying other functionalities of the system.

There are two main directions of the future work in the MIPITS system. The first one is to add more types of problems. The concept mapping described in (Anohina et al., 2009) is the next type of the problem to be integrated into the system using the described procedure to add new types of problems. The second direction is to use open holons to implement other types of openness into the system, for example, usage of different types of learning materials is possible by implementing the teaching strategy agent as an open holon. Moreover, new types of adaptation (for different kinds of adaptation in ITS see (Brusilovsky and Peylo, 2003)) can be implemented in the system, for example, adaptation of the learning materials to a learner's knowledge level and cognitive characteristics.

ACKNOWLEDGEMENTS

This work has been supported by the European Social Fund within the project „Support for the implementation of doctoral studies at Riga Technical University".

REFERENCES

Anohina, A. et al., 2009. Concept Map Based Knowledge Assessment System with Reduction of Task Difficulty. *Proceedings of the 16th International Conference on Information Systems Development (ISD`2007)*, August 29-31, 2007, Galway, Ireland, Vol.2, Springer, pp. 853-866.

Brusilovsky, P. and Peylo, C., 2003. Adaptive and intelligent Web-based educational systems. *International Journal of Artificial Intelligence in Education* 13 (2-4), pp. 159-172.

Carbonell, J.R., 1970. AI in CAI: An Artificial Intelligence Approach to Computer-Assisted Instruction. *IEEE Transactions on Man-Machine Systems*, Vol. 11, No. 4, pp. 190-202.

Capuano, N. et al., 2000. A Multi-Agent Architecture for Intelligent Tutoring. *Proceedings of the International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet (SSGRR 2000)*, Rome, Italy.

Crowley, R.S. and Medvedeva, O., 2005. An Intelligent Tutoring System for Visual Classification Problem Solving. *Artificial Intelligence in Medicine*, August, 2005- 2006:36(1), pp. 85-117.

Devedzic, V. et al., 2000. Teaching Formal Languages by an Intelligent Tutoring System. *Educational Technology & Society*, Vol. 3, No. 2, pp. 36-49.

Fischer, K. et al, 2003. Holonic Multiagent Systems: A Foundation for the Organisation of Multiagent Systems. *Lecture Notes in Computer Science 2744*, Springer, pp. 71-80.

Gascuena, J.M. and Fernández-Caballero, A., 2005. An Agent-based Intelligent Tutoring System for Enhancing E-learning/E-teaching. *International Journal of Instructional Technology and Distance Learning*, Vol. 2, No.11, pp. 11-24.

Grundspenkis, J. and Anohina, A., 2005. Agents in Intelligent Tutoring Systems: State of the Art. *Scientific Proceedings of Riga Technical University „Computer Science. Applied Computer Systems"*, 5th series, Vol.22, Riga, pp.110-121.

Hospers, M. et al., 2003. An Agent-based Intelligent Tutoring System for Nurse Education. *Applications of Intelligent Agents in Health Care (eds. J. Nealon, A. Moreno)*. Birkhauser Publishing Ltd, Basel, Switzerland, pp. 141-157.

Lavendelis, E. and Grundspenkis, J., 2008. Open Holonic Multi-Agent Architecture for Intelligent Tutoring System Development. *Proceedings of IADIS International Conference „Intelligent Systems and Agents 2008"*, Amsterdam, The Netherlands, 22 - 24 July 2008, pp. 100-108.

Lavendelis, E. and Grundspenkis, J., 2009a. MASITS – A Multi-Agent Based Intelligent Tutoring System Development Methodology. *Proceedings of IADIS International Conference „Intelligent Systems and Agents 2009"*, 21-23 June 2009, Algarve, Portugal, pp. 116-124.

Lavendelis, E. and Grundspenkis, J., 2009b. MASITS - A Tool for Multi-Agent Based Intelligent Tutoring System Development. *Proceedings of 7th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS 2009)*, Salamanca, Spain, 25-27 March 2009. Springer, pp. 490-500.

Luger, G.F., 2005. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*, Addison-Wesley, Harlow, England, 903 p.

Smith, A.S.G., 1998. Intelligent Tutoring Systems: personal notes. - School of Computing Science at Middlesex University. http://www.cs.mdx.ac.uk/staffpages/serengul/table.of.contents.htm (last visited 18.04.05).

Virvou, M. and Tsiriga, V., 2001. Web Passive Voice Tutor: an Intelligent Computer Assisted Language Learning System over the WWW. *Proceedings of the IEEE International Conference on Advanced Learning Technology: Issues, Achievements and Challenges*, Madison, WI, USA, 6-8 August, 2001, pp. 131-134.