ISSN 1407-7493 COMPUTER SCIENCE DATORZINĀTNE 2009-7493

APPLIED COMPUTER SYSTEMS
LIETIŠĶĀS DATORSISTĒMAS

REQUIREMENTS ANALYSIS OF MULTI-AGENT BASED INTELLIGENT TUTORING SYSTEMS

DAUDZAĢENTU SISTĒMĀS SAKŅOTU INTELEKTUĀLU MĀCĪBU SISTĒMU PRASĪBU ANALĪZE

Egons Lavendelis, M.sc.ing., researcher, Riga Technical University, Meza 1/4, Riga, LV 1048, Latvia, phone: +371 29876891, egons.lavendelis@rtu.lv
Janis Grundspenkis, Dr.habil.sc.ing., professor, Riga Technical University, Meza 1/4, Riga, LV 1048, Latvia, phone: +371 67089581, janis.grundspenkis@cs.rtu.lv

Multi-Agent Systems, Intelligent Tutoring Systems, Requirements Analysis, Agent Oriented Software Engineering

1. Introduction

During the last decades agents and multi-agent systems are widely used for Intelligent Tutoring System (ITS) development. However, in the most cases no specific methodologies are used to create multi-agent based ITSs. Nevertheless, the methodology is necessary to bring the technology into an industry.
Extensive research in the agent oriented software engineering (AOSE) is ongoing and several agent oriented software engineering methods and methodologies have been proposed. Some of the proposed methodologies include requirements engineering, too, for example, Gaia [1], Prometheus [2], MESSAGE [3], MaSE [4], Tropos [5] and PASSI [6].
At the same time, ITSs have been studied on their own. So, the main characteristics and possible requirements are already known. The set of agents for ITS development is defined [7], as well as a multi-agent architecture for ITS development [8]. This research has to be taken into consideration during the development of a specific ITS.
Multi-agent based ITSs consist of specific agents and modules [9], and are intended to realise specific and complex process of tutoring, which is not taken into consideration in the general-purpose methodologies. Moreover, ITS is a very specific type of agent-oriented software, due to the following characteristics. ITSs are weakly integrated into organisation and have only a few actors (usually no more than three: a learner, a teacher, and an administrator). Thus, requirements can hardly be derived from organisational analysis or roles. Majority of requirements have to come from the tutoring process that a system implements. For details about the impact of the characteristics of ITS on the development process, see [10]. As a consequence, only a small part of requirements analysis techniques used in AOSE are applicable to ITS. There are no general purpose AOSE methodologies that use only appropriate techniques for ITS and allow to use results of the research carried out in the field of ITS. So, the general purpose methodologies are hardly usable in ITS development. At the same time, there are no specific methodologies for multi-agent based ITS development. Thus, there is a need to develop specific approaches for every phase of multi-agent based ITS development and finally integrate them into a multi-agent based ITS development methodology. The developed methodology should include the main ideas from both fields, namely, AOSE and ITS research and take into consideration the main characteristics of ITSs. This paper describes a specific requirements analysis approach for multi-agent based ITS. The approach consists of the most suitable AOSE techniques for ITS requirements analysis.
The remainder of the paper is organised as follows. The second section describes the steps done during the requirements analysis and the proposed approach in general. Section three includes a simple case

study for the proposed approach. Section four describes produced artefacts of analysis phase and their usage in further development phases. Section five includes conclusions and future work.

2. Requirements analysis steps

Various requirements analysis techniques exist. The following ones are used in AOSE: goal modelling (Prometheus [2], MESSAGE [3], MaSE [4], Tropos [5]), use case modelling (MaSE [4], Styx [11], MAS-CommonKADS [12]), task modelling (Prometheus [2], MESSAGE [3], MaSE [4], DESIRE [13], PASSI [6]), organisation analysis (Gaia [1], Tropos [5], MESSAGE [3], MAS-CommonKADS [12], CoMoMAS [14]), agent or role identification (MESSAGE [3], AORML [15]). Agent based ITSs are a specific class of systems with their characteristics. Thus, some requirements analysis techniques are more suitable for ITSs development than others. Organisation analysis and agent or role identification techniques in most cases are not suitable for ITSs at all. Organisation analysis is used to elicit requirements from organisational structure and other organisational characteristics, but ITSs in their turn hardly depend from any organisation. They are used to teach students and are weakly integrated into organisational processes. Mainly only a few actors (a learner, a teacher, and an administrator) are communicating with the system. So the model including up to three actors is hardly informative. We believe that the following three techniques are applicable to ITS development: goal modelling, use case modelling and task modelling, because these techniques are more dedicated to the functionality of the system.

We have chosen to start requirements analysis with the goal modelling step, because goal is one of the easiest forms to formulate the knowledge of domain experts. Teachers usually are used as experts in their courses. Teachers are able to state the goals of the system, because they know what they want to achieve by the system. At the same time they hardly ever know, how agent based ITSs work and how these goals can be achieved. Goals can also be used to help use case creation later during the requirements analysis. DeLoach states that once captured and explicitly stated, goals are less likely to change then activities involved in accomplishing them [4]. Therefore, any other artefact built during the requirements analysis or design phases can be checked against goals – does the system specified in the artefact achieves all goals. These are additional positive characteristics of the goal modelling.

The use case modelling has been chosen as the second technique for requirements analysis. This technique is well elaborated and well known requirements analysis technique analysing the system from different perspectives. Use cases at their own state the high level functional requirements of the system. Use case scenarios specify the way, how these requirements must be fulfilled. Tasks to be accomplished by the system can be derived from scenarios. Interaction among components (agents in case of AOSE) can be derived from scenarios, too. Different use case based techniques can be used during design, implementation and even testing phases.

The third technique that can be successfully used in ITS development is task modelling. Although, having the goal model, creation of the task model is already a design of the tasks that the system has to accomplish to achieve the goals. Therefore, we believe that task modelling has to be included in design phase but not in requirements analysis phase.

So, in our approach during the analysis phase two main successive steps are done, namely goal modelling and use case modelling. The results of the goal modelling are used during the use case modelling. Analysts doing requirements analysis and using the proposed approach need the following knowledge and skills:

- Knowledge about the system under development and its goals. This knowledge is used in the goal diagram and use case models. Thus, domain experts have to participate in the requirements analysis process.
- Use case modelling skills.
- Goal modelling and a goal tree building skills.

2.1. Goal modelling step

During the goal modelling step goal elicitation and decomposition is performed. The step is carried out using the following algorithm (see Figure 1):

1. Define one or more goals to be accomplished using the intelligent tutoring system.
2. Add subgoals that have to be achieved to reach each higher level goal.
3. Continue a goal decomposition until each lower level goal can be achieved by doing direct action or actions.
4. In case the requirements analysis team is bigger than a few people, write goal descriptions, because team members that have not participated in a goal modelling can misunderstand goals if only names of the goals are used. The goal description contains:
    a. Goal description – a brief free text description of the goal.
    b. Condition to be used to determine if the goal is achieved.
    c. An actor that needs to achieve the goal. In many cases the majority of ITS's goals are linked to a learner. In these cases an actor can be specified only for goals needed for other actors, for example, a teacher or an administrator. Actors should be specified also in case if goal descriptions are not written, because they are used during the use case modelling.

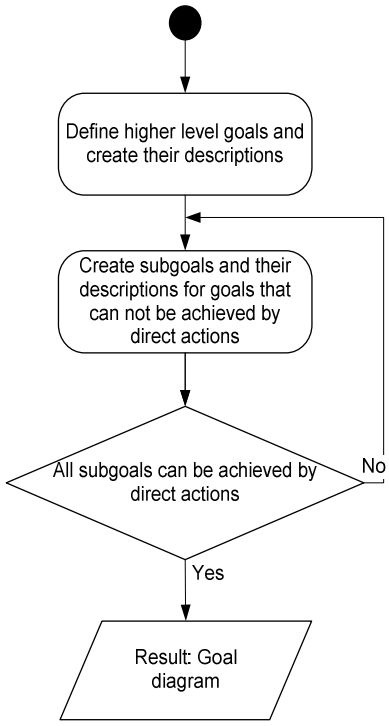


Fig.1. Algorithm used for goal modelling

As a result of goal modelling a goal diagram is created, depicting goals and hierarchical relationships among them. Goals are denoted by rounded rectangles, while decomposition links are denoted with unoriented links. The direction of the hierarchy is determined by the placement of the goals. Decomposition is the only possible semantics of the relationships in the goal diagram. Two types of decomposition are distinguished:

- AND decomposition, meaning that all subgoals have to be achieved to reach the higher level goal. This kind of decomposition is denoted with a simple line.
- OR decomposition, meaning that at least one of the subgoals has to be achieved to reach the higher level goal. This kind of decomposition is denoted by a diamond and connecting lines.

Elements of goal decomposition are denoted in Table 1.

Table 1

Notation of the Goal Diagram

No.	Element of diagram	Notation
1.	Goal	Goal
2.	„AND" decomposition	
3.	„OR" decomposition	

2.2. Use case modelling

The second step done during the analysis is use case modelling. At this step system level use cases are modelled, identifying actors which interact with the system and processes (use cases) happening in the system. This step consists of the following tasks:

1. Identify all actors. ITSs usually have only one primary actor (a learner), which initiates use cases. However, an analyst has to remember that other stakeholders either receiving information from the system or interacting with it in any other way are secondary actors. These secondary actors are added to actors identified during the goal modelling (included in goal descriptions).
2. Create use cases corresponding to actions that have to be done to achieve the lower level goals of the goal hierarchy. Add created use cases to the use case diagram.
3. Create the main success scenario (i.e. scenario in which all actions are completed successfully) for each use case. Creation of the scenario is described in [16].
4. Identify possible exceptions from the success scenario and create use cases and scenarios for identified exceptions. Add created use cases to the use case diagram as extensions of corresponding use cases.
5. Create a description of each use case. A description contains the following mandatory fields: name, goal (achieved by the use case), main success scenario, exception scenarios, preconditions and postconditions. The following optional fields can be added to the description if needed: actors involved in the use case, relationships with other actors and notes. Writing of use case descriptions is described in [17].
6. Use case optimisation using links "<<extend>>" and "<<include>>" among use cases can be done, if necessary.

Algorithm used in use case modelling is shown in Figure 2.

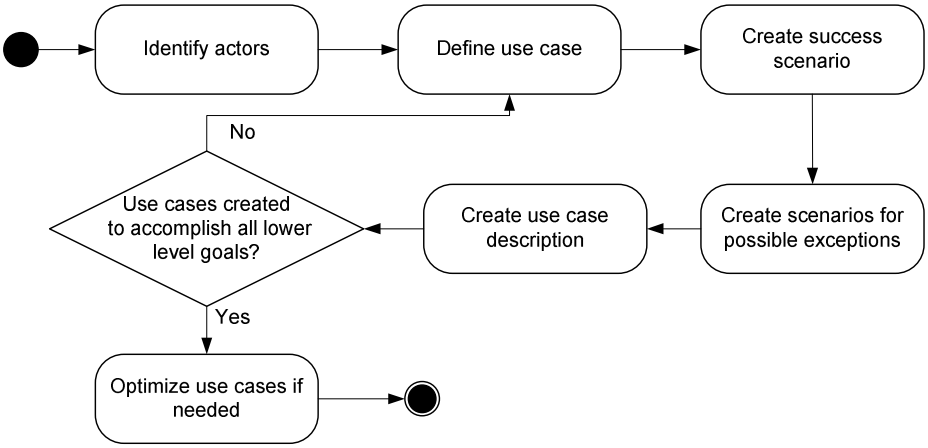


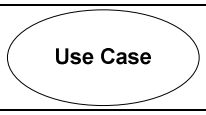
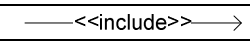
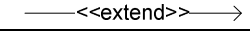
Fig.2. Algorithm used for use case modelling

As a result of use case modelling step the following artefacts are produced:

- UML use case diagram, consisting of actors, use cases, and three types of links among them: an interaction link between an actor and the use case, "<<include>>" and "<<extend>>" links between use cases. Notation used to create use case diagrams is described in Table 2.
- Use case scenarios.
- Use case descriptions.

Table 2

Notation of the Use Case Diagram

No.	Element of diagram	Notation
1.	Actor	 Actor
2.	Use case	 Use Case
3.	Link „interacts"	
4.	Link „include"	 <<include>>
5.	Link „extend"	 <<extend>>

3. Case study

To illustrate described requirements analysis approach a simple case study is developed. Requirements analysis is done for an ITS with the following main requirements. The system has to teach a course to a learner. Course has to be taught by providing learning materials of each topic. After a learner has studied the provided materials the system has to provide a problem or a test for a learner to solve. A problem or test for each learner has to be unique to prevent the learners from copying results. After solving the problem a learner has to receive appropriate feedback. After a learner has passed some tests further learning materials have to be adapted to his knowledge level. Additionally, the system has to inform a teacher about very weak testing results to be able to assist learners if necessary.

3.1. Goal modelling

The first step of the goal modelling is definition of higher level goals. The analysed ITS has one main goal: "Learner taught and tested by the system". This goal is not achievable executing simple actions. So subgoals have to be defined. To achieve the top level goal the following goals have to be reached (goal's name and a brief description provided):

- *Appropriate learning materials provided.* The system has to provide learning materials of the appropriate level according to the learner's knowledge level.
- *Unique problems provided.* The system has to provide unique problems to each student to prevent students from copying solutions.
- *Feedback provided.* After the learner has finished the task, the system has to provide appropriate feedback about his/her mistakes.
- *Teacher informed*. System has to inform a teacher if there is a learner having very poor testing results and needing teacher's guidance.

Table 3

Description of the goal "Feedback provided"

Name:	Feedback provided
Description:	After the learner has finished the task, the system has to provide appropriate feedback about learner's mistakes.
Condition:	All learner's mistakes are clearly explained to the learner.
Actor:	Learner

The first three of the abovementioned second level goals are needed for the learner, but the last one is needed for the teacher. So, the system has two actors. In Table 3 an example of writing goals' descriptions is given.

None of the second level goals defined above is concrete enough to be achieved by a simple action. So, all these goals are decomposed into third level goals. To achieve the goal "Appropriate learning materials provided" the learner's knowledge has to be modelled (goal "Learner's knowledge evaluated using test results") and learning materials have to be adapted to the knowledge level of each learner (goal "Learning materials adapted"). To provide a unique problem to every learner, problems have to be generated from the domain knowledge by the system. So, the goal "Unique problems provided" can be achieved by reaching only one more specific goal – "Problems generated from the domain knowledge". To provide the appropriate feedback to the learner after problem has been solved three goals need to be achieved: generated problems must be solved by the system; learner's mistakes have to be detected by comparing a learner's and a system's solutions; and appropriate feedback as reaction to each mistake has to be chosen. Thus, three corresponding subgoals are defined, namely, "Generated problems solved", "Learner's mistakes detected" and "Appropriate feedback chosen for each mistake". To inform a teacher about learners' success, each learner has to be identified, his results must be registered and a teacher has to be communicated if needed. Three appropriate subgoals are defined, namely, "Learner identified", "Learner's results registered", "Results sent to the teacher". Now, all the lower level goals can be achieved by doing simple actions. Thus, goal modelling step is finished. The goal diagram built during this step is shown in Figure 3.

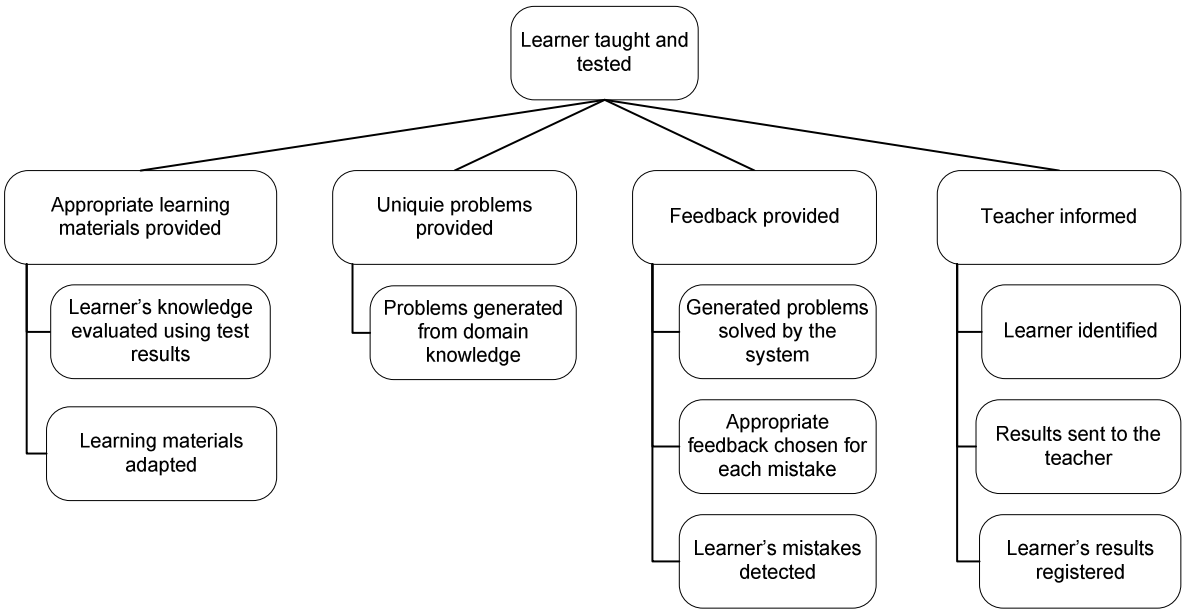


Fig.3. Goal diagram

3.2. Use case modelling

After finishing goal modelling, use case modelling is done step-by-step. The first step is the *actor identification*. Goal descriptions contain two actors: a teacher and a learner. However, these actors are not the only ones, because a system administrator is needed to manage learners' profiles and to give learners access to specific courses. So, system has three actors.

The second step is *use case definition* step. During this step use cases that are needed for the achievement of all lower level goals from the goal diagram are specified one by one. The first lower level goal is "Learner's knowledge evaluated using test results". Two use cases are defined to achieve this goal, namely "Evaluate solution" and "Update student model", because student model can be updated either by the system as a result of learner's solution's evaluation or by a teacher who manually evaluates learner's knowledge level. Use case "Evaluate solution" is shown as an example, including all steps of use case modelling. The success scenario of this use case is the following:

1. A learner submits the solution;
2. The solution is compared with the system's solution;
3. Learner's mistakes and their causes are identified;
4. Feedback is generated and provided to the learner;
5. Student model is updated, including results of new testing;

After defining a success scenario, possible exceptions are defined. The use case under consideration has one main possible exception: if learner's results are too low, then instead of updating a student model, a teacher is informed. The exception scenario includes one step: inform teacher. The use case description of the use case "Evaluate solution" is shown in Table 4.

Table 4

Description of use case "Evaluate solution"

Name	Evaluate solution
Goal:	Learner's knowledge evaluated using test results, learner's mistakes detected
Success scenario	1. Learner submits the solution; 2. The solution is compared to the system's solution; 3. Mistakes and their causes are identified; 4. Feedback is generated and provided to the learner; 5. Student model is updated, including results of new testing.
Exceptions:	During the 5th step it is checked, if the learner's results are too low. If results are too low, the use case "Inform teacher" is done.
Preconditions:	Problem is generated and handed over to the learner. Problem is solved by the system.
Postconditions:	Student model is updated; Feedback is given to the learner.

After completing work on the first use case, use cases needed to achieve other lower level goals are elaborated. Table 5 includes the summary of use cases defined to achieve all lower level goals.

When all use cases are defined, i.e. when all lower level goals can be achieved by the defined use cases, the use cases are optimized. The following optimisation of use cases is performed: the use case "Generate feedback" includes all steps done during the use cases "Generate feedback" and "Update student model". So, two relationships "<<include >>" are created. The use case "Problem generation" includes creating a solution for the generated problem. Thus, one more inclusion is created. The resulting use case diagram is shown in Figure 4.

Table 5

Use cases created to achieve goals

No.	Goal	Use cases
1.	Learner's knowledge evaluated using test results	Evaluate solution, update student model
2.	Learning materials adapted	Generate learning material
3.	Problems generated from domain knowledge	Generate problem
4.	Generated problems solved by the system	Solve problem
5.	Appropriate feedback chosen for each mistake	Generate feedback, evaluate solution
6.	Learner's mistakes detected	Evaluate solution
7.	Learner identified	Register learners to the system, log in to the system
8.	Result sent to the teacher	Inform teacher
9.	Learner's results registered	Update student model

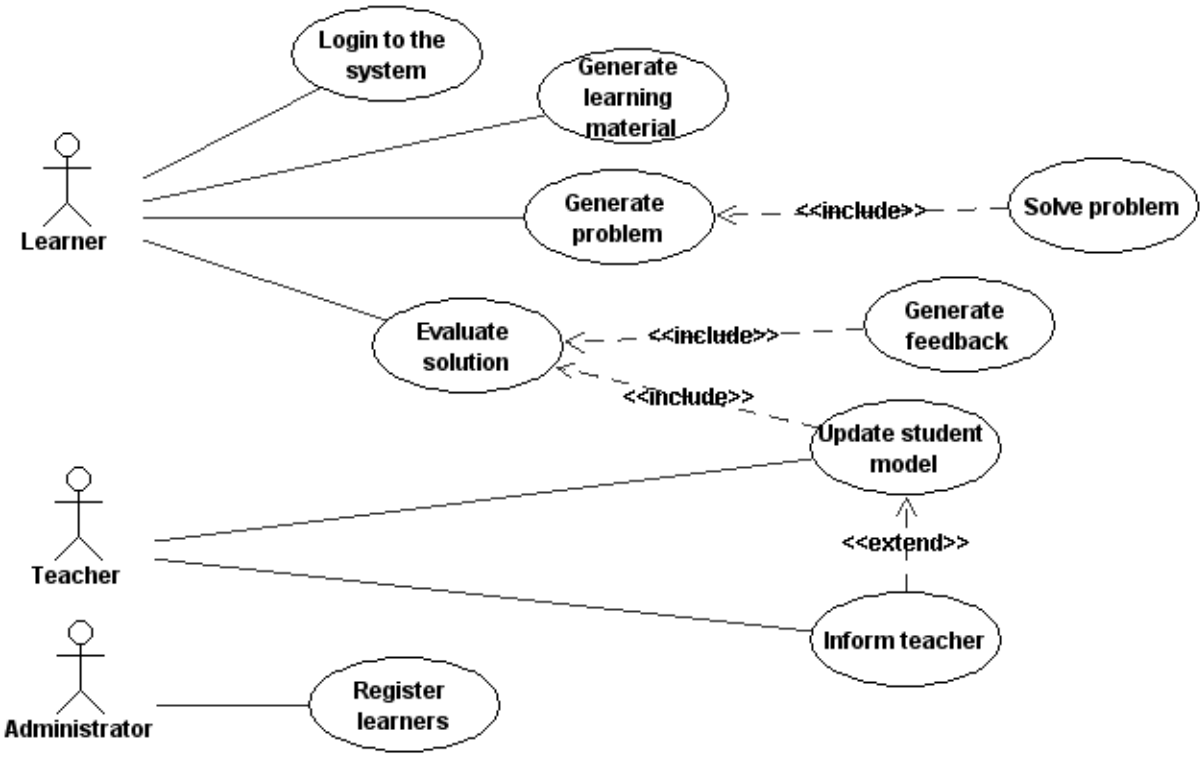


Fig.4. Use case diagram

4. Produced artefacts and their usage in further development phases

During the analysis phase the requirements specification is developed. This document contains two main parts, namely goals that have to be achieved by the system and use cases of the system. The goal model of the system includes the goal diagram (hierarchy) and goal descriptions. The use case model consists of the use case diagram and use case descriptions, including use case scenarios.
Requirements analysis of the ITS is the first, but not the last phase in the ITS development. Thus, one of the main prerequisites for requirements analysis approach is that the artefacts developed during the requirements analysis can be used during the later phases.

Traditionally a goal model is used to create tasks (for example, in MaSE methodology [4]) that have to be accomplished or functionalities (for example, in Prometheus methodology [2]) that are needed in the system to achieve goals. However, goals can be used in an additional way. Each artefact created during the design process can be checked against goals. For example, after system's task definition a set of defined tasks is checked against goals by checking if each lower level goal is achieved by the tasks defined. So, if tasks are defined using use cases not goals, then an important crosscheck can be added to the approach. Such crosschecks are important to eliminate possible design errors and are used, for example, in MaSE methodology [4].
Use cases have even wider usage during the design phase. Steps included in use case scenarios define tasks that have to be accomplished by agents. Although, there is no direct mapping between use case scenario steps and tasks (a step may correspond to more than one task and vice versa), tasks are created to accomplish steps of the use case scenario. After creating tasks based on use case scenarios, a crosscheck against goals is carried out. If an unachieved goal is found then use case scenarios are checked against goals. If a use case, that achieves the goal, exists, then only tasks are refined. If the use case model does not include achieving a goal, then the project is returned to the requirements analysis phase and the use case model is refined. To summarise, we propose to create tasks using use cases and to check created tasks against goals, for details see [18].
After defining a set of agents and assigning tasks to agents, the interaction among agents is designed. Interaction design from scratch can be very complicated process. Errors in such design are highly possible. Use case scenarios can be used to assist the designer during this process. In fact use case scenarios partly include interaction among agents in the following way. If two tasks corresponding to consecutive steps of use case scenario are assigned to different agents, then communication among these agents is mandatory. Use case scenarios are not directly usable in interaction design, because the interaction is not explicitly denoted. To explicitly denote the information about the interaction among agents included in use case scenarios, use case maps are used. Traditionally, use case maps are used to model the control passing path during the execution of use case, for details see [19]. In multi agent design use case maps include agents, their tasks and message path among them. The message path corresponds to the use case scenario in the following way. Each task corresponds to a step (or steps) in the use case scenario [11]. From one side, each message link is a link among two consecutive steps of the scenario. From the other side, each link between two tasks can be considered as a message between corresponding agents. Thus, after creating the use case map an interaction among agents can be easily specified just by adding a message content [18].

5. Conclusions and future work

A specific ITS requirements analysis approach has been proposed. The suitability to the main characteristics of the ITSs of the most popular requirements analysis techniques has been studied. Two suitable requirements analysis techniques has been chosen to include in the approach.
The proposed requirements analysis approach is also well connected to the further ITS development phases, in the way that artefacts developed during the requirements analysis are suitable for usage in the later stages of ITS development, especially in design. The proposed approach enables creating the ITS design approach, which uses these artefacts. Thus, the requirements analysis artefacts have been chosen to be suitable for usage in the design phase. Goal hierarchy can be used for checking if the designed system achieves all goals. Use cases are suitable for task definition and interaction design.
The selected requirements analysis techniques worked well for the simple case study included in the paper. More detailed case study of our approach is under development. After finishing the case study it will be possible to evaluate the proposed approach. Although, we think that the abovementioned advantages of the approach make it to be a promising one.
We are currently developing approaches for all phases of the agent oriented software development life-cycle and plan to integrate them, creating a full life-cycle methodology for multi-agent based ITS development. A graphical tool that supports a full life cycle and uses the proposed diagrams during the requirements analysis is under development, too.

References

1. Wooldridge M., Jennings N.R., Kinny D. The Gaia methodology for agent-oriented analysis and design // Journal of Autonomous Agents and Multi-Agent Systems, 2000. – pp. 285-313.
2. Padgham L., Winikoff M. Prometheus: A Methodology for Developing Intelligent Agents // Agent-Oriented Software Engineering III, LNCS – Vol. 2585, Springer, 2003. – pp. 174-185.
3. Caire G., Garigo F. Gomez J., Pavon J., Leal F., Chainho P., Kearney P., Stark J., Evans R., Massonet P. Agent Oriented Analysis Using MESSAGE/UML // AOSE 2001. – pp. 119-135.
4. DeLoach S. Analysis and Design Using MaSE and agentTool // Proceedings of the 12th Midwest Artificial Intelligence and Cognitive Science Conference – Oxford OH, March 31 - April 1 2001. – pp. 1-7.
5. Giunchiglia F., Mylopoulos J., Perini A. The Tropos Software Development Methodology: Processes, Models and Diagrams. Technical Report No. 0111-20, ITC - IRST, Nov 2001.
6. Burrafato P., Cossentino M. Designing a multi-agent solution for a bookstore with the PASSI methodology // Electronic Proceedings of Fourth International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS-2002) at CAiSE'02. – Toronto, Canada, 2002.
7. Grundspenkis J., Anohina A. Agents in Intelligent Tutoring Systems: State of the Art // Scientific Proceedings of Riga Technical University, 5th series „Computer Science. Applied Computer Systems" – Vol.22 (2005), pp. 110-121.
8. Lavendelis E., Grundspenkis J. Open Holonic Multi-Agent Architecture for Intelligent Tutoring System Development. // Proceedings of IADIS International Conference „Intelligent Systems and Agents 2008". – Amsterdam, The Netherlands, 22 - 24 July 2008. – pp. 100-108.
9. Smith A.S.G. 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 18.04.2005)
10. Lavendelis E., Grundspenkis J. MASITS - A Tool for Multi-Agent Based Intelligent Tutoring System Development // Proceedings of 7th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS 2009) – Salamanca, Spain, 25-27 March 2009. Advances in Intelligent and Soft Computing Vol. 55., Springer, 2009. – pp. 490-500.
11. Bush G., Cranefield S., Purvis M. The Styx Agent Methodology, 2002. http://citeseer.ist.psu.edu/cache/papers/cs/22112/http:zSzzSzdivcom.otago.ac.nzzSzinfoscizSzpublctnszSzcompletezSzpaperszSzdp2001-02.pdf/bush01styx.pdf (Last visited: 10.10.2007)
12. Iglesias C., Garijo M., Gonzales J.C., and Velasco J.R. Analysis and design of multiagent systems using MAS-CommonKADS // Intelligent Agents IV (ATAL97), LNAI 1365, Springer-Verlag, 1998. –pp. 313-326.
13. Brazier F. M. T., Dunin-Keplicz B. M., Jennings N. R. and Treur J. DESIRE: Modelling Multi-Agent Systems in a Compositional Formal Framework // International Journal of Cooperative Information Systems, 6(1), pp. 67-94, 1997.
14. Glaser N. The CoMoMAS Approach: From Conceptual Models to Executable Code. http://www.loria.fr/~glaser/extern/PUBLICATIONS/maamaw97-1.ps.gz (Last visited: 25.09.2007).
15. Wagner G. A UML profile for external AOR models // Electronic Proceedings of the Third International Workshop on Agent-Oriented Software Engineering, 2002.
16. Getting Started With Use Case Modeling. An Oracle White Paper, 2005.
17. Rolland C., Achour C.B. Guiding the Construction of Textual Use Case Specifications // Data & Knowledge Engineering Journal. – Vol 25, No.1-2, 1998. – pp. 125-160.
18. Lavendelis E., Grundspenkis J. Design of Multi-Agent Based Intelligent Tutoring Systems // Scientific Proceedings of Riga Technical University „Computer Science. Applied Computer Systems", RTU Publishing, Riga, 2009 (accepted for publishing).
19. Buhr R.J.A., Elammari M., Gray T., Mankovski S. Applying Use Case Maps to Multi-Agent Systems: A Feature Interaction Example // HICSS(6), 1998, – pp. 171-179.

Lavendelis E., Grundspeņķis J. Daudzaģentu sistēmās sakņotu intelektuālu mācību sistēmu prasību analīze
Pētījumi aģentorientētā programmatūras inženierijā piedāvā vispārīgus aģentorientētas programmatūras izstrādes principus. Savukārt, intelektuālu mācību sistēmu (IMS) pētījumi piedāvā specifisku arhitektūru un citas specifiskas zināšanas IMS izstrādei. Abi šo pētījumu rezultāti ir jāņem vērā, veicot IMS izstrādi. Tādēļ ir nepieciešamas specifiskas pieejas visām izstrādes fāzēm tieši aģentos sakņotu IMS izstrādei. Šīm pieejām jāiekļauj abu minēto virzienu pētījumi un jāņem vērā IMS īpatnības. Šajā rakstā ir piedāvāta aģentos sakņotu IMS prasību analīzes pieeja. Rakstā iekļauts arī vienkāršas aģentos sakņotas IMS prasību analīzes praktisks piemērs. Piedāvātā prasību analīzes pieeja sastāv no diviem galvenajiem soļiem: mērķu modelēšanas un lietošanas gadījumu modelēšanas. Mērķu modelēšanas gaitā tiek identificēti sistēmas mērķi un izveidota izstrādājamās sistēmas mērķu hierarhija. Lietošanas gadījumu modelēšanas gaitā tiek izstrādāti lietošanas gadījumi, ar kuru palīdzību ir iespējams sasniegt izvirzītos sistēmas mērķus. Aprakstīto prasību analīzes pieeju ir plānots iekļaut pilna dzīves cikla aģentos sakņotu IMS izstrādes metodoloģijā. Analīzes laikā izveidotie artefakti tālākās fāzēs ir izmantojami šādā veidā. Lietošanas gadījumu modelis (īpaši lietošanas gadījumu scenāriji) tiek izmantoti veicamo uzdevumu definēšanai un aģentu mijiedarbības projektēšanai. Projektēšanas laikā mērķu diagrammu var izmantot, lai pārbaudītu, vai katrs izstrādātais artefakts paredz visu mērķu sasniegšanu.

Lavendelis E., Grundspenkis J. Requirements analysis of Multi-Agent Based Intelligent Tutoring Systems
The agent oriented software engineering research proposes general assumptions for agent oriented software development, while intelligent tutoring system (ITS) research proposes specific ITS architecture and other specific knowledge for ITS development. Both of these views should be taken into consideration while developing multi-agent based ITSs. Thus there is a need for specific approaches for all phases of agent based ITS development which take into consideration main ideas from both agent oriented software engineering and ITS research. In this paper we propose a requirements analysis approach for multi-agent based ITSs. A case study of a simple ITS is included, too. Requirements analysis in the proposed approach consist of two main steps, namely goal modelling and use case modelling. During the goal modelling the main goals of the system are identified and a goal hierarchy for the system is created. During the use case modelling use cases needed to achieve each lower level goal and their descriptions are created. The proposed approach of the requirements analysis is intended to be a part of the full life cycle methodology for multi-agent based ITS development. The developed use case model (especially use case scenarios) is used during the agent interaction design and task definition. Goal hierarchy during the design phase is mainly used for checking, if the results of design achieve all system's goals.

Лавенделис Е., Грундспенкис Я. Аналис интелектуалних систем обучения основанних на многоагентних системах
Исследования в области проектирования информационных систем основанных на интелектуальных агентах предлагают основные принципы разработки систем, основанных на агентах. В свою очередь исследования в облости разработки обучающих информационных систем предлагают методы применимые для специальных обучающих систем. Создавая обучающую систему основанную на интелектуальных агентах, необходимо брать во внимание оба выше упомянутых подхода к проектированию. В данной статье представлен подход анализа требований к многоагентной обучающей системы. Также включен небольшой пример практического применения данного методя
Предложенный способ описания требований для системы состоит из двух этапов: моделирование целей системы и моделирование вариантов использования системы. На этапе моделирования целей системы выявляются основные назначения системы и составляется иерархия целей. На этапе моделирования вариантов использования определяются и описываются варианты использования необходимые для достижения целей. Расмотренный в статье подход к созданию требований к системе планируется включить в методологию разработки обучающих информационных систем основанных на агентах, которая обобщит весь жизненный цикл проекта. Разработанная модель вариантов использования системы используется при проектировании взаимодействия агентов и определении заданий. Иерархия целей на этапе проектирования главным образом используется для проверки того, достигнуты ли все цели системы.