MASITS Methodology Supported Development of Agent Based Intelligent Tutoring System MIPITS

Conference Paper *in* Communications in Computer and Information Science · January 2011

DOI: 10.1007/978-3-642-19890-8_9

2 authors:

Egons Lavendelis
Riga Technical University
34 PUBLICATIONS **148** CITATIONS
SEE PROFILE

Janis Grundspenkis
Riga Technical University
136 PUBLICATIONS **867** CITATIONS
SEE PROFILE

MASITS Methodology Supported Development of Agent Based Intelligent Tutoring System MIPITS

Egons Lavendelis and Janis Grundspenkis

Department of Systems Theory and Design, Riga Technical University, 1 Kalku street, Riga, Latvia
egons.lavendelis@rtu.lv, janis.grundspenkis@cs.rtu.lv

Abstract. Many Intelligent Tutoring Systems (ITS) have been developed to add adaptivity and intelligence to e-learning systems. Intelligent agents are widely used in ITSs due to their characteristics such as modularity and facilitation of intelligent mechanism implementation. At the same time development of agent based ITS is complicated and methodological support is needed to enable industrial adoption of agent based ITSs. The paper describes a specific agent based ITS development methodology, named MASITS, and MIPITS system developed with the methodology. The system is created for the course "Fundamentals of Artificial Intelligence". It offers learning materials, provides practical problems and gives feedback to the learner about his/her solution evaluating his/her knowledge. The main focus of the system is on problem solving. The problems are adapted to the learner's knowledge level and preferences about difficulty, size and practicality of problems. The system offers three types of problems: tests, state space search problems and two person games algorithm problems.

1 Introduction

Nowadays, the availability of education, including lifelong learning, must be increased to successfully develop knowledge society. There is a need for easy available and individualised tutoring process that adapts to every learner. 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.

Unfortunately, e-learning as well as traditional tutoring process which is done in the classroom is not effective if many learners with different knowledge levels and learning styles are taught simultaneously. 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 realize individualized tutoring by adapting to any specific characteristics of the learner. 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. They simulate 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 [1]. During the last 40 years since the first ITS named SCHOLAR which taught geography [2], many ITSs have been developed. Well-known examples of ITSs are FLUTE [3], ABITS [4], Slide Tutor [5] and Ines [6].

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. During the last decade many ITSs have been built using intelligent agents to implement these modules (for an overview see [7]). Well-known examples of agent based ITSs are ITS for enhancing e-learning [8], ABITS [4] and Ines [6]. Despite the fact that agents increase modularity of the system and facilitate implementation of intelligent mechanisms needed in ITSs [9], development of agent based ITSs is still complicated and appropriate development methodologies are needed to bring agent based ITSs into industry. Unfortunately, general purpose agent oriented software engineering (AOSE) methodologies [10] do not take into consideration all ITS characteristics and do not allow to plug in knowledge from ITS research [7].

The paper gives a brief overview of an agent based ITS development methodology MASITS and describes an ITS for the course "Fundamentals of Artificial Intelligence" (MIPITS). The MIPITS system offers learning materials and problems to a learner, evaluates learner's knowledge in each topic and provides feedback to a learner. The system adapts problems to the learner's knowledge level and preferences.

The remainder of the paper is organized as follows. A brief description of the MASITS methodology is given in the Section 2. The Section 3 contains general description of the developed system. The architecture of the system is given in the Section 4. The tutoring scenario implemented in the system is described in the Section 5. The Section 6 shows how to extend the system with new types of problems. The Section 7 concludes the paper and gives brief outline of the future work.

2 The MASITS Methodology

To facilitate adoption of agent-oriented software by software developers a large number of general purpose AOSE methodologies have been developed during the last years. The well known examples of AOSE methodologies are Gaia [11], Prometheus [12] and MaSE [13]. However general purpose AOSE methodologies are developed to support as many different agent systems as possible by providing high level techniques that are suitable for different agents. Thus these methodologies do not allow to adjust the development process to the characteristics of specific systems. On the other hand, ITSs are specific systems with the specific characteristics that limit

effective usage of general purpose methodologies, for details see [9]. Additionally, ITSs have many important results gained in specific research that can facilitate the development process. The main results of ITS research are the following: identification of types of knowledge used in ITS, modular architecture, set of agents used to implement modules and tasks done by these agents as well as different agent architectures for ITS implementation [7], [9].

In the absence of general purpose methodologies and corresponding tools allowing to plug in domain specific knowledge and adjust the development process to the characteristics of specific systems, a specific agent based ITS development methodology, named MASITS methodology, and the corresponding tool have been developed [9], [14].

The MASITS methodology is a full life cycle methodology for agent based ITS development. The methodology includes the most important results of ITS research and reusable techniques of AOSE methodologies, which allow to integrate specific knowledge for the ITS development. Additionally, new techniques are introduced in steps where known techniques do not allow integration of ITS knowledge.

The methodology and tool support development of agent based ITSs using the open holonic architecture described in Section 4. The methodology is connected with the JADE platform (http://jade.tilab.com/) – the lower level design is done in concepts used in the JADE platform to enable direct code generation from the design.

The development process of the MASITS methodology consists of the following phases: analysis, design (divided into two stages: external and internal design of agents), implementation, testing, deployment and maintenance. The *analysis phase* consists of two steps. Hierarchy of goals is created during the goal modelling step. Use cases corresponding to the lower level goals are defined during the use case modelling step. The *external design of agents* answers the question what agents have to do. During this stage tasks are defined according to steps of use case scenarios and allocated to agents from the set of higher level agents of the architecture. Additionally, interactions among agents are designed in use case maps and the interaction diagram. Ontology is created to define predicates used in agent communications. The *internal design of agents* answers the question how agents achieve their behaviour specified during the external design of agents. Agents are designed in terms of events perceived, messages sent and received as well as behaviours carried out. The internal design of agents in the MASITS methodology is done according to the open holonic architecture for ITS development [15]. Choice of the architecture is justified in [9]. In the holonic architecture each agent can be implemented as a holon and consist of several smaller agents. For agents that are implemented as holons, design of the holon is done using techniques of both external and internal design of agents. Additionally, the MASITS methodology includes crosschecks among diagrams during the whole development process, to ensure that all elements from one diagram have corresponding elements in other diagrams [9].

Usage of open holonic architecture allows reusing some agents from previous projects during the *implementation phase*. Classes for agents that can not be reused and their behaviours as well as ontology classes are generated by the MASITS tool from the diagrams created during the design [16]. Agent and behaviour classes are manually completed after generation. *Testing phase* of the MASITS methodology is carried out in the following order, separate agents are tested first, holons as a whole

are tested second and finally testing of the whole system is done. *Deployment* of the system is specified in the deployment diagram, which is used by the tool to generate configuration of the system. The MASITS methodology supports *maintenance phase* by providing mechanisms to implement some types of changes into the system. The way how to extend system with new tasks is shown in Section 6. For further details of the development process with the MASITS methodology and tool, see [9][14].

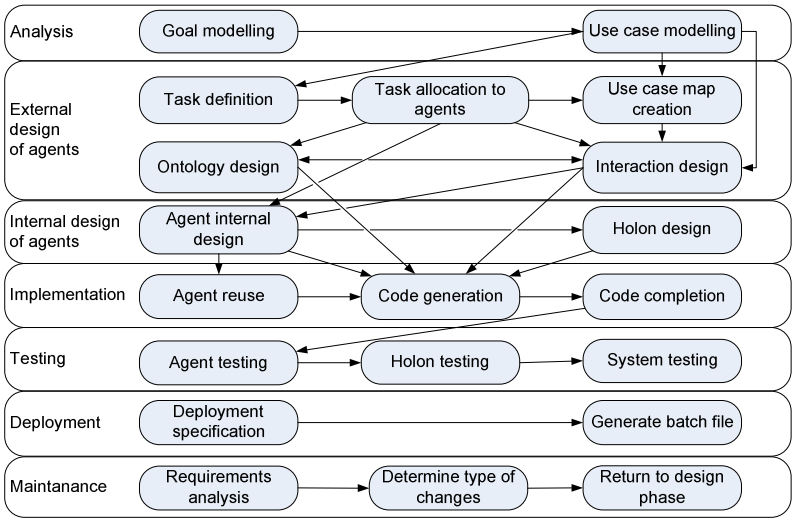

Fig. 1. Phases of the development process and steps done during each phase.

3 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 person 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 levels and learning styles. Thus another goal of the system is to 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 assignment 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" [17]. 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 [17].
- Two person game problems, where a learner has to apply the MINIMAX algorithm or Alpha-Beta pruning to the given state space [17].

Other types of problems can be added to the system (for details see Section 6). 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 delivers it to the learner.

