ISBN: 978-972-8924-60-7 © 2008 IADIS

OPEN HOLONIC MULTI-AGENT ARCHITECTURE FOR INTELLIGENT TUTORING SYSTEM DEVELOPMENT

Egons Lavendelis, Janis Grundspenkis
Riga Technical University, Department of Systems Theory and Design
Kalku 1, LV-1658, Riga, Latvia

ABSTRACT

During the last decade Intelligent Tutoring Systems (ITS) are becoming more and more popular. One of the research directions of ITS is agent and multi-agent system usage to build ITSs. Despite very intensive research in the agent-based ITS field technologies for ITS development have several drawbacks. Firstly, systems are built without any structure and consisting from large-scale agents, making their development complex. Secondly, reuse of agents is impossible and implementation of changes into the developed systems is complicated due to the need to change existing large scale agents. In this paper we propose open holonic agent architecture. The architecture eliminates the mentioned drawbacks by introducing a holonic structure of relatively small agents. Any agent and holon can be easily added to the system, removed from it and replaced with other agents or holons.

KEYWORDS

Multi-agent architecture, holonic multi-agent system, intelligent tutoring system

1. INTRODUCTION

Initially Intelligent Tutoring Systems (ITS) were developed as monolith systems, because it was the easiest way to get the working system with needed functionality. Later on, to manage complexity more successfully a modular architecture was widely used (Smith, 1998). Unfortunately, this architecture still had large-scale modules, thus having complicated development. Reuse of any parts of the system is impossible in most cases, too.

During the last decade it has been proven that agents can be effectively used to facilitate ITS development. A lot of agent-based ITSs and ITS prototypes have been built (Hospers et al, 2003), (Gascuena et al, 2005), (Postal et al, 2004), (Novac Ududec & Barla, 2007), (Devedzic et al, 2000), (Dorca et al, 2003). However, these systems consist of large scale agents and despite agents having some advantages over traditional modules, they do not solve change implementation complexity and reuse problems in ITSs. For example, if a system has one general problem generation agent, this agent can hardly ever be reused in any other system. Change implementation into the system is complex, because existing agents (their code) has to be changed and every change in any agent can have impact on any other agent, which makes possibility of errors very high. Traditional large-scale agent usage in the ITSs has also such additional drawbacks: (1) agents have multiple tasks, which causes high complexity of their development, (2) a system is incapable to provide sufficient adaptation, if it has only single expert, tutoring strategy, and problem generation agents, (3) single agents can not solve problems which can be easily solved by multi-agent systems using different conflict resolution mechanisms.

To solve mentioned problems, there is a need for a specialized architecture for ITS development. A few architectures have been proposed (Capuano et al, 2000), (de Antonio et al, 2003), (Triantis & Pintelas, 2004). However, they are not open and agents with new functionality can not be easily added to the system. Thus, there is a need for an open architecture, consisting of small-scale agents. Holonic multi-agent systems well satisfy previously mentioned needs. The paper proposes an open holonic multi-agent architecture for ITS development to eliminate abovementioned drawbacks.

The paper is organized as follows. The second section of the paper contains a brief overview of traditional ITS architectures and their drawbacks. The third section consists of an overview of multi-agent architectures

and their comparison in terms of their usage in ITS development. The fourth section is dedicated to the developed open multi-agent architecture for ITS development. Conclusions and outline of future work are given at the end of the paper.

2. OVERVIEW OF INTELLIGENT TUTORING SYSTEM ARCHITECTURES

In the early years of Intelligent Tutoring System (ITS) research systems and their prototypes were developed for teaching specific courses. These systems were developed with monolith architecture, because it was the easiest way to get a working system. Though, such an approach has significant drawbacks. Firstly, development of monolith large-scale systems is very time and resource consuming. Secondly, it is very complicated to implement changes into such a monolith system. Thus, the need for a flexible architecture appeared.

Despite the fact that ITSs have been implemented using different technologies, are built for teaching of different areas and, as consequence, have different characteristics, all ITSs have three main types of knowledge: knowledge about what to teach (domain knowledge), how to teach (pedagogical knowledge) and knowledge about the learner (Grundspenkis & Anohina, 2005). These types of knowledge determine three main modules (a "traditional trinity" (Smith, 1998)) used in a traditional architecture of ITSs (Grundspenkis & Anohina, 2005):

- Expert module most commonly is capable to solve problems in the domain and contains the domain expert's knowledge representation schema.
- Student module collects knowledge about learner's conceptions, errors made, as well as about learner's preferences during the learning.
- Tutoring module holds teaching strategies and instructions to implement teaching process. The primary tasks of this module are controlling of 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, and answering learners' questions. The goal of this module is to reduce the gap between expert's and learner's knowledge levels.

The fourth component of modern ITSs is a user interface or a communication module, which controls interaction with the learner.

Several researches have proved that ITSs have suitable characteristics to be implemented using intelligent agents and multi-agent systems, because ITSs, as a rule, consist of multiple modules, have multiple ways to accomplish different tasks (for example, different teaching strategies), various perspectives (for example, various input devises) and multiple problem solving realities. More details are given in (Grundspenkis & Anohina, 2005), (Webber & Pesty, 2002). 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 (Hospers et al, 2003), MATHTUTOR (Postal et al, 2004), Formal Languages and aUTomata Education system FLUTE (Devedzic et al, 2000), WADIES – a Web- and agent-based adaptive learning environment for teaching compilers (Georgouli et al, 2003), etc.

Anohina and Grundspenkis (2005) offer a set of agents that can be used to implement traditional components of the ITS. *Student model* agents and their tasks are the following. A knowledge evaluation agent builds a model of learner's current level of knowledge based on assessments. A psychological agent builds a profile of learner's psychological characteristics. A cognitive diagnostic 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 curriculum for the specific learner. Each tutoring strategy agent realizes one or multiple teaching strategies. A feedback agent generates feedback and help for the learner. *Expert module* agents or expert agents solve domain specific problems and tasks.

A few multi-agent architectures for ITS development have also been developed. These architectures mainly consist of previously mentioned agents. The greatest part of the architectures is closed – changes in the system are impossible during runtime and implementation of any changes is time and work consuming. Such closed architectures are ABITS (Capuano et al, 2000), IVET (De Antonio et al, 2003), and X-genitor (Triantis & Pintelas, 2004). Several open architectures are proposed: JADE (Silveira & Vicari, 2002) and two

level multi-agent architecture for distance learning environment (Webber & Pesty, 2002). However these architectures are open only for new student agents to join the system and are closed for any other agent types. These proposals give an opportunity to create a learning environment for multiple learners, but they do not help to change existing systems and to develop large-scale agents.

To summarize, all mentioned systems and architectures have some significant drawbacks. The main drawback is complicated change implementation into the system, because existing large-scale agents have to be changed if modification of the system is necessary. Thus, existing code has to be edited and possible impact on other parts of the code must be taken into consideration. In such situations errors are highly possible. Multi-agent systems used to implement ITSs are traditional, one level systems without any organization. Any agent is allowed to communicate with any other, agents are large-scale and they have many tasks. These factors make ITS development complicated. The developed systems do not offer sufficient adaptation as well, because they have only one expert, one tutoring strategy and one problem generation agent (Frasson et al, 1996). Some problems in ITSs need approaches of distributed computing, like various conflict resolution mechanisms.

3. OVERVIEW OF MULTI-AGENT SYSTEM ARCHITECTURES

Development of small multi-agent systems usually includes only definition of agents and interactions among them. In such an approach interaction is possible between every two agents. The complexity of the system increases exponentially when the number of agents increases. It makes development of large-scale multi-agent systems complex and time-consuming. Unstructured multi-agent systems also can not well fit in the organisation, because organisational structure usually is hierarchical. Therefore during the last decade a couple of multi-agent architectures are developed to add organization to the multi-agent systems. Below a brief overview of the main contributions in the field of multi-agent architectures is given.

The most popular multi-agent architecture is *holonic multi-agent system*, where autonomy of agents is reduced and agents are merged into holons, which appear to outside as a single entity (Fischer et al, 2003). The idea of self-similar structures and term “holon” (Greek word “holos” has meaning “whole” and suffix ”-on” denotes “part”) is adopted from biological system research done by A. Koestler (1967). Holon denotes a self-organizing structure which consists of substructures and is a part of larger superstructure. In terms of multi-agent systems holon or holonic agent is an agent that consists of other agents (subholons). Agents join or are joined into holons during the design phase to make them capable to accomplish tasks which they are not capable to deal with alone.

In holonic multi-agent systems agents form a hierarchical structure, i.e., each holon can join a higher level holon and consist of lower level holons. Such hierarchical structure allows adapting the system to the structure of the domain. It is well suitable for task allocation and result sharing in the holons. If the holon has to complete a task, a task can be decomposed into some subtasks that are assigned to subholons, which can decompose them into the next level subtasks. If the agent receives a task that it is not able to accomplish it can also find other agents to create a holon, which is capable to accomplish a task. Also partial hierarchy is possible – some agents may participate in more than one holon, what gives an opportunity to create different structures (Fischer et al, 2003).

Agents that form holons can be either merged into one holonical agent or keep complete autonomy. If agents are merged into one agent, benefits of distribution are lost and complicated mechanisms for merging and splitting agents are needed. If agents keep full autonomy, coordination mechanisms are needed. Thus both extremes have their drawbacks. Usually a model between these extremes is chosen: one agent (called head or head agent) is given privilege to do resource and task allocation. The head of the holon can have partial or total control over other agents. Agents that are parts of the holon, but are not head agents are called body of the holon or body agents (Gerber et al, 1999). Holons have an interface (head agent) and they can be developed separately like modules in traditional software engineering. Holons also make it easier to implement changes in the system, because change of agent in one holon affects only agents from the same holon.

The second well-known multi-agent architecture is *multi-multi-agent system*, which has been developed inside the Agent.Enterprise methodology (Nimis & Stockheim, 2004). This architecture is developed as a result of multi-agent system integration research. The main ideas of holonic multi-agent systems and multi-

multi-agent systems are similar. Both architectures propose to create systems that consist of subsystems. Subsystems are called holons and multi-agent systems, respectively.

Multi-multi-agent systems are created from separate multi-agent systems. Interaction between multi-agent systems is held by gateway agents. Gateway agents accomplish routing and message conversion tasks between different message formats used in different multi-agent systems. Like head agent of the holon gateway agents are interfaces of their multi-agent systems. Although, it is only a mediator between agents of different multi-agent systems and it does not carry out any other tasks, like coordination, task allocation, etc.

Multi-multi-agent systems have weak interaction among different multi-agent systems and intensive interaction inside the multi-agent systems. Multi-agent systems accomplish weakly coupled tasks and interact only to obtain results of other tasks. Multi-multi-agent systems offer to create one higher level (multi-multi-agent system) and one lower level (all multi-agent systems). Holonic multi-agent systems allow creating of unlimited number of levels.

Comparing holonic multi-agent systems and multi-multi-agent systems one may conclude that the basic ideas are similar, although there are the following significant differences in context of ITS building:

- The head of holon unlike the gateway agent can accomplish not only mediator tasks, but also coordination, task allocation and any other tasks needed. It allows creating one main (head) agent of each ITS module and it is a mediator and a coordinator at the same time. In multi-multi-agent systems two agents must be created, thus increasing the complexity of the system and communication in the system.
- Multi-agent systems in multi-multi-agent system are weakly coupled. Contrary, modules in ITS are not weakly coupled, for example, all modules must ask student module for student model data to accomplish their tasks.
- Holonic multi-agent systems allow creating more than two levels of hierarchy, allowing to use hierarchical task decomposition.
- Holons can be dynamically changed providing modification of system's functionality to adapt it to different courses, learning materials and learners.

Thus, holonic multi-agent systems are more appropriate for ITS development than multi-multi-agent systems. Although, it does not mean that holonic multi-agent systems are effective or more effective than simple multi-agent systems for ITS development. Holonic multi-agent systems are more complex technology than traditional multi-agent systems. So, they are effective only for problems, where traditional multi-agent systems have drawbacks. The authors of holonic multi-agent system architecture provide criteria to determine, if a domain is holonic. The most important criteria are the following (Gerber et al, 1999):

- Operator abstraction. Holonic systems are well suited for domains with actions or tasks of different granularity. Higher level tasks are carried out by holon's head and decomposed to subtasks, and assigned to subholons. Tasks in ITS are of different granularity. Firstly, ITS consists of modules. Each module has its own tasks. Secondly, each module has different tasks and these tasks also consist of different types of subtasks. For example, one task of tutoring module is problem generation and one problem generation subtask is test generation.
- Hierarchical structure. Domain with hierarchical structure is well suited for holonic multi-agent systems. In that case subholon hierarchies can be created in accordance with the domain hierarchy. ITS consists of modules, modules consist of agents, which can consist of subagents.
- Decomposability. One of the main prerequisites to use distributed systems like multi-agent systems is that tasks for the system must be decomposable. In some domains tasks are not fully decomposable. In partly decomposable domains tasks can not be fully decomposed and assigned to single agents. Holonic systems are suitable for such domains: decomposable tasks can be decomposed and assigned to the body agents of the holon, not decomposable tasks can be accomplished by the head of the holon. As previously mentioned ITSs are decomposable, although there are tasks that are not decomposable, for example, student module must sustain full student module.
- Social elements. System has to be cooperative to use holonic approach or at least have cooperative elements in the domain. Holons do not enhance non-cooperative interaction and thus it is not reasonable to use holons in non-cooperative domains. ITSs have common goals – to teach the learner appropriately as possible. Thus, ITS has cooperative interactions.

As described previously, ITS domain satisfies all criteria for holonic multi-agent system application. Next section describes the proposed holonic architecture for ITS development.

4. OPEN AGENT-BASED INTELLIGENT TUTORING SYSTEM ARCHITECTURE

In this section the developed open multi-agent architecture is described. The main goal of the architecture is to eliminate mentioned drawbacks of traditional agent-based ITS architectures. The proposed architecture is holonic and hierarchical. The higher level holon consists of agents described in the previous section. Each agent of the higher level is realized as a second level holon. Each agent of the second level in its turn can be realized as a holon. Hence, ITS is developed as a hierarchical holonic multi-agent system with unlimited depth of the hierarchy.

In the developed architecture, ITS appears to the user as a single holon. This holon is called the higher level holon and is represented by an interface agent, which is a head of the holon. The interface agent is the only agent interacting with the learner. Interface agent realizes all functions of communication module. Other modules are realized as subholons and are included in the body of the main holon. Each module can be realised as a single holon or multiple holons, i.e., modules like tutoring and student modules which realize wide functionality are divided into several holons. Student module is realised as student modelling agent and knowledge evaluation agent. Tutoring module is realised as curriculum, teaching strategy, problem generation and feedback generation agents. Expert module is realised as a single expert agent. Thus, the higher level holon consists of 7 agents. These agents have the same functionality as the previously described traditional agents, except three student model agents are merged into one – student modelling agent.

