Extending the MASITS Methodology for General Purpose Agent Oriented Software Engineering

Egons Lavendelis
Department of Artificial Intelligence and Systems Engineering, Riga Technical University, Riga, Latvia

Keywords: MASITS Methodology, Agent Oriented Software Engineering, Multi-agents Systems.

Abstract: The aim of the paper is to extend the agent oriented software engineering methodology MASITS that was initially developed for agent based Intelligent Tutoring System (ITS) development to make it usable for other agent oriented system development. The paper analyses the steps of the methodology, finds the specific ones that are either adapted to ITS characteristics or use particular artefacts from ITS research. Three extensions of the methodology have been developed, namely, a general holonic architecture, agent definition method and interaction design method. As a result, the extended version of the methodology can be used for agent oriented system development in case the system has similar characteristics to agent based ITSs. Case study of the insurance policy market automation software is used to validate the use of the extended version in the development of other kind of systems than ITSs.

1 INTRODUCTION

Multi-agent paradigm has been developed a few decades ago and is theoretically applicable to solve a wide range of complex problems. Despite of intensive research there are very few representative commercial applications. One of the reasons for that is the lack of sufficient methodological support for industrial development process. To overcome this obstacle, during the last two decades large number of Agent Oriented Software Engineering (AOSE) methodologies has been proposed to support the development of multi-agent systems. The existing methodologies can be divided into two main groups, namely general purpose and domain specific agent oriented software engineering methodologies. The first group provides general enough techniques to be usable for any type of the agent oriented software. This causes also the main disadvantage – the process is not adapted to the particular type of the systems and often includes unnecessary tasks. Examples of this group are Gaia (Zambonelli et al, 2005), Prometheus (Padgham and Winikoff, 2004) and O-MaSE (DeLoah, 2014).

The second group provides tools and techniques that are more appropriate to the particular type of systems. Usually the specific methodologies have been developed for very narrow class of systems, for example, Intelligent Tutoring Systems (ITSs) (Lavendelis and Grundspenkis 2009a), electricity distribution systems (Varga et al, 1994) or organization integration (Kendall et al 1995). These methodologies usually adapt the AOSE process to the characteristics of particular systems to simplify the software engineering process, by including knowledge from the particular domain and using more specific techniques compared to general purpose methodologies. Thereby it is obvious that these methods are not applicable outside the specific domain. Still some of the development mechanisms used in specific purpose agent oriented software engineering methodologies are suitable for wider classes of systems than initially designed. Thus it might be useful to reuse these specific purpose methodologies to other domains with similar characteristics. The paper analyses the MASITS methodology that was originally developed as a specific purpose agent oriented software engineering methodology for ITS development (Lavendelis and Grundspenkis, 2009a) and extends it for the wider range of agent oriented systems.

The remainder of the paper is organied as follows. The Section 2 briefly outlines the original version of the MASITS methodology. The Section 3 analyzes the original methodology and concludes the needed extensions for the methodology to be usable in the development of wider range of the systems.

The Section 4 describes the extended version of the methodology. The Section 5 outlines the case study of the methodology's extended version. The Section 6 concludes the paper.

2 THE MASITS METHODOLOGY

MASITS (MAS for ITS) is a specific purpose AOSE methodology for ITS development (Lavendelis and Grundspenkis, 2009a). The MASITS methodology comprises the most important results of ITS development research and AOSE methodologies. The development techniques are adapted to the main characteristics of the ITSs.

The process of software engineering according to the MASITS methodology consists of the following phases: analysis, design (divided into two stages: external and internal design of agents), implementation, testing, deployment and maintenance. These phases are presented sequentially, although the development process is iterative. Iterations are used both inside the phases and across different phases. Developer of the system is allowed to return to any previous phase and refine the previously created models (Lavendelis & Grundspenkis, 2009a). Phases of the development process and steps included in these phases are shown in Figure 1.

One of the successful developments is the MIPITS system that teaches course "Fundamentals of Artificial Intelligence" to undergraduate students at Riga Technical University. The agent based approach is used to provide different types of tasks and adapt tasks to the needs of individual students (Lavendelis and Grundspenkis, 2010). The MIPITS system also proved the usefulness of the holonic architecture and openness of the system. As described in (Lavendelis and Grundspenkis, 2011), the system can be easily extended with a new task and its assessment mechanisms without changing the existing code of the system. Based on this experience it was concluded that MASITS methodology can be successfully used for the ITS development and could possibly be extended to be usable for wider range of systems. The next section analyses the ITS specific steps in the development process and identifies the needed extensions for the methodology to be usable in the development of the wider range of systems.

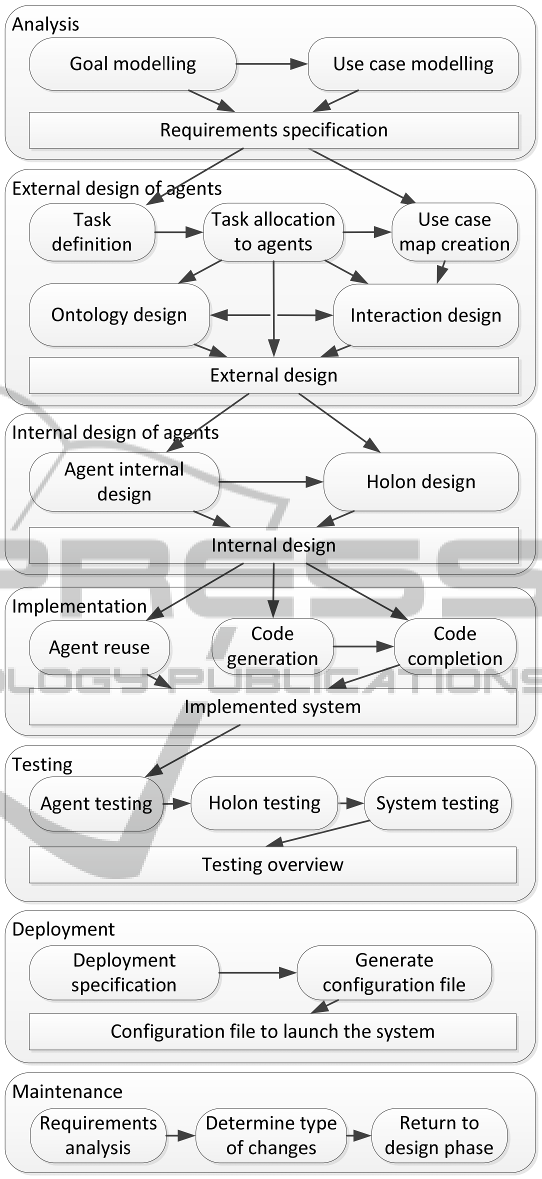


Figure 1: Structure of the MASITS methodology.

3 METHODOLOGY ANALYSIS

The MASITS methodology adapts the software engineering process for the agent based ITS development. It includes knowledge from the ITS research and contains specific techniques that are suitable for the characteristics of ITSs. The methodology simplifies several steps of the development by eliminating redundant and repeating tasks that are the same for all the ITSs, because the results of these steps are already included in the methodology, for example, the methodology contains a typical set of agents that is used to implement any ITS and the designer can modify (if needed) and reuse it (Grundspenkis and Anohina, 2005).

It can be concluded that the adapted development process not only simplifies ITS development, but also is the main obstacle for the use of the methodology in other agent based system development if these systems have other characteristics than ITSs or do not have the corresponding artefacts that are used from ITS research in the original version of the methodology. For example, the original version of the methodology will not be efficient for systems where the set of agents must be defined or complex communications among agents must be designed. The remainder of this section lists the steps of the methodology and outlines the specific techniques and artefacts that must be changed to enable usage of the methodology for wider class of agent based systems.

The **analysis phase** consists of two consecutive steps:

- *Goal modelling* resulting in goal diagram that depicts goal hierarchy of the system and goal descriptions. This step is specific for goal based systems, but is not ITS specific.
- *Use case modelling* resulting in a use case model containing diagrams and use case descriptions. This is a well-known technique in software engineering and is not ITS specific.

The first stage of the design phase is **external design of agents** when agents are designed in terms of their functionalities and interactions among them. This stage consists of the following steps:

- *Task definition*. This step contains task definition according to the steps of use case scenarios and crosscheck of defined tasks against goals. The step is not ITS specific.
- *Task allocation to agents*. During this step tasks are organized into hierarchies and assigned to agents. This step uses the following ITS based artefacts (1) higher level agent set (Grundspenkis and Anohina, 2005), (2) holonic agent architecture (Lavendelis and Grundspenkis, 2008), (3) informal rules for task allocation to agents (Lavendelis and Grundspenkis, 2009a). These artefacts allow to modify the existing ITS research based set of agents instead of defining the set of agents for every ITS from scratch. The need for these artefacts limits the use of the methodology for any other type of systems.
- *Interaction design*. This step results into interaction diagram that depicts agents, users and links among them. Links among agents are designed only in terms of messages sent. In more complex cases this step may include a use case map creation as a sub-step. This step is adapted to the fact that agents in ITSs send relatively low number of messages, there are no complex interactions and as a consequence the order of messages is self-explanatory. This step is not suitable to systems with intensive communications among agents or any complex interaction mechanisms that must be designed separately.
- *Initial ontology design*. The domain ontology is created in parallel to the interaction design to define the contents of the messages sent in terms of the predicates. The step results in the initial ontology diagram. This step is not ITS specific.

During the second stage of the design, namely, the **internal design of agents**, the internal structure of agents is defined in the internal views of agents. The MASITS methodology designs agents in terms of low level concepts that correspond to the selected implementation platform JADE (Java Agent DEvelopment Framework, http://jade.tilab.com/). It ensures easy transition to implementation phase and easy code generation. Therefore, agents are designed in terms of messages sent and received, events perceived and actions done by the agents. Additionally the MASITS methodology supports design of holonic MASs. The holonic architecture used for this purpose is ITS specific. As a consequence the whole stage is implementation platform specific, and in case of holonic MASs also ITS specific.

The **implementation phase** of the MASITS methodology consists of the following three steps:

- *Agent reuse*. During this step lower level agents from previous projects can be reused if they have the needed functionality. An ITS specific library of agents is used for this purpose.
- *Code generation*. In this step the MASITS tool is used to generate the source code from the detailed design. This step is not ITS specific.
- *Code completion*. This step is programming. The generated code is completed by adding detailed code of all methods.

During the **testing phase** JADE test suite (Cortese et al., 2005) is used to develop and execute tests. Thus the phase is implementation platform specific, but is not ITS specific.

Deployment is done by using modified version of the UML deployment diagram that shows JADE containers and agent instances deployed in each container. The diagram is used by MASITS tool to generate a configuration file. This step is implementation platform specific, but not ITS specific.

During the **maintenance phase** the MASITS methodology supports change implementation into the system. The holonic architecture simplifies the implementation of changes that are related to the functionality of only one or small number of holons. Thus the phase is specific to the holonic multi-agent systems.

The Table 1 summarizes the specific steps that restrict the usage of the MASITS methodology for development of other types of systems.

Table 1: ITS specific approaches in the steps of the MASITS methodology.

Step	**Specific approaches**	**Results of ITS research used**
	External design of agents	
Task allocation to agents	Tasks are assigned to already known set of higher level agents	Set of higher level agents and holonic architecture
Interaction design	Interactions among agents are designed only in the form of messages sent	-
	Internal design of agents	
Agent internal design	Specific concepts to JADE	-
Holon design	Specific to holonic multi-agent systems	Holonic ITS architecture
	Implementation	
Agent reuse	Reuse agents from previously created ITSs	Library of previously created ITSs
Code generation	Agents are implemented in JADE platform	-
Code completion	Agents are implemented in JADE platform	-
	Testing	
The whole phase	Uses JADE Test Suite and thus is platform specific	Holonic ITS architecture
	Deployment	
Generate config. file	JADE specific configuration	-
	Maintenance	
The whole phase	Holonic architecture is used in change implementation	Holonic ITS architecture

4 EXTENSIONS OF THE METHODOLOGY

Based on the analysis done in the previous section it has been concluded that the main obstacles to use the MASITS methodology in the development of other types of systems than ITSs are the following:

- Multiple steps of the methodology use a set of higher level agents and the holonic ITS architecture. It is solved by providing a general holonic architecture described in the Section 4.1.
- The methodology uses predefined set of agents and customizes it as needed in the ITS development, that is not possible in case of other systems. The Section 4.2 outlines agent definition method that substitutes the task allocation method.
- The interactions are designed only in terms of messages sent among agents that is not enough in case of complex interactions. To resolve it, new interaction design method is proposed in Section 4.3.
- The internal design of agents, implementation, testing and deployment are done in the platform specific way that limits the use of the methodology to the systems implemented in JADE. Still this is not changed, because more general methods make the transition from the design to the implementation more complex.

4.1 General Holonic Architecture

One of the advantages of the MASITS methodology is the possibility to design systems of any complexity due to the usage of holonic multi-agent paradigm initially proposed by Fischer et al (2003). The methodology designs the system by decomposing it into holons. Holons are agents that consist of subholons or subagents. They consist of a single head and some body agents. The head represents the holon outside it and coordinates body agents. The design of holons is done in the top down approach. Each holon initially is designed as a single agent and afterwards its internal structure is designed. This approach requires holonic architecture. In case of ITSs a domain specific architecture (Lavendelis and Grundspenkis, 2008) is used. To make the methodology usable for other types of systems and keep the advantage of the holonic system design general holonic architecture has been included in the methodology and used instead of the specific one.

The developed general architecture (see Figure 2) keeps the main characteristics from the specific one. It is open, hierarchical and holonic. The system 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 the head of the holon. The interface agent is the only agent interacting with the user. Other functions of the system are realized in higher level agents that are included in the body of the main holon. The higher level holon may contain any number of higher level agents. These agents are defined during the design time (agent definition method is given in the next subsection). Thus the higher level holon is closed in the sense that no new agents can be added to the system after the development has ended. Each of the higher level agents can be implemented in one of the following ways:

- As a single agent. This option is chosen if the corresponding functionality is simple and can be implemented in a single component. It is not recommended if the functionality will be changed frequently.
- As a closed holon. In this case the higher level agent is implemented as a multi agent system. Term closed means that the lower level agents are defined in the design time and cannot be changed during the runtime.
- As an open holon. Similarly to the previous option the agent is implemented as a multi-agent system, but at the design time only the types of body agents are defined. Actual number and instances of body agents may vary in the runtime of the system.
- As a multi-level holon hierarchy. Each lower level agent in any holon can be implemented as a multi-agent system itself, so creating multi-level hierarchy. The depth of the hierarchy is unlimited (Figure 2 shows only the first two levels for readability reasons).

Figure 2 gives examples of all possible types of higher level agents.

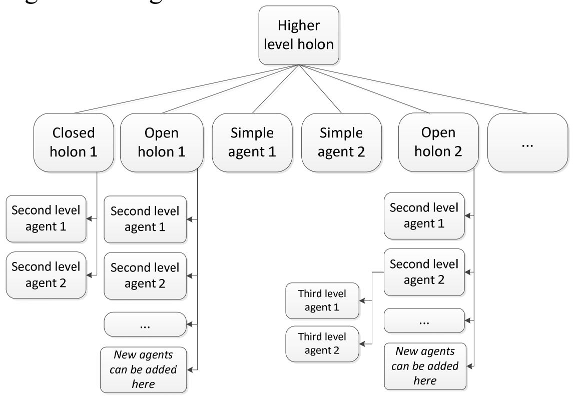


Figure 2: General holonic architecture.

4.2 Agent Definition Method

The original version of MASITS methodology uses predefined set of higher level agents coming from the ITS research (Grundspenkis and Anohina, 2005) and instead of agent definition step includes task assignment step to the higher level agents. To enable development of other kinds of agent oriented software a method for agent definition is proposed.

The agent definition is started by grouping tasks that will be executed by the same agent. An agent will be created to perform each group of tasks. Groups and as a consequence agents can be created for one of the following entities:

- Users or user roles. This approach is used if there are many users or user roles with different functionality needs. This approach is especially efficient if it is necessary to autonomously represent each user and negotiate with others agents on behalf of the corresponding user.
- Organizations or stakeholders. This option is chosen if there is a need to autonomously represent each stakeholder in the system. An example of such systems is electronic marketplaces.
- Legacy systems or any standalone components that must be integrated into the system. This approach is used if the role of multi-agent system is to integrate several systems.
- Use cases. This approach is used if the system has several relatively unrelated use cases and each of them can be processed by a corresponding agent.
- Knowledge types. This option is chosen if agents have to process different types of knowledge.

Based on the above principles the following iterative method has been included in the methodology:

1. Choose one or several of the abovementioned approaches for agent definition.
2. Group tasks and create task hierarchies from the previously defined tasks according to the chosen grouping strategies.
3. Define agents for task hierarchies. An agent can be defined for one or several task hierarchies.
4. Principles for hierarchy grouping have been taken from Prometheus methodology (Padgham and Winikoff, 2004):
    a. If two tasks are closely related then they should be allocated to the same agent. If two tasks are not related they should not be assigned to the same agent.
    b. If two tasks include processing of the same knowledge then they should be assigned to the same agent.
5. If more than one approach is chosen in the step 1 then steps 2-4 are repeated to define sets of agents for all these approaches.
6. The sets are evaluated according to the principles defined in the Step 4 and the best one is chosen.

The output of the method is a set of higher level agents and task-agent diagram that is identical to the one in the original version of the MASITS methodology and can be used in the following steps without any changes.

4.3 Interaction Design Method

In the original version of the MASITS methodology interactions are modelled only in terms of messages sent among agents and not in terms of interaction protocols. Such an approach is sufficient due to the specifics of ITSs, but is not enough in other systems. Thus the interaction design step has been changed to the following form to include also protocol design if it is needed:

1. Interaction design and creation of the MASITS interaction diagram. This diagram is sufficient for the simplest interactions.
2. If the interactions are too complex or unknown for the designer use case maps are created in the same way as in the original version of the methodology.
3. Interactions containing too many interactions and making the diagram unreadable are depicted in separate interaction diagrams.
4. Interactions where it is important not only to specify the messages sent, but also the order and context of the messages are depicted in the protocol diagrams.
5. Creation of the initial ontology. It is done during the interaction design to enable definition of the message contents in terms of predicates sent and concepts that are parts of the predicates.

Original MASITS notation is used for steps 1-3 and 5 while Agent UML protocol diagram notation (Huget and Odell, 2005) is used for protocol design.

4.4 The Extended Version of the MASITS Methodology

The structure of the extended version of the MASITS methodology is given in Figure 3. The changed steps and phases are highlighted with grey colour. The last 3 phases, namely, testing, deployment and maintenance are omitted, because they contain no changes compared to Figure 1.

The extended version of the MASITS methodology is usable for development of systems that have the following main characteristics similar to ITSs:

- Software system where agents implement system's modules.
- Software that implements intelligent mechanisms like adaptation mechanisms in ITSs.
- Goal directed system, because MASITS development process starts with goal definition and the development is goal oriented.
- Highly modular system. System consists of large number of agents that implement system's functionality.

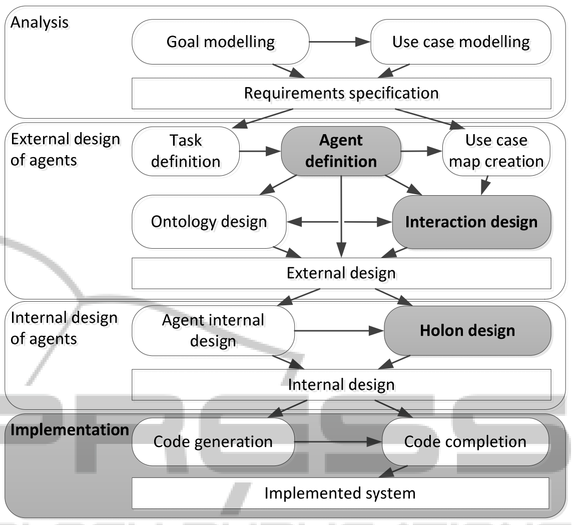


Figure 3: Structure of the extended version of the MASITS methodology.

Additionally the usage of the holonic architecture provides the following possibilities:

- The architecture of the system is hierarchical enabling to implement any agent as a multi-agent system. As a consequence system can be of any complexity, because it is divided into holons that are designed separately.
- Open system. It is possible to create an open system by defining what types of agents can be added to the system. It is extremely important if the system may need to implement some function in new ways, for example to provide new types of learning materials in ITSs.

Finally, the methodology specifies use of a particular implementation environment. As discussed in (Lavendelis and Grundspenkis, 2009a) it simplifies the practical development process, but also adds the following limitations: (1) System is implemented in JADE platform (Bellifemine et al, 2007) and Java programming language; (2) the system consists of behaviour based agents.

5 CASE STUDY

A simulation tool for automated interactions between insurance companies and their clients is developed as a case study of the extended version of the MASITS methodology. The tool automates travel insurance market by implementing interactions among a client and Latvian insurance companies during policy buying (Lavendelis and Grundspenkis, 2014). The tool implements auction mechanism where a client is an auctioneer and the insurance companies are bidders.

The simulation tool has three main modules, namely, client module, insurance company module and monitoring module. Thus the system consists of three different modules that can be implemented as agents. The mechanisms needed to represent both parties in the market (client and insurance companies) need intelligence to be executed adequately. First of all components need autonomy to work in the simulated marketplace. Secondly, they need reactivity to monitor the situation in the market and proactivity to find the correct strategies that achieve the goals of the corresponding stakeholder. So one can conclude that the insurance market automation tool complies with the characteristics of systems that can be developed by the extended version of the MASITS methodology and at the same time it is from different domain than ITS and thus was chosen as a case study for the new version of the methodology.

All the steps of the MASITS methodology were done to develop the simulation tool. In this chapter the use of the three extensions proposed above will be outlined.

During the agent definition step it was concluded that two types of the stakeholders must be autonomously represented in the insurance policy market, namely the client and insurance companies. Thus tasks were grouped and agents were defined according to the stakeholders. As a result, each insurance company and the client are represented by particular autonomous agents, namely client agent and company agents. The market monitoring agent is added to carry out the functions of the monitoring module, in particular, to monitor the actions done by the agents of insurance companies, so ensuring that they comply with the legislation. As a consequence the higher level of the system will contain these three types of agents.

During the whole development process the MASITS tool (Lavendelis and Grundspenkis 2009b) was successfully used for the automation of the development process. It was used for the following tasks:

- Drawing all diagrams. The task hierarchy assigned to the company agent is shown in the Figure 4 as an example of the diagram developed in the MASITS tool.
- Connecting elements with the same meaning in different diagrams ensuring the compliance between diagrams created during different stages of the development.
- Lastly, it was used for Java code generation of JADE agents and their behaviours.

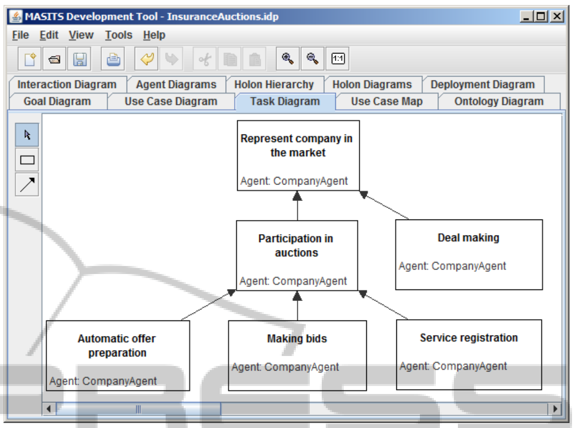


Figure 4: Task hierarchy assigned to the company agent.

The interaction design was done in two levels. First the interaction diagram was created. Initially the interactions were designed in the interaction diagram. Afterwards protocol diagrams were created for the auction protocols. This approach had two advantages comparing to the initial version of the MASITS methodology and other methodologies: (1) the interaction diagram was kept simple (see Figure 5) and (2) the order and context of messages was specified. The messages sent in the protocols were designed according to the FIPA standards (FIPA 2014). The interaction design step was the last step that differed from the initial version of the MASITS methodology.

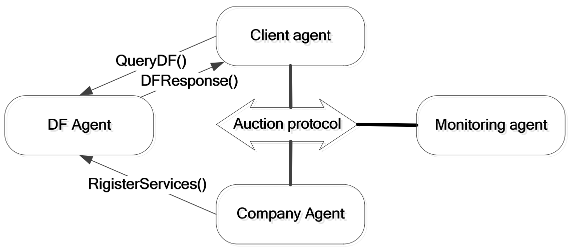


Figure 5: The interaction diagram.

The case study showed the advantage over other methodologies that the extended version of the MASITS methodology keeps from the original one, namely, the easy transformation from the design to the implementation. It is ensured by the following two facts. Firstly, the design is done in terms of the JADE platform and no transformation to the implementation concepts is needed. Secondly, the MASITS tool generates the code of the agent and behaviour classes and only the method code must be completed manually.

As a result of the development process a simulation tool was successfully implemented. The user interface of the developed simulation tool is given in the Figure 6. It is in Latvian, because Latvian insurance market is modelled.

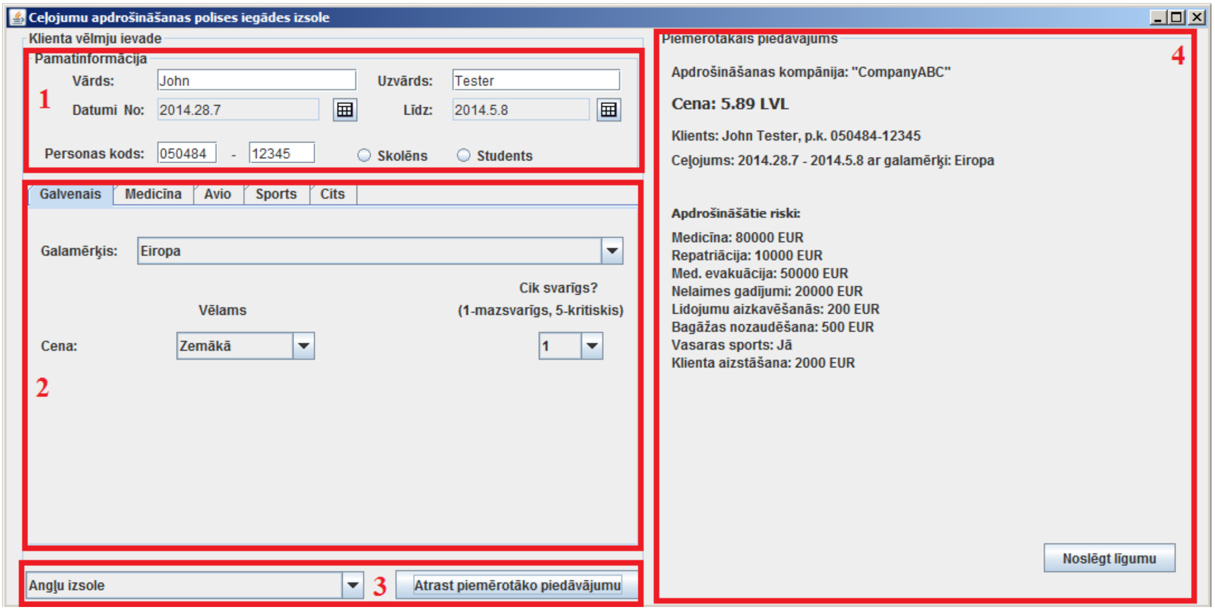
The usage of the developed simulation tool consists of the following steps:

1. Input of the information about the client and the travel details for which the insurance is needed; The information about the client and travel dates must be entered in the upper left area denoted with 1 in Figure 6. The area denoted with 2 in Figure 6 is for input of preferred values and importance weights of all the criteria used in the evaluation of the deal. The first criterion is price and other criteria are coverage of different insurance risks. The tabs in this area splits criteria input into thematic groups.
2. Auction choice and start of the auction. The client must select the auction type (English, Dutch, First price sealed bid and Vickrey auctions are available) and start the execution of the auction. It is done in the area of the user interface denoted by 3 in the Figure 6. Afterwards client agent carries out the chosen type of the auction autonomously and finds the winning deal. The best deal is determined by multiple criteria. Thus the multi-criteria insurance deal evaluation model and multