Scientific Journal of Riga Technical University
Computer Science. Applied Computer Systems
2011
Volume 47

Multi-Agent and Service Oriented Architectures for Intelligent Tutoring System Development

Egons Lavendelis[1], Janis Bicans[2], [1-2]*Riga Technical University*

***Abstract* – Traditional modular architecture of Intelligent Tutoring Systems (ITSs) does not provide sufficient modularity of complex ITSs. Distributed technologies like services and agents are used to increase modularity of ITSs by implementing traditional modules as sets of services or agents. The paper describes holonic agent architecture that implements each module as one or more holonic agents. It uses the lessons learned from the multi-agent architecture to propose service oriented ITS architecture.**

Keywords: **Intelligent tutoring system, service oriented architecture, holonic multi agent system.**

I. Introduction

Nowadays many learners with different knowledge levels and learning styles have to be taught together to increase the availability of education. The traditional tutoring process can not adapt to different learners and as a consequence is not effective enough. Additionally lifelong learning becomes very important, because people need to study new things after graduation.

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 provide learning objects (mainly theoretical materials and different kinds of tests) at any place with internet connection allowing learners to choose when and where to study. Still they use the same materials and tests for all learners. Thus, traditional e-learning systems can not realize individualized tutoring. Moreover, the teacher must create all learning materials and tests that are used in the course.

To provide intelligent support of adaptive tutoring thus eliminating the abovementioned drawbacks of e-learning systems the concept of Intelligent Tutoring System (ITS) is introduced. ITSs realize individualized tutoring, using domain, pedagogical knowledge and 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 [1]. So, ITSs add adaptivity to the above mentioned benefits of e-learning systems. During the last 40 years since the first ITS named SCHOLAR and teaching geography [2], a large number of ITS have been developed. Well-known examples of ITSs are FLUTE [3], Passive Voice [4], Slide Tutor [5] and Ines [6].

Initially ITSs were developed as monolith systems, because it was the easiest way to develop a system with the needed functionality. However this approach did not allow managing complexity as ITSs became more and more complex. Therefore modular ITS architecture was widely used [7]. It consists of four modules – the pedagogical module, the expert module, the student module and the communication module. Still, four modules do not offer sufficient modularity to allow easy development, reuse of components that implement common functionality and change implementation in the system. At the same time ITSs have to implement different intelligent mechanisms to realize individualized tutoring based on knowledge about the learner [8]. The need to implement such intelligent mechanisms makes the development process even more complex.

During the last decades new distributed technologies such as services [9] and agents [10] have emerged and became popular enabling development of highly modular systems. Both of these technologies – multi agent systems and Service Oriented Architecture (SOA) have been used to implement ITSs. Agents and Multi-Agent Systems (MASs) have been widely used to implement traditional modules of ITS [6], [11], [12]. Moreover, different multi-agent architectures for ITS development have been proposed [1], [8], [12], [13]. The architectures implement the main principles of two fields – MASs and ITSs. Usage of multi-agent architectures facilitates the development process and increases modularity of the system, thus enabling reuse of different components of the system (in particular, different agents). Moreover, high modularity makes change implementation easier.

Despite the fact that services and SOA are used to implement different e-learning environments, it is not clear how one should implement a service oriented ITS and simultaneously benefit from the ITS research and main principles of SOA. Thus there is a need for specific SOAs for ITS development. At the same time MASs and SOA are similar technologies, because both of them are based on the ideas of distributed computing. The paper analyses how the lessons learned in multi-agent ITS architecture research can be reused in service oriented ITS architectures. As a result SOA for ITS development is proposed.

The remainder of the paper is organized as follows. Section 2 describes modular ITS architecture. Section 3 analyses different agent architectures for ITS development. Section 4 outlines the lessons learned in the agent based ITS architecture and presents the proposed service oriented ITS architecture. Section 5 gives a brief overview of the ITS development case study using the proposed service oriented ITS architecture. Section 6 concludes the paper.

II. Modular Intelligent Tutoring System Architecture

Despite the fact that every individual ITS is implemented to teach different subject and to realize different functionality, all ITSs are built with the same goal, respectively, to simulate a human tutor during the learning process or to adaptively teach certain domains or subjects to the learners. To do it, ITSs have to generate curriculum for a particular learner to learn the course, realize different teaching strategies, generate tasks, problems and questions to evaluate learner's knowledge as well as give feedback to explain learner's mistakes and offer appropriate remedial actions. All these activities have to be done in adaptive manner to implement one-to-one tutoring and meet the needs of different learners. To carry out these activities ITSs use three types of knowledge, namely, knowledge about what to teach (domain knowledge), how to teach (pedagogical knowledge) and knowledge about the learner [11].

After declining monolith architectures of ITS it was realized that the architecture of ITS have to fit the types of knowledge used in ITSs, to separate different perspectives needed in ITS. As a consequence a modular architecture was introduced. It consists of three modules named "traditional trinity" [7], [11]:

- The *expert module* contains the domain expert knowledge and is responsible for problem solving in the domain taught by the system. It is used as a standard which learner's knowledge and actions are compared to.
- The *student module* collects knowledge about the tutoring process, learner's conceptions and errors made (misconceptions), as well as knowledge about learner's preferences and learning style.
- The *tutoring module* contains pedagogical knowledge about the way to teach. It holds teaching strategies and instructions to implement teaching process. The primary tasks of this module are controlling selection, sequencing and presentation of learning material that is most suitable for the needs of the learner, determining the type and contents of feedback and help, as well as answering learners' questions. The goal of this module is to reduce the gap between expert's and learner's knowledge levels.

Additionally, the fourth component – the *communication module* is responsible for communication with the learner. The modular architecture consisting of four modules is shown in Figure 1.

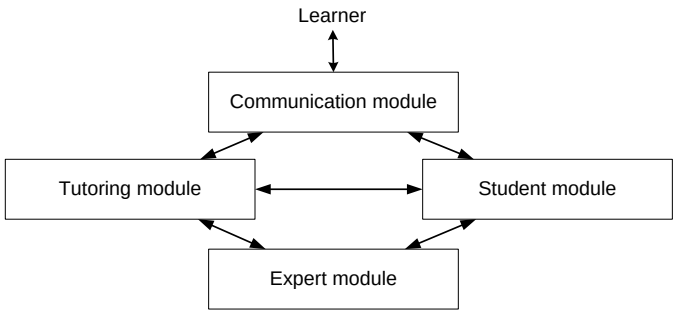

Fig 1. Modular ITS architecture [7]

ITSs are built using various implementation technologies, starting from traditional object-oriented approaches and finishing with SOA and MASs. At the same time, majority of well known systems, for example, Ines [6], AlgeBrain [14], and FLUTE [3] include clearly identifiable traditional modules. These modules are used, because they allow separating components that are working with completely different knowledge. Thus, the usage of the traditional modules has a clear advantage that should be used developing ITSs with any technology.

On the other hand, such technologies as MASs and SOA use decentralized approach. As a consequence, if an ITS is built using these technologies, there is a need for an architecture that keeps the modules corresponding to the knowledge types and at the same time adds ideas from highly modular and decentralized computing. The following chapters analyse how it can be done using intelligent agents and services.

III. Multi-Agent Architecture for ITS Development

During the last decade agents and MASs have been widely used to implement ITSs. Agents are suitable for ITS development due to their characteristics like reactivity, proactivity and communication capabilities [8]. Agents are considered to be natural technology to implement intelligent mechanisms, which are needed in ITSs. Several researches proving that agents are suitable for ITS development can be found in [11], [15]. Moreover, intensive research in the agent-based ITSs field has been carried out during the last decade and numerous systems have been built. Well known examples of such systems are: Ines system for nurse education [6], Formal Languages and aUTomata Education system FLUTE [3], WADIES – a Web- and agent-based adaptive learning environment for teaching compilers [16], Fundamentals of Artificial Intelligence teaching system MIPITS [17].

Majority of the abovementioned systems use the same approach. The system consists of the traditional four modules, which are implemented as one or more agents. Grundspenkis and Anohina [11] have analysed the usage of agents for ITS implementation and have proposed a set of agents that are used to implement traditional components of the ITS. Each module is implemented as a set of agents that may consist of one or several agents. Every individual system may contain all agents from the proposed set or just those agents that are needed to implement the functionality included in the system.

Student model agents and their tasks are the following. A knowledge evaluation agent carries out knowledge assessments and builds a model of learner's current level of knowledge. A psychological agent builds learner's psychological profile. A cognitive diagnosis agent determines and registers learner's mistakes. Interaction registering agent registers history of learner's interaction with the system. *Tutoring module* agents and their tasks are the following. A curriculum agent evaluates, generates, and updates the individualized curriculum for the specific learner. Each tutoring strategy agent realizes one or multiple teaching strategies. A feedback agent generates feedback, hints and help for the learner. *Expert module* agents named expert

agents solve domain specific problems and tasks. There may be one or more expert agents in the system.

Some multi agent architectures have been proposed, too. Majority of these architectures extend the described set of agents and mainly consist of previously mentioned agents. Examples of such architectures are ABITS [12], IVET [13], X-genitor [18] and two level multi-agent architecture for distance learning environment [15].

However, the majority of agent based architectures for ITS development are not open. Thus they do not support easy change implementation [1]. These architectures consist of a small number of large scale agents, too. As a consequence they do not offer so high modularity as such technologies as MASs and services can offer.

Open holonic architecture for agent based ITS development has been proposed using the above mentioned set of agents [1]. Its main goal is to add openness and increase modularity of agent based ITSs. The architecture uses holonic MAS approach in the sense that each agent may consist of several smaller agents, so creating hierarchical structure with unlimited depth. As a consequence, the system is highly modular. The higher level holon consists of the abovementioned set of agents used to implement modules of ITS. Each module is implemented as one or more subholons. The interface agent implements the communication module and is the only agent that communicates with the learner. Thus it is the head of the higher level holon. The student module is realised as student modelling agent and knowledge evaluation agent. The tutoring module is realised as curriculum, teaching strategy, problem generation and feedback generation agents. The expert module is implemented as a single expert holon. For further details about the higher level holon agents and interactions among them see [1].

Each agent from the higher level holon is realized as a holon consisting of a single head agent and unlimited number of body agents (which may be next level holons if needed). A set of predefined tasks is assigned to each higher level subholon. The head of each holon is responsible for communication outside the holon and for accomplishing holon tasks. Some tasks are physically done by the head of the holon, but for others body agents are used. Each holon includes body agents that are capable to perform certain tasks which are subtasks of the tasks assigned to the holon.

The proposed architecture is open in the following way. Two types of holons are distinguished. Closed holons contain only a set of concrete agents. Agents in the closed holons have principally different capabilities. Body agents in closed holons may vary from one ITS to another one, but they have to be specified during the design. Open holons consist of a head and body agents of a certain type. Agents in the open holon have capabilities of the same type, for example generation of different kinds of problems. The number of concrete agents and their capabilities can be changed during the runtime of the system. Open holons allow modifying the functionality of already implemented and running system. The head of the holon uses directory facilitator agent to find body agents that are capable to accomplish certain tasks. Heads of open holons have the role of mediator between agents that are using services provided by the holon and the body agents of the holon. Usually the head just finds the most appropriate agent or agents that do the needed task and forward the result. For example, the main expert agent (the head of the expert holon) finds the body agent that is capable to solve problems of certain type and uses it to find the solution. The problem generation agent acts differently. It requests all body agents to generate tasks and then chooses the most appropriate. Other heads use similar principles, for details, see [1].

The generalized holonic multi-agent architecture for ITS development is given in Figure 2. The figure shows what types of agents can be used in each second level holon. Rectangles with shadows denote typical body agents. For example, problem generation holon is an open holon that contains one body agent for each type of problems that are included in the system. Such approach allows to add (remove, change as well) new types of problems by just adding new agents to the open holons without changing existing code. For details about agents included in each holon and communications in open and closed holons as well as the algorithms used by heads of holons see [1].

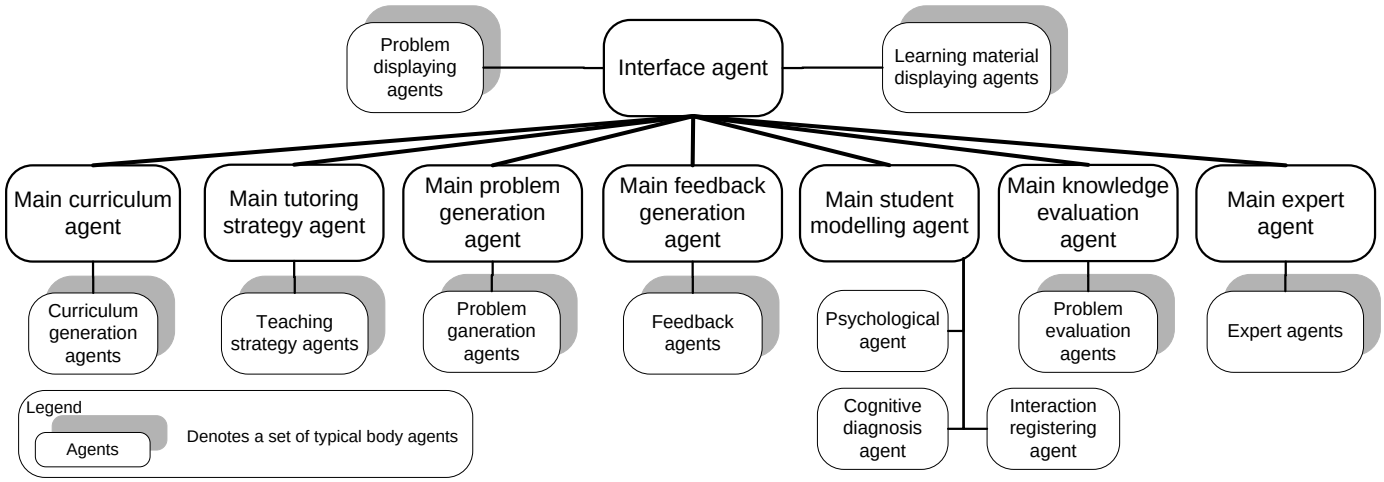

Fig. 2. Generalized holonic multi-agent architecture for ITS development

The holonic multi-agent architecture has been used to implement an ITS named MIPITS and teaching "Fundamentals of Artificial Intelligence" to undergraduate students. Firstly, the system gives brief learning materials on each topic. Secondly, it provides different kinds of problems to evaluate learner's knowledge of the topic. The problem solving is the main focus of the system. It provides different kinds of problems and tries to adapt problems to learners' knowledge level and other characteristics. Moreover, new types of problems can be easily added to the system by just adding agents that can generate, show to the learner, solve and evaluate these problems. For further implementation details and tutoring scenario implemented in the MIPITS system see [17].

IV. Service Oriented architecture for ITS Development

Multi-agent and service technologies have been introduced by researchers of two different fields. Agents have been proposed by artificial intelligence researchers and services have been proposed by software development researchers. Nevertheless, these two technologies are similar in the sense that they implement highly modular systems based on the principles of distributed computing. As a consequence agents and services have the following similarities:

- High modularity. Both SOA and MASs offer high modularity, because they support systems that consist of small distributed entities.
- Openness. Both technologies allow dynamically changing the system. Agents and services can be added to and removed from the system. Dynamically added agents have directory facilitator agent service. Registry service finds the dynamically added services to the SOA.
- Reactive agents can be considered as services. For example lower level agents from the holonic agent architecture provide specific service upon request.
- Usage of protocols. Both agents and services use some kind of protocols to interact.

Still, agents and services have some differences that have to be taken into consideration when choosing the technology to implement ITS:

- Agents have reasoning capabilities. Agents are considered as reasoning entities, for example BDI (Belief, Desire, Intension) agents reason in terms of beliefs, desires and intensions [20]. At the same time, services have no built in reasoning capabilities.
- Services are strictly reactive. They have no autonomy, while agents are autonomous and are capable of proactive actions.
- SOA is a more commonly and industrially accepted standard while agents are used more in research projects and have not yet come into industry.

So one can conclude that both agents and services can be used to implement highly modular and open systems. Services are more common for industrial products, while agents offer built in reasoning components and are capable of proactive actions. As a consequence, if there is no need for proactive components in ITS one can use services instead of agents to implement ITS and benefit from the industrial standards of SOA. Actually, services may be used in the same way as agents to implement modules of ITS. The remainder of this section presents principles that must be used if the modules of ITS are implemented as services and the developed SOA for ITS development.

A. Layered architecture

Different software architecture paradigms state that user interface must be separated from the logical part of the system [19]. It can be done by creating layered architecture, where interface is included in the separate layer. Additionally ITSs usually interact with different learning object (LO) repositories and fundamental technologies, like video streaming. So we can get a layered architecture consisting of three layers (see Figure 3):

- Layer one contains learning object repositories and may contain such technologies as video streaming.
- The second layer or logical layer contains the student module, the tutoring module and the expert module. This layer is responsible for implementation of adaptivity in the system.
- The third layer is the presentation layer that contains all technologies that are needed to present the contents of ITS to the learner. This layer contains the communication module of the ITS.

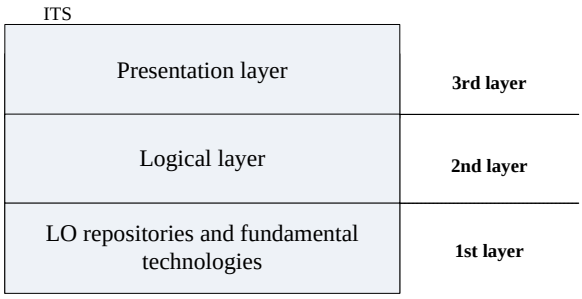

Fig. 3. Layered ITS architecture

B. Lessons learned from agent architecture

Implementation of multi-agent architecture for ITS development showed that modules should be implemented in open manner to allow adding new functionality just by including new agents in the system. In our case study it allowed adding new types of problems to the MIPITS system [17]. The openness was implemented with open holons.

Implementation of holonic agents allowed increasing modularity and decreasing coupling of the system, because body agents do not interact with any agents outside the holon. The head of the holon serves as an interface of the holon. These two characteristics of multi-agent architecture can be sustained in the service oriented ITS architecture in the following way. Each module contains one (the expert module) or more (the tutoring module, the communication module and the student module) main services that have the role of interface of the module, like the heads of the corresponding holons. The interactions in the system are organized in the

following way. First, the main service receives service requests from other main services. Second, it uses service registry to find other services of the module and forwards the request to the appropriate service. The appropriate service does its job and returns the result to the main service, which forwards it to the initial requester. So interactions take place only among main services and among main services and services of the corresponding module. For example, the communication module receives learner's request for a problem in a certain topic. The request is sent to the main problem generation service, which finds the service that is capable to generate a problem in the topic or even the most suitable problem for a particular learner.

As a consequence, there are no interactions among other services than main ones of one module with services from another module. It decreases the coupling of the system and facilitates its modularity. Additionally, the communication module must be open (like the interface agent holon is open), because if there is new functionality in the system, then it must be presented to the learner, otherwise it is useless.

If the multi-agent architecture shown in Figure 2 is compared to the layered architecture shown in Figure 3, it can be concluded that these architectures match each other well. The interface agent holon shown in the first row of Figure 2, has the same functions as the presentation layer. As a consequence, services implementing this layer have to realise all interactions with the learner.

During the implementation of the multi-agent architecture it appeared that the majority of adaptation mechanisms are included in the heads of the holons – they are not only the mediators, but they make intelligent choices, based on different algorithms and reasoning to provide adaptivity. For example, body agents of the problem generation holon are capable of problem generation, but the head of the holon chooses the most appropriate problem for the learner. So, it is concluded that the body agents mostly implement certain actions with the repositories, like extracting or generating learning objects. These agents mostly work with the repositories and show only reactive behaviour. As a consequence, it may be concluded that the heads of the higher level holons mostly fit the logical (the second) layer of the ITS, while the body agents of these holons realise interface among the second and the first layer, because these agents are used by the agents from the second layer to access the first layer – the repositories. Nevertheless, one should note that this division among layers is true only if very basic multi-agent architecture is used. The multi-agent architecture is customizable to include different intelligent mechanisms to the body agents of the holon. If this is done, the body agents also should be assigned to the logical part of the ITS and as a consequence to the second layer.

Additionally, from certain perspective the body agents of the lower level fit the concept of the service more than an agent, because they are not proactive and show only reactive behaviour by executing some tasks upon the request of the head of the holon. Of course, from another perspective agents provide several advanced communication mechanisms among the head of the holon and body agents. This option is lost if the system is realised using SOA.

Moreover, agent architecture has more features that can not be implemented using services. Firstly, by implementing all modules of ITS as services instead of agents one has to give away all reasoning mechanisms that are built in intelligent agents and natural to them. Thus it is not clear how to implement such mechanisms as BDI reasoning inside a service. As it was concluded intelligent mechanisms are needed mostly in the logical layer of the ITS. Such mechanisms are rare in other layers, namely presentation layer and access to repositories. As a consequence two layers of ITS can be built by replacing agents with services. To realise the second layer the developers have to implement all needed intelligent mechanisms from scratch inside the services. Potentially this is the main disadvantage in moving from multi-agent to service oriented ITS architecture. Additionally, agents allow easy implementation of proactivity – agents can start processes in the system, while services can not. If some proactive behaviour is needed, then usage of agents is preferable.

C. The proposed service oriented ITS architecture

The proposed service oriented ITS architecture provides a set of services to implement each module of ITS. Each module contains one or a few main services that are used from the outside of the module and unlimited number of other services that are not used from the outside of the module. So the proposed architecture consists of two levels, too. Services that are used from other modules are named the main services of certain type and included in the higher level of the architecture, while other services are included in the lower level. The levels of the proposed SOA match the layered architecture similarly to the holonic agent architecture. All services of the communication module correspond to the third layer. All main services of the pedagogical module, the student module and the expert module correspond to the second level and all lower level services from these modules can be considered as interfaces between the first level (repositories) and the second level (logical level), because they are used to retrieve and store data in the repositories.

As in the modular architecture, the only module that interacts with the learner is the *communication module*. The module is responsible for showing curriculum of the course, the visualisation of LOs, giving feedback to the learner and receiving all learner's requests. Two types of LOs are distinguished in the architecture: all theoretical materials and examples are considered as one type and problems to be solved by the learner are considered as another type of LOs. Such a division is made, because these two types of LOs need different approach in the system. Theoretical materials need just to be displayed, while learner's solution of the problem has to be saved and used for knowledge evaluation and feedback generation. The module contains five main services:

- The main interface service that is responsible for registering all learner's actions and forwarding them to the appropriate services, for example, when it receives

request for a problem in a certain topic, it forwards the request to the main problem generation service.

- The main material visualisation service handles requests for visualisation of certain types of materials.
- The main task visualisation service. It is similar to the material visualisation service, only it visualizes problems instead of materials.
- The curriculum visualisation service that shows the curriculum of the course. It can be done differently, for example as a list of topics or as a topic map like in the case study described in Section 5.
- And the feedback visualisation service. It has only one function – to visualise feedback.

Additionally, the module contains two types of specific lower level services that do not interact with any other service than the corresponding main service. The main material visualisation service has one lower level service for each type of materials that is needed to be visualised differently. For example, if the system offers materials as HTML documents, PDF documents and video streams, then there are three corresponding visualisation services. Similarly, the main problem visualisation service has lower level services corresponding to each type of problems. The main material and problem visualisation services receive requests for visualisation of certain types of materials and problems and just find the corresponding lower level service to forward the request to. All lower level services of the communication module have one function – visualisation of certain type of material/problem.

The communication module can be implemented as a web application and interactions among this module and other parts of the system can be done through the firewall to enhance security of the system.

The *pedagogical module* has to generate the curriculum of the course, materials and problems in each topic of the course as well as to provide feedback to the learner. Thus, it contains the following main services:

- Curriculum generation service that provides curriculum of the course which may be created by the teacher or generated by the service.
- Main material generation service, which is responsible for material generation on each topic. Additionally, second level services are added corresponding to types of materials. For example, separate generation services can be created for each of the abovementioned different formats of learning materials. When the main service receives generation request, it requests student model from the student modelling service described below. When it receives the student model, the request for materials on the certain topic together with needed characteristics is sent to the second level services. The second level services create or retrieve from the repository the most suitable materials for the learner's characteristics on the current topic. The second level services send materials to the main material generation services. The main service chooses the most appropriate material and forwards it to the main interface service.
- Main problem generation service, that is responsible for problem generation. The architecture supports generation of different types of problems. Each type of the problem is generated by corresponding second level service. Interactions among services are identical with the interactions during the material generation described above.
- Feedback generation service, which receives knowledge evaluation and provides feedback to the learner. The feedback in different systems can vary from just comments about the correctness of learner's answer to the detailed explanation of mistakes.

The *expert module* consists of one higher level service named main expert service. It is responsible for solving problems given to the learner. Solutions provided by this service are named system's solutions and used during the knowledge evaluations to compare to student's solutions. Each type of problems has its own second level service that solves the problem. The main service just has to find the correct second level service.

The *student module* consists of two higher level services: the main knowledge evaluation service and the main student modelling service. The student modelling service collects information about learner's actions, topics learned, problems solved by him/her as well as learner's preferences and his/her knowledge evaluations. It provides full student model upon request of any other service. The main knowledge evaluation service is responsible for knowledge evaluation by comparing system's and learner's solutions and finding learner's mistakes. Knowledge evaluation with each type of problems is done by separate second level service. The main knowledge evaluation service just has to find the second level service that is capable to carry out the knowledge evaluation.

To illustrate workflows that are executed in the ITS with the proposed architecture, the following example can be considered. The system provides three types of problems, namely, tests, some kind of practical problems and some kind of theoretical problems. The main interface service receives learner's request for problem in the specific topic. This request is forwarded to the main problem generation service, which then uses student modelling service to get the student model data about problems that the learner prefers. Then it uses service registry to find all services that are capable to generate problems and send requests for problem generation. These requests include information about the preferred problem. The problem generation services create problems and forward them to the main problem generation service, which chooses the most appropriate one and forwards it to the main interface service that finds appropriate service to visualize the problem. At the same time problem is forwarded to the expert service to provide system's solution that will be used as an etalon for learner's solution.

The proposed architecture is open in the following sense. Existing services do not have to be changed to add new types of materials or problems. Only new services corresponding to the new type of problem or material must be added in each component of the architecture where each type of

material/problem is handled by separate service. If any new types of materials are added to the system (for example RTF documents) then two services must be added to the system, namely corresponding lower level material generation service and lower level material visualisation service. Addition of new types of problem requires four new lower level services to be added, namely, corresponding problem generation service, expert service, knowledge evaluation service and problem visualisation service.

The discussed openness is not the only way to modify the architecture. Different new modules needed in specific ITSs can be added to the traditional modules. For example separate module (the system evaluation module) can be added to collect learner's opinions about the system. Such a module contains two services – one for collecting evaluations and one for analysing them. Moreover, many real ITSs need to give the teacher access to the system. Teachers interface has separate main interface service that manages his/her interface, results retrieving service and services that allow creating/retrieving/editing topics of the course and all kinds of LOs used in the ITS. If the system has two distinct modules for learners and teachers, the separate authorisation service is needed that carries out the login process and assigns correct role depending on the provided login information. The service oriented ITS architecture with the described extensions is given in Figure 4. The figure contains repositories used to store data, but links among services and repositories are omitted to keep the figure readable. The following repositories can be used in the ITS:

- Student data repository that contains both personal data and student models of all registered learners.
- Course/topic repository that contains data about courses taught by the ITS and topics that the courses consist of. Additionally logical links among courses and topics can be included.
- LO repository that contains all materials used in the system and metadata describing them.
- The problem repository that contains problems provided to the student.
- The system evaluation repository contains system's evaluations provided by the learners.

The proposed architecture is built in the way that all intelligent mechanisms needed in ITS to implement adaptivity, such as curriculum sequencing, choice of materials and problems, as well as feedback generation is done in the higher level services or main services. However, particular ITSs may not need intelligent mechanisms in each higher level service. In this case the main service may be omitted. Then services that use the main service in the proposed architecture use service registry to find appropriate lower level service and use the lower level service directly.

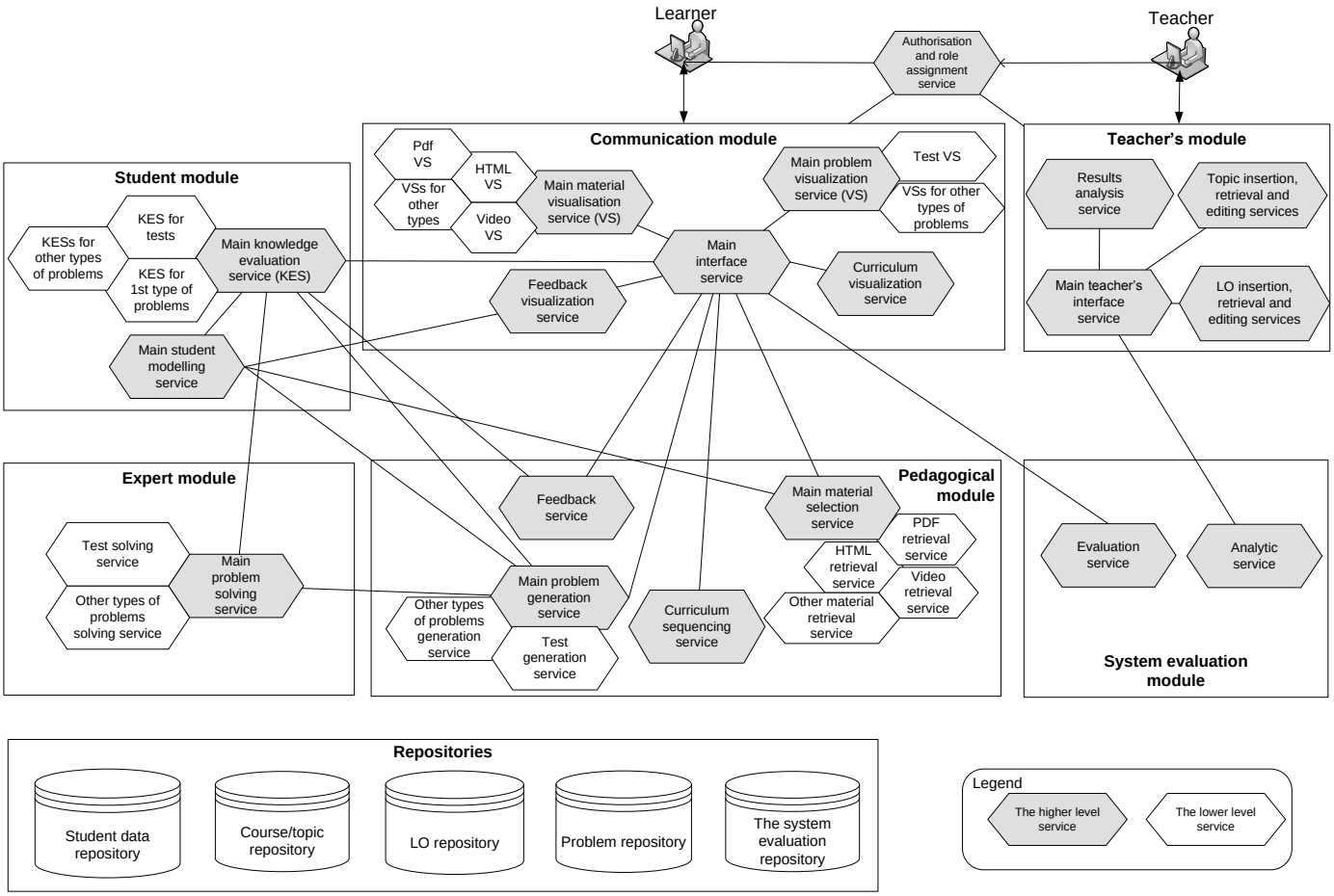

Fig. 4. The proposed service oriented architecture for ITS development

V. Case Study

The developed ITS prototype is using the LO approach. *"LOs are defined here as any entity, digital or non-digital, which can be used, re-used or referenced during technology supported learning. Examples of LOs include multimedia content, instructional content, learning objectives, instructional software and software tools, and persons, organizations, or events referenced during technology supported learning"* [21]. LOs describe any chunk of decontextualized learning information. They can be digital or non-digital. Typical LOs are images, videos, text documents, educational game or sound files. The aim of those entities is to provide learning knowledge to the student. LOs are described with metadata. Pedagogical strength of LO approach is that it supports concepts originating from different learning theories such as flexibility theory, constructivism and learning style theory. This approach gives control over the student's cognitive needs and allows the tutor or ITS to present to the student suitable objects that will fit his/her expectations. Traditional LOs like text and presentations support only students whose learning style prescribes visual information observation and processing. Text LO cannot support students whose learning style prescribes audio or audiovisual information digestion. Learning materials in video format are used to eliminate these drawbacks.

The ITS is developed for the course "Artificial Intelligence" taught at Riga Technical University. This course covers main artificial intelligence topics, such as agents, ontologies, reasoning, etc. The course has video and text LOs with different levels of granularity. LOs cover theory, examples and tasks in the topics of the course. The aim of the developed ITS is to support adaptive out of classroom learning for students with different learning styles presenting single or a combination of the LO on the given topic. The main focus of the developed ITS is to store and identify different kinds of LO describing them with standardized metadata [21]. Therefore, such an approach provides ITS with option to use more LOs from different repositories- self-held and partner. To support video LOs, mechanisms for LO metadata and video time code markers integration were proposed, i.e., the problem was how to link either a part or the whole video with LO metadata and with ITS.

The case study proved that the usage of SOA provides flexibility with the mixing of technologies, because it uses standardized protocols for information exchange. It is important for system future evolution, i.e., agent technology integration. SOA architecture is flexible for use where different methods of instructional adaptation are implemented. The developed ITS has services that are grouped into logical modules where each logical module is presented with set services. The master service serves as a communication interface between different components of the system. The system has a topic map with curriculum structure and provides intuitive navigation. This approach shows to a student where a topic of interest is placed, i.e., it draws on a net of topics and supports overall understanding of a concept and/or role within the curriculum. Course curriculum generation process consumes "Curriculum sequencing service", "Curriculum visualization service" and "Authorization and role assignment service". The topic map and LOs are not connected together using references. The ITS has a built in algorithm for information extraction. This algorithm does topic and LOs keywords intersection and obtains a set of objects that are connected with this topic. In the next step this algorithm presents only those LOs that support the learner's learning style, i.e., for users who better accumulate knowledge by reading theory it provides theory text files and video, for those who better accumulate knowledge by solving tasks or viewing examples it provides tasks or examples. This algorithm consumes the following services: "Main interface service", "Main Material visualization service", "Authorization and role assignment service", "Main problem visualization service" and "Main material selection service". The mentioned services are a part of service hierarchy and use other lower level services which do operations with databases and video streaming technologies.

The ITS was developed with Microsoft technologies, i.e., Silverlight Media Framework (SMF), SQL server 2008 R2, Windows server 2008 R2 and ASP.NET. SMF directly support video streaming with various bit rates, therefore, the student will receive the highest available video quality depending on internet connection latency, i.e., the learner will not experience buffering and therefore will not waste time on activities which are not connected with tutoring. Moreover SMF provide video splitting without doing it physically, i.e., the whole video learning object may be split into multiple smaller objects therefore providing reusability. This approach simplifies video LO generation and possible video LO integration from other resources.

The developed ITS was approbated with master students at Riga Technical University. Approbation feedback showed that granularity of LOs should vary, because some students like LOs with low granularity, while others like high. Different levels of granularity can be achieved by enabling the ITS to generate combinations of learning objects. Video LOs were used frequently and on average each student viewed 90% of presented videos at least once.

VI. Conclusions and Future Work

Both of the analyzed distributed technologies, namely MASs and SOA can be used in the same way to implement ITSs – each module of the traditional ITS architecture can be implemented as a set of smaller components. The service oriented ITS architecture is proposed in the paper using lessons learned in agent based ITS development. As a consequence the proposed architecture allows using industrially accepted technology in the ITS development and still has such advantages as openness and high modularity.

The described ITS architectures have high modularity and are open and customizable. They can be justified to match the needs of each system by adding new components or removing the ones that are not needed in the particular system. Moreover, new types of materials and problems can be added

to already implemented systems without changing the existing code. The developed architectures also facilitate reuse of small scale components in different systems. For example, components (agents or services) dealing with the same kind of materials (for example PDF documents) can be reused in new systems where such materials are used. On the other hand technologies to implement SOA are much more widely used in the industry, thus developers do not have to study new environments to use SOA in ITS development. For, example Microsoft .NET framework can be used.

Both architectures have been successfully used to develop working prototypes of open ITSs, thus proving that the developed architectures are suitable for development of open systems. The developed systems were approbated at study courses. Additionally, the openness of the architecture has been verified by adding new types of problems to the MIPITS system [22].

The proposed service architecture allows separating non intelligent services from services that implement intelligent mechanisms and are the main subject of future research. The lower level services of the proposed service architecture provide certain functionality upon request and mainly do not include any intelligent mechanisms. These services just retrieve data from repositories, carry out some generation algorithms or show something to the learner. All intelligent mechanisms that implement adaptive tutoring in the ITS are included in the higher level services and mainly in the higher level services of the tutoring module, the student module and the expert module.

The main directions of the future work are the following: firstly, we intend to improve the prototype of service oriented ITS developed as a case study. It is planned to implement more detailed student model and improve learning object adaptation algorithms to take into consideration more student's characteristics like knowledge level and different preferences. Secondly, it is planned to study how the two technologies described in the paper can be used together in the hybrid architecture. As it has been concluded above, some of the components of the ITS provide simple services (like visualisation of learning object) while others have to make complex decisions based on multiple criteria (like choice of suitable learning object or evaluation of learner's knowledge). It is planned to develop a framework consisting of both agents and services. Agents will implement intelligent components of ITS while services will implement routine tasks. In such a framework agents choose among the right services and execute them when needed.

REFERENCES

[1] E. Lavendelis, and J. Grundspenkis. "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.

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

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

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

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

[6] M. Hospers et al., "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, 2003. pp. 141-157.

[7] A.S.G. Smith, "Intelligent Tutoring Systems: personal notes". School of Computing Science at Middlesex University. - 1998. - http://www.cs.mdx.ac.uk/staffpages/serengul/table.of.contents.htm [Accessed April 18, 2005].

[8] E. Lavendelis "Open multi-agent architecture and methodology for intelligent tutoring system development". Summary of Doctoral Thesis. -R.:RTU, 2009, 49 p.

[9] N. Josuttis: *SOA in Practice. The Art of Distributed System Design*. O'Reilly Media, 2007.

[10] M. Wooldridge *An introduction to Multiagent Systems.* Chichester, England: John Wiley & Sons, 2002, 348 p.

[11] J. Grundspenkis and A. Anohina. "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, RTU Publishing, Riga, 2005, pp.110-121

[12] N. Capuano et al., "A Multi-Agent Architecture for Intelligent Tutoring". Presented at the International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet (SSGRR 2000), 2000, Rome, Italy.

[13] A. de Antonio, et al. "An Agent-Based Architecture for the Development of Intelligent Virtual Training Environments." In Proceedings of the Second International Conference on Multimedia and Information and Communication Technologies in Education (m-ICTE 2003), Badajoz, Spain, December 3-6, 2003, p.1944-1949.

[14] S.R. Alpert, et al. "Deploying Intelligent Tutors on the Web: An Architecture and an Example". In: International Journal of Artificial Intelligence in Education, 1999, Vol. 10, No. 2, pp.183-197.

[15] C. Webber and S. Pesty "A two-level multi-agent architecture for a distance learning environment". In ITS 2002/Workshop on Architectures and Methodologies for Building Agent-based Learning Environments, E.de Barros Costa, 2002, pp.26-38.

[16] K. Georgouli, et al. "A Web Based Tutoring System for Compilers." In Proceedings of the 14th EAEEIE Annual Conference on Innovation in Education for Electrical and Information Engineering (EIE), Gdansk, Poland, June 16-18, 2003

[17] E. Lavendelis and J. Grundspenkis "MIPITS - An Agent Based Intelligent Tutoring System". In Proceedings of the Second International Conference on Agents and Artificial Intelligence (ICAART 2010), Valencia, Spain, January 22-24 2010. pp. 5-13.

[18] T. Triantis and P. Pintelas "An Integrated Environment for Building Distributed Multi-agent Educational Applications". In C. Bussler and D. Fensel (Eds.): AIMSA 2004, LNAI 3192, 2004, pp. 351–360.

[19] G.V. Bochmann. „High-level design for user and component interfaces". In Knowledge-Based Systems, 17(7-8), 2004 pp. 303-310.

[20] D. Kinny et al. "A Methodology and Modelling Technique for Systems of BDI Agents". In Agents Breaking Away, 7th European Workshop on Modelling Authonomous Agents in a Multi-Agent World. Eindhoven, The Netherlands, LNCS, Vol. 1038, Springer, 1996. pp. 56-71.

[21] "Draft Standard for Learning Object Metadata", IEEE 1484.12.1-2002

[22] E. Lavendelis and J. Grundspenkis "MASITS Methodology Supported Development of Agent Based Intelligent Tutoring System MIPITS" In "Communications in Computer and Information Science" (CCIS), Springer, 2010 (accepted for publication).

Egons Lavendelis has studied at the Faculty of Computer Science and Information Technology of Riga Technical University Riga, Latvia since 2001 and received Dr.sc.ing in 2009. The topic of the doctoral thesis was "Open multi-agent architecture and methodology for intelligent tutoring system development"
He started to work as a researcher in 2005. Since 2010 he is a senior researcher at Riga Technical University, Department of Systems Theory and Design. His research interests are agent technologies, multi-agent systems, agent oriented software engineering and intelligent tutoring systems.

Janis Bicans is a doctoral student at the Faculty of Computer Science and Information Technology of Riga Technical University, Riga, Latvia since 2010. He started his studies in 2005 and recently received his master degree. The topic of the Master Thesis was "Development of learning objects repository and intelligent tutoring system for intellectual learning support".
He started to work as a researcher in 2010 at Riga Technical University, Department of Systems Theory and Design. His main research interests are data mining technologies, multi-agent systems and intelligent tutoring systems**.**

Egons Lavendelis, Jānis Bicāns. Daudzaģentu un servisorientētas arhitektūras intelektuālu mācību sistēmu izstrādei
Tradicionāli intelektuālās mācību sistēmās (IMS) tiek izmantota modulāra arhitektūra, kas sastāv no četriem moduļiem: pedagoģiskā moduļa, eksperta moduļa, studenta diagnosticēšanas moduļa un komunikāciju moduļa. Tomēr, pieaugot IMS-as sarežģītībai, šāda arhitektūra nenodrošina pietiekamu modularitāti, kā arī nedod iespējas atkārtoti lietot atsevišķas sistēmas komponentes. Lai novērstu šos trūkumus, pēdējās dekādes laikā IMS-u izstrādē tiek izmantotas tādas izkliedētas skaitļošanas tehnoloģijas kā daudzaģentu sistēmas un servisi. Tajā pašā laikā tradicionāliem moduļiem ir savas priekšrocības - tie ļauj izdalīt sistēmas komponentes, kas izmanto pilnībā atšķirīgas zināšanas. Līdz ar to IMS-as dalījumu četros moduļos vēlams saglabāt neatkarīgi no izstrādes tehnoloģijas, tajā skaitā gadījumos, kad tiek izmantotas izkliedētas skaitļošanas pieejas. Pāreja no modulāras arhitektūras uz izkliedētu tiek veikta, realizējot katru moduli kā izkliedētu komponenšu (servisu vai aģentu) kopu. Rakstā apskatīta Rīgas Tehniskajā universitātē izstrādāta atvērta holoniska daudzaģentu arhitektūra un izdarīti secinājumi, kā šajā arhitektūrā iestrādātie principi un gūtā pieredze var tikt izmantoti, veidojot arhitektūru, kas sastāv no cita veida izkliedētām skaitļošanas komponentēm – servisiem. Tāpat analizētas aģentu un servisu kopīgās un atšķirīgās īpašības ar mērķi secināt, kādi mehānismi servisorientētā arhitektūrā ir pārņemami no daudzaģentu arhitektūras un kādi nav. Veikta arī analīze, kādu komponenšu realizācijai servisi ir piemērotāki par aģentiem un kādu komponenšu realizācijai mazāk piemēroti. Rezultātā, ir izstrādāta atvērta servisorientēta IMS-as arhitektūra, kas līdzīgi daudzaģentu arhitektūrai realizē katru moduli kā servisu kopu. Raksts iekļauj visu arhitektūrā esošo servisu aprakstu. Tāpat aprakstītas sarežģītākās darbaplūsmas, ko servisi veic, lai nodrošinātu sistēmas atvērtību. Balstoties uz izstrādāto servisorientēto arhitektūru, ir izstrādāts IMS-as prototips un sekmīgi veikta tā aprobācija studiju kursā „Mākslīgais intelekts". Rakstā iekļauts arī vispārīgs izstrādātās sistēmas apraksts.

Эгонс Лавенделис, Янис Бицанс. Мультиагентные и сервисориентированные архитектуры для интеллектуальных систем обучения
Традиционно интеллектуальные обучающие системы (ИОС) используют модульную архитектуру, которая состоит из четырех основных модулей: педагогического модуля, экспертного модуля, модуля диагностирования студента и модуля коммуникации. Однако при возрастании сложности ИОС такая архитектура не обеспечивает достаточную модульность и не допускает повторное использование отдельных элементов системы. Для преодоления этих недостатков в последнее время при разработке ИОС используют распределенные вычислительные технологии, такие как мультиагентные системы и сервисы. В то же время у традиционных модулей есть свои преимущества - они позволяют выделить компоненты системы, которые полностью используют различные знания. Следовательно, деление ИОС на четыре модуля желательно сохранить вне зависимости от технологий разработки, в том числе при использовании распределенных вычислений. Переход от модульной архитектуры к распределенному вычислению осуществлен, реализуя каждый модуль как набор распределённых элементов (сервисов или агентов). В статье рассматривается открытая мультиагентная архитектура, разработанная в Рижском техническом университете. Можно сделать вывод, что встроенные в архитектуру принципы и извлеченные уроки могут быть использованы в создании архитектуры, которая состоит из других типов распределенных вычислительных компонентов - сервисов. Кроме того, проведён анализ общих и отличительных черт агентов и сервисов, чтобы выяснить, какие механизмы мультиагентной архитектуры может перенять сервисориентированная архитектура, а какие нет. Также проанализировано, для каких компонентов подходит реализация в виде сервисов и для каких не подходит. В результате разработана открытая сервисориентированная архитектура ИОС, которая, как и мультиагентная система, реализует каждый модуль как набор сервисов. Статья включает в себя описание всех сервисов архитектуры. Также описываются важнейшие комплексы задач, какие выполняют сервисы с целью обеспечения открытости системы. На основе сервисориентированной архитектуры разработан прототип ИОС и успешно проведена его апробация в учебном курсе "Искусственный интеллект". В статье также дано общее описание разработанной системы.