Interaction among higher level subholons is held using single messages. No protocols are specified, because only three basic types of messages are used: (1) service request – a holon requests another holon to accomplish a task; (2) result of the service – answer to the first type of message, containing result of the task; (3) inform message – agent that has acquired new knowledge or made changes in the existing knowledge informs agents that are interested in such knowledge. To characterize interaction among higher level subholons acquaintance diagram (proposed by authors of Gaia methodology (Wooldridge et al, 2000) to specify the lines of communication among different agents) is used (Figure 1). Detailed interaction is beyond the scope of the paper due to space limitations.

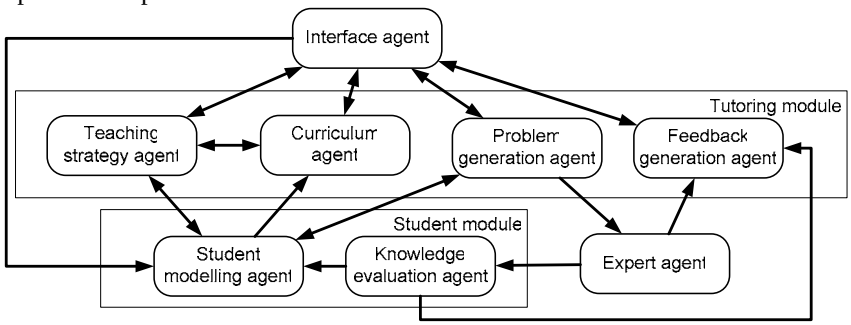

Figure 1. Acquaintance diagram of the higher level holon

Each agent of the higher level is realised as a holon consisting of a single head agent and unlimited number of subholons. 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 holons tasks. Some tasks are physically accomplished by the head, for other tasks the head cooperates with the body of the holon.

The head of the holon coordinates all agents in the holon. Coordination is done by centralised planning and task allocation. To distribute tasks among subholons the head has to discover other agents in the holon and their capabilities. One or multiple global directory facilitator agents are used for that purpose. Subholons have partial autonomy – they are not merged, but the head of the holon has total control over all other agents.

Additionally to the coordination the head of the holon accomplishes tasks that need one commonly known performer, for example, the head of students module maintains full student model and sends it to any other agent that requests it.

Each agent from the body of the holon is capable to perform certain tasks which are subtasks of the tasks assigned to the holon. The capabilities of body agents can be either:

- Principally different. For example, student modelling holon consists of interaction registering agent, cognitive diagnosis agent and psychological agent.
- Types of the same capability. For example, each problem generation agent can generate the most appropriate problem of one category (test, question, different problem solving tasks). This mechanism is the most important to make agents reusable.

Each agent of the body has to register its services with directory facilitator on start-up. Each body agent can be realized as a holon of the next level. Thus, it is possible to build a hierarchy of holons with unlimited number of levels. The proposed architecture is open. Two types of holons are distinguished:

- Closed holons. Closed holons contain only a set of concrete agents. Agents can not be added to the closed holons. Agents in the closed holons have principally different capabilities. Heads of closed holons have complete knowledge about capabilities of all body agents. Thus, centralised planning and task allocation is used in closed holons. Feedback and student modelling agent holons are closed holons.
- Open holons. Open holons consist of a head and body agents of a certain type. Agents in the open holon have capabilities of the same type. The number of concrete agents and their capabilities can change during the runtime of the system. Body agents of the open holons are capable to do certain types of the tasks assigned to the superholon. This mechanism gives an opportunity easily to change functionality of the system by adding and removing agents from the system. For example, task generation holon consists of test generation agent and fill-in-blanks task generation agent. Geometry theorem proving tasks can be introduced just by adding a specific agent. Curriculum, tutoring strategy, problem generation, knowledge evaluation and expert agent holons are open holons.

Heads of open holons have no initial knowledge about subholons of the holon and their capabilities. Thus the following algorithm is used to coordinate agents in the holon. When the head agent receives a request to complete a task, it uses the directory facilitator agent to determine all subholons and their capabilities. Appropriate subholons are chosen and messages with request to perform the task are sent to the subholons. Subholons have to perform their tasks and send results to the head of the holon. If more than one result is provided, the head of the holon chooses the best one as the final outcome (Figure 2). For example, main

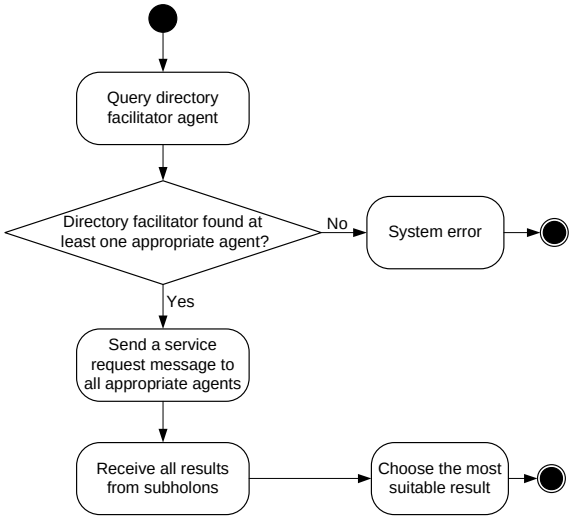

Figure 2. Algorithm used by heads of open lower level holons

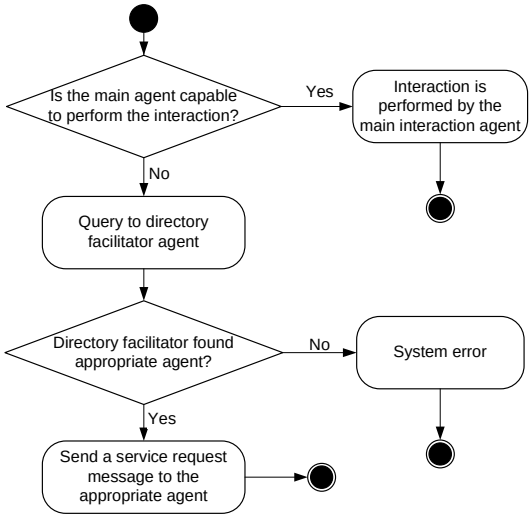

Figure 3. Algorithm used by the main interface agent

problem generation agent receives a request to generate a problem for the specific learner in the specific field. It uses a directory facilitator and finds a test generation agent, a fill-in-blanks problem generation agent and a question generation agent. All three agents are used to get three possible problems which are the best ones of each type. And finally, the most suitable one of all proposed problems is chosen to give to the learner.

Traditional holons consist of a single head agent and multiple body agents. This approach does not allow creating completely open systems because all systems functionality must be delivered to its user. It is done by the interface agent (the head of the holon). If the interface agent is realized as a single agent, it has to be changed every time system's functionality is changed. Thus holonic multi-agent architecture is extended by

realizing the head of the higher level holon (interface agent) as an open holon. Interface agent holon consists of the main interface agent (head of the second level holon) and interface agents. The main interface agent is capable to perform most common interaction with the learner. Other interface agents are capable to perform specific tasks and can be added in case of extending system's functionality.

When an interface agent has to interact with the learner it acts according to the following algorithm. At first it checks its own capabilities – is it capable to perform interaction by itself or not. If yes, then interaction is performed. Otherwise the yellow pages agent is used to find an agent capable to perform needed interaction and request message to the appropriate agent is sent (Figure 3). For example, the main interface agent is capable to show only textual learning materials. In case it needs to show some multimedia learning material, it has to find appropriate agent which is capable to show video to the learner.

Figure 4 shows higher levels of the proposed architecture. The hierarchy of the first level head is detailed along horizontal axis, other agents are detailed along vertical axis. Rounded rectangles with question marks which are added to open holons have meaning that any number of agents can be added to that holon.

Interaction in the lower level holons is realised in the same way as in the higher level – mainly single messages are used. Interaction in closed holons is realised in exactly the same way as in the higher level holon. Interaction in open holons can be organized in the same way, but also any other types of interactions can be used if necessary. For example, agents can use argumented negotiations to find the most appropriate problem for the learner.

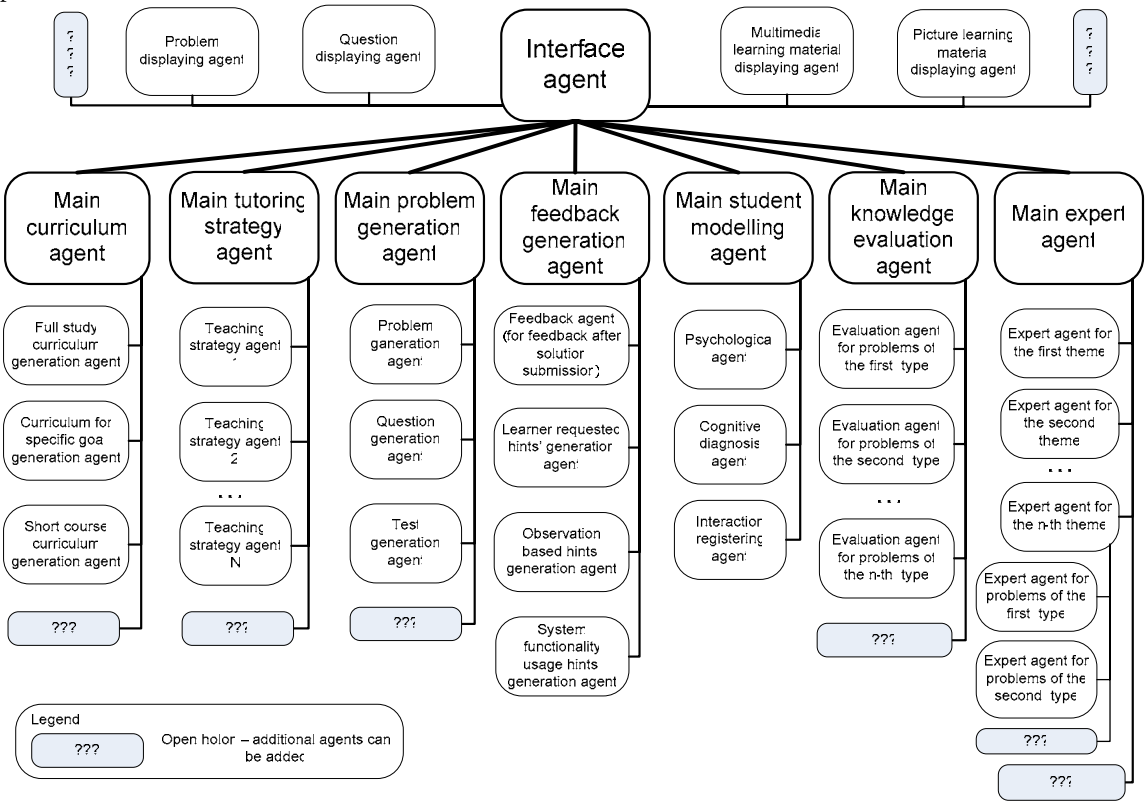

Figure 4. Higher levels of the proposed architecture

Proposed architecture is a high level architecture's framework which can be customized to different needs for the development of various ITSs. It is proposed that all first level agents are realized as holons, although each of these agents can be realized as monolith agent if there is no need to create a holon. All agents can also be realized as holons if it is necessary for any specific functionality of the system. Thus, the developed architectural framework can be simplified and detailed to fit the needs of specific project.

5. CONCLUSION

An open hierarchical holonic multi-agent architecture for ITS development is developed to deal with significant drawbacks in agent-based ITS development. The main advantages of the developed architecture are the following. Firstly, the architecture gives an opportunity easily to implement changes into the system. Specific agents can be added to or removed from the system to change the functionality. Each agent can be replaced with a holon and vice versa to extend system's functionality in some areas and to simplify the system in another areas. Any changes made in the system have less impact on the whole system, because agents interact only within the specific holon not with all agents in the system. Agents that are built for specific small tasks can be easily reused in development of other ITSs. Secondly, development of structured system consisting of small agents is simpler.

The proposed architecture is used in ITS development. The system under development will be used for teaching artificial intelligence course to first year graduate students. Though, the development of the system is in the early phase and evaluation of the effectiveness of the proposed architecture is not possible yet. After finishing the development of the first system it will be possible to evaluate complexity of the change implementation into the system to adapt it to the needs of other courses, too.

Currently we also work on a methodology for holonic multi-agent system based ITS development. The methodology is intended to support full lifecycle of holonic ITS development. It will use a set of diagrams for design phase and transformations from diagrams to code for implementation phase. A tool for ITS development based on the methodology and proposed architecture is also under construction.

ACKNOWLEDGEMENT

This work has been partly supported by the European Social Fund within the National Programme „Support for the carrying out doctoral study programm's and post-doctoral researches" project „Support for the development of doctoral studies at Riga Technical University.

REFERENCES

de Antonio, A. et al, 2003. An Agent-Based Architecture for the Development of Intelligent Virtual Training Environments. *Proceedings of the Second International Conference on Multimedia and Information and Communication Technologies in Education (m-ICTE 2003)*, Badajoz, Spain, pp.1944-1949.

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

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

Dorça, F.A. et al, 2003. A multiagent architecture for distance education systems. *Proceedings of the 3rd IEEE International Conference on Advanced Learning Technologies (ICALT'03)*, Athens, Greece, pp. 368-369.

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

Frasson, C. et al, 1996. An Actor-based Architecture for Intelligent Tutoring Systems. *Proceedings of the Third International Conference, ITS '96*, Montréal, Canada, pp.57-65.

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

Gerber, C.et al, 1999. *Holonic multi-agent systems*, Technical Report R-99-03, DFKI GmbH.

Georgouli, K., Paraskakis, I., Guerreiro, P., 2003. A Web Based Tutoring System for Compilers. *CD Proceedings of the 14th EAEEIE Annual Conference on Innovation in Education for Electrical and Information Engineering (EIE)*, Gdansk, Poland.

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

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

Koestler, A., 1967. *The Ghost in the Machine*. Hutchinson & Co, London, Great Britain.

Nimis, J. and Stockheim, T., 2004. The Agent.Enterprise Multi-Multi-agent System. *Coordination and Agent Technology in Value Networks. Proceedings of the Conference on Agent Technology in Business Applications (ATeBA 2004), part of the Multi-Conference on Business Information Systems (MKWI 2004)*, Essen, Germany, pp. 31-43.

Novac Ududec, C. and Birla, R., 2007. A Multi-Agent Intelligent Platform for Student Tutoring. *Engineering Education in the eWorld*, 2007.

Postal A. et al, 2004. MathTutor: A Multi-agent Intelligent Tutoring Systems. *Proceedings of the First IFIP Conference on Artificial Intelligence Applications and Innovations (AIAI 2004)*, Toulouse, France. - Vol.1. - pp. 231-242.

Silveira, R.A. and Vicari, R.M., 2002. Developing Distributed Intelligent Learning Environment with JADE – Java Agents for Distance Education Framework. *S.A. Cerri, G. Gouardères, and F. Paraguaçu (Eds.): ITS 2002, LNCS 2363*, pp. 105–118.

Smith, A.S.G., 1998. 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 (last visited 18.04.05).

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

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

Wooldridge M. et al., 2000. The Gaia methodology for agent-oriented analysis and design. *Journal of Autonomous Agents and Multi-Agent Systems*, Vol 3., No 3, pp. 285-312.