4 The Architecture of the MIPITS System

According to the MASITS methodology, the MIPITS system is developed using open holonic multi agent architecture for ITS development described in [15]. The architecture consists of the higher level agents that implement the traditional modules of ITSs. All higher level agents can be implemented as holons [18]. Each holon consists of one head agent and a number of body agents. The head of the holon communicates outside the holon and coordinates 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 the holon. Agent instances in closed holons are specified during the design and can not be changed afterwards.

Definition of specific agents used to implement the system is started during the external design of agents. Tasks are defined and allocated to the agents in this stage. During the internal design of agents a decision is made, whether to implement each higher level agent as a holon or as a single agent. The final architecture of the system is shown in Figure 2. Heads of open holons are denoted by 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: (1) Registration. (2) Log in. (3)

Perceiving learner's requests and starting the processes in the system by forwarding learner's requests, actions and data to respective agents. (4) Giving all information to a learner, including learning materials, problems and feedback. The interface agent is the only agent interacting with a learner. Thus, it is the head of the higher level holon.

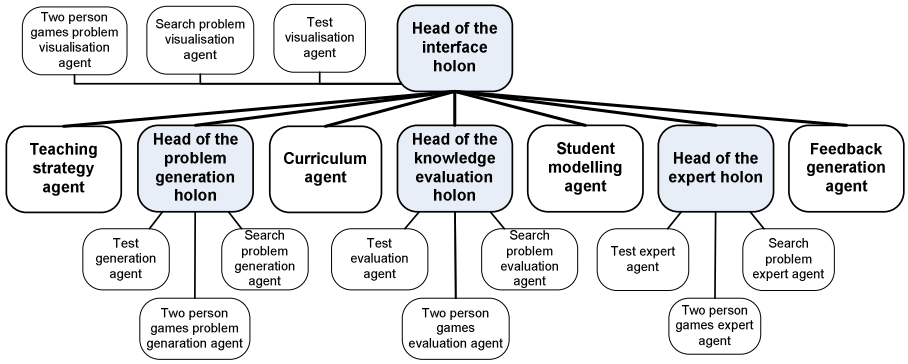

Fig. 2. Architecture of the MIPITS system.

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:

- Personal data of a learner that are collected during the registration process.
- The learner's preferences that are collected during the registration process: the preferred (initial) difficulty of problems, the preferred practicality of problems and the preferred size of problems described below.
- The curriculum that 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, namely 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: the test generation agent, the search problem generation agent and the two person games problem generation agent. Similarly, body agents of the expert holon are capable to solve problems of 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.

5 The Tutoring Scenario of the MIPITS System

The first activity a learner has to do is to *register in the system*, because a learner must be identified to adapt problems to his/her characteristics. 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 and checks data from the form, inserts user data into database and sends 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 3.

Each time a registered learner logs into 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, the interface agent validates learner's data and sends the data 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 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 4.

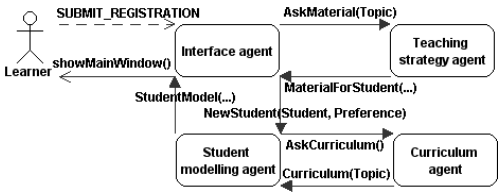

Fig. 3. Interactions done during the registration.

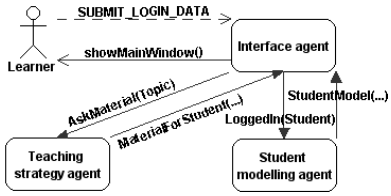

Fig. 4. Interactions done during the login.

The curriculum of the course consists of modules that, in their turn, consist of topics. To teach a topic the MIPITS system performs the scenario consisting of three steps. When a learner starts 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 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 mistakes made in each problem. When the knowledge evaluation step is over, a learner can start 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.

5.1 The Theoretical Step

The goal of the theoretical step is to hand out 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 modifies the student model by changing the status of the topic from "initial" to "started". Messages sent among agents during the theoretical step are shown in Figure 5.

Fig. 5. Interactions done during the theoretical step.

5.2 The Problem Solving Step

The goal of the problem solving step is to provide 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 6). 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, because the type of the problem does not match the topic.

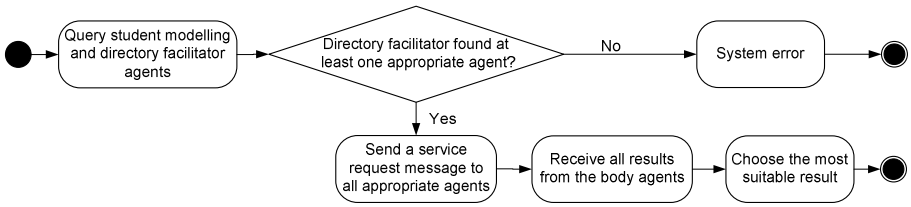

Fig. 6. Algorithm for the head of the problem generation holon.

After receiving problems from body agents the head chooses the most appropriate problem by using the following criteria, whose preferred values are calculated first:

- The degree of difficulty of the problem. The problem must match the preferred level of difficulty as close as possible, because it should not be too complex (unsolvable) nor too easy (not challenging) for a learner. During the registration process learner evaluates his/her knowledge level as an initial degree of difficulty in the scale from 1 (the lowest) to 5 (the highest). This is only a subjective his/her estimation that may be inaccurate. Thus, the system calculates the preferred degree of difficulty using the initial degree of difficulty and learner's results in previous knowledge evaluations. Moreover, the more problems a learner has solved, the more valuable is knowledge evaluation by the system and the less valuable is the value given by a learner. The preferred degree of difficulty is calculated as follows:

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

  init – coefficient denoting how much points for problem solving are equivalent to the initial degree of difficulty. Its value is determined empirically and is 50. For comparison, one test question is 2 to 4 points worth.
  dif_{init} – initial degree of difficulty.
  max – maximal number of points that can be scored for problems solved so far.
  l – level of difficulty corresponding to results that a learner has achieved in problem solving. To calculate the level, firstly, learner's result is calculated in percents of maximal number of points that can be scored for the problems solved by him/her. Secondly, the level is determined using the following empirical function: 0-34% go in level 1, 35-49% go in level 2, 50-64% go in level 3, 65-80% go in level 4, results over 80% go in level 5.
- The size of the problem. During the registration learner may choose whether his/her knowledge evaluation will be carried out with small and concrete problems or large and time consuming problems.

- The practicality of the problem. During the registration learner may 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 the learner's characteristics may lead to the situation, that only one type of problems is used and knowledge evaluation process becomes monotony. Thus, the frequency of the type of problems should be minimized. The frequency is 0 if a learner has not solved any problem yet, otherwise it is calculated by dividing the number of problems of certain type given to the learner by total number of problems given to him/her.

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

$$A = -(\left|dif_{pref} - dif_r\right| * c_d + \left|s_{pref} - s_r\right| * c_s + \left|pr_{pref} - pr_r\right| * c_p + f_t * c_f), \text{ where} \quad (2)$$

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 problem's type;
c_f – the weight of the frequency.

Weights 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 handing out the problem to a learner.
- To the expert agent, that has to find the correct solution of the problem.
- 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 use algorithm, similar to one shown in Figure 6. 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, passes the problem to a learner or solves it). The body agent of the expert holon sends the solution to the head of the holon that forwards it to the head of the knowledge evaluation holon, which saves the solution to use it during the knowledge evaluation. All messages sent among agents during this step are shown in Figure 7.

As a result of the problem solving step, the problem is handed out to a 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 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. The screenshot of the system shown in Figure 8 contains the state space search problem. The panel of the problem consists of three parts: the statement of the problem, the state space denoted with 3 and tools needed to do the search denoted with 4.

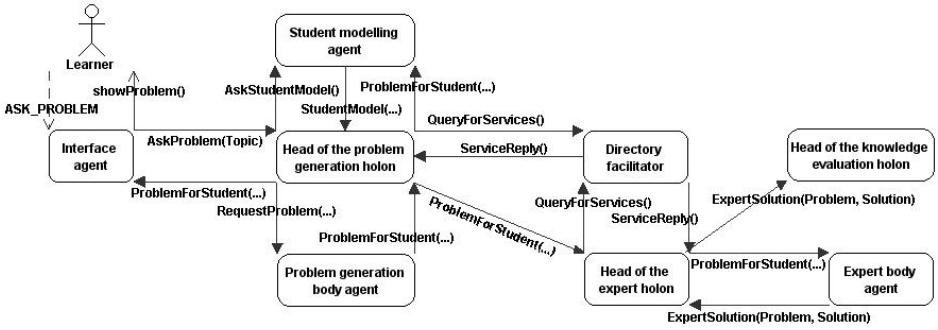

Fig. 7. Interactions done during the problem solving.

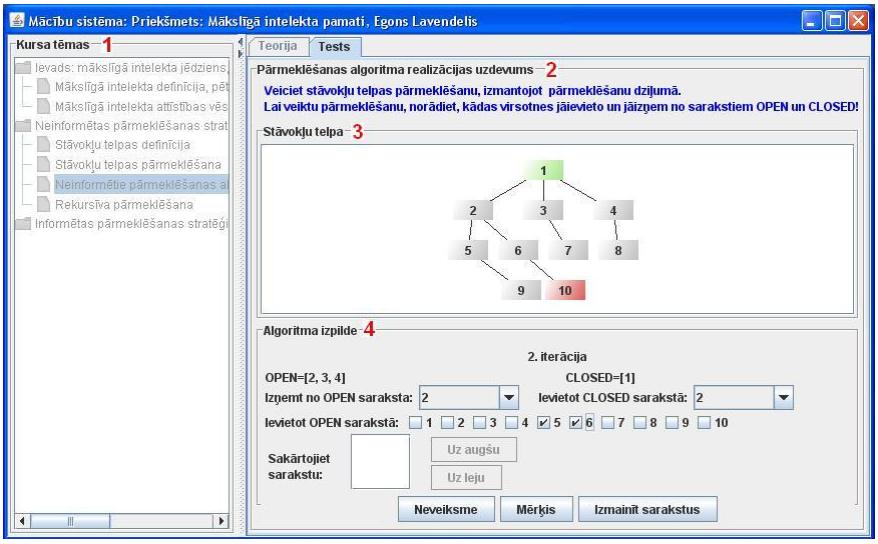

Fig. 8. The interface of the MIPITS system during the problem solving step.

5.3 The Knowledge Evaluation Step

The goal of the knowledge evaluation step is to evaluate learner's solution created in the previous step and provide him/her the feedback about the solution. A learner starts the step by submitting his/her solution of the problem. The interface agent sends learner's solution to the knowledge evaluation agent. The head of the knowledge evaluation agent uses the same algorithm as the heads of the expert and interface holons. The body agent compares the system's and the learner's solutions finding the 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 passes it to a learner. Interactions among agents done in this step are shown in Figure 9.

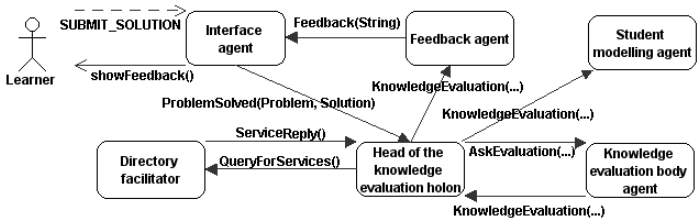

Fig. 9. Interactions done during the knowledge evaluation.

6 Extending the System with New Types of Problems

The open architecture of the MIPITS system makes it extendable for teaching new topics of the course or even some other courses by including new types of problems and appropriate materials. To enable extension of the system, 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 code of already existing agents. It can be done by adding new agents to the open holons. A single body agent has to be added to all four open holons (heads of open holons are denoted with gray colour in Figure 2): the problem generation holon, the expert holon, the knowledge evaluation holon and the interface holon.

To illustrate the extendibility of the MIPITS system a topic about propositional logic and inference is added to the system. To practice in the basic concepts of inference, a problem about usage of the Modus Ponens is added to the system. The learner receives initial working memory with one fact and the knowledge base consisting of a number of implication rules. He/she then has to show how the rules are fired and the working memory changed.

To implement this task the following changes have been made in the system:

- Four new body agents are added. Modus Ponens problem generation agent is added to the problem generation holon, Modus Ponens visualisation agent is added to the interface holon, Modus Ponens knowledge evaluation agent is added to the knowledge evaluation holon and Modus Ponens expert agent is added to the expert holon. Newly added body agents register themselves to the directory facilitator agent in order the heads of the holons be able to find them.
- The ontology is extended with class of problem about Modus Ponens and its solution.
- Information about new type of problems is added to the corresponding table in the database to link up new type of problem with corresponding topics.

7 Conclusions and Future Work

The system is an example how an ITS for individualized tutoring can be created with the MASITS methodology and a tool. The system proves that the MASITS methodology and tool are suitable for development of open agent based ITS with holonic architecture. The MASITS methodology facilitates the development process by enabling usage of the main results of ITS research, like architecture, set of agents and tasks done by these agents. The usage of holonic agents allow to increase the modularity of the ITS, where agents are responsible for concrete and separate tasks. Moreover, the system can be modified by adding or removing types of problems used in the system without changing existing code.

An agent based ITS adapting 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 than any problem that could be given to all learners.

There are two main directions of the future work in the MIPITS system. The first one is to add more types of problems corresponding to other topics. The second one is to use open holons to implement other types of openness, 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 [1]) can be implemented in the system. The main direction of the future work in the MASITS methodology is to create library of reusable agents and a basic reusable ITS ontology to facilitate reuse in ITS development.

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

1. Brusilovsky, P., Peylo, C.: Adaptive and intelligent Web-based educational systems. In: International Journal of Artificial Intelligence in Education 13 (2-4), pp. 159-172 (2003).
2. Carbonell, J.R.: AI in CAI: An Artificial Intelligence Approach to Computer-Assisted Instruction. In: IEEE Transactions on Man-Machine Systems, Vol. 11, No. 4, pp. 190-202 (1970).
3. Devedzic, V., Debenham, J., Popovic, D.: Teaching Formal Languages by an Intelligent Tutoring System. In: Educational Technology & Society, Vol. 3, No. 2, pp. 36-49 (2000).
4. Capuano, N., De Santo, M., Marsella, M., Molinara, M., Salerno, S.: 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, 2000.
5. Crowley, R.S., Medvedeva, O.: An Intelligent Tutoring System for Visual Classification Problem Solving. In: Artificial Intelligence in Medicine, August, 2005- 2006:36(1), pp. 85-117 (2005).
6. Hospers M., Kroezen E., Nijholt, A., op den Akker, R., Heylen, D.: An Agent-based Intelligent Tutoring System for Nurse Education. In: Applications of Intelligent Agents in Health Care (eds. J. Nealon, A. Moreno). Birkhauser Publishing Ltd, Basel, Switzerland, pp. 141-157 (2003).
7. Grundspenkis, J., Anohina, A.: Agents in Intelligent Tutoring Systems: State of the Art. In: Scientific Proceedings of Riga Technical University „Computer Science. Applied Computer Systems", 5th series, Vol.22, Riga, pp.110-121 (2005).
8. Gascuena, J.M., Fernández-Caballero, A.: An Agent-based Intelligent Tutoring System for Enhancing E-learning/E-teaching. In: International Journal of Instructional Technology and Distance Learning, Vol. 2, No.11, pp. 11-24 (2005).
9. Lavendelis, E., Grundspenkis, J.: MASITS – A Multi-Agent Based Intelligent Tutoring System Development Methodology. In: Proceedings of IADIS International Conference „Intelligent Systems and Agents 2009", 21-23 June 2009, Algarve, Portugal, pp. 116-124 (2009).
10. Henderson-Sellers, B., Giorgini, P.: Agent-Oriented Methodologies. Idea Group Publishing, London, p. 414 (2005).
11. Zambonelli, F., Jennings, N. R., Wooldridge, M.: Multi-Agent Systems as Computational Organisations: The Gaia Methodology. In: Agent-Oriented Methodologies, Idea Group Publishing, London, pp. 136-171 (2005).
12. Winikoff, M., Padgham, L.: The Prometheus Methodology. In: Methodologies and Software Engineering for Agent Systems. The Agent-Oriented Software Engineering Handbook, pp. 217-236 (2004).
13. DeLoach, S.: Analysis and Design Using MaSE and agentTool. In: Proceedings of the 12th Midwest Artificial Intelligence and Cognitive Science Conference, Oxford OH, March 31 - April 1 2001. pp. 1-7 (2001).
14. Lavendelis, E., Grundspenkis, J.: MASITS - A Tool for Multi-Agent Based Intelligent Tutoring System Development. In: Proceedings of 7th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS 2009), Salamanca, Spain, 25-27 March 2009, pp. 490-500 (2009).
15. Lavendelis, E., Grundspenkis, J.: Open Holonic Multi-Agent Architecture for Intelligent Tutoring System Development. In: Proceedings of IADIS International Conference „Intelligent Systems and Agents 2008", Amsterdam, The Netherlands, 22 - 24 July 2008, pp. 100-108 (2008).
16. Lavendelis, E., Grundspenkis, J.: Multi-Agent Based Intelligent Tutoring System Source Code Generation Using MASITS Tool. Scientific Journal of Riga Technical University (In Press) (2010).
17. .Luger, G.F.: Artificial Intelligence: Structures and Strategies for Complex Problem Solving, Addison-Wesley, Harlow, England, 903 p (2005).
18. Fischer, K., Schillo, M., Siekmann, J.: Holonic Multiagent Systems: A Foundation for the Organisation of Multiagent Systems. In: LNCS 2744, Springer, pp. 71-80 (2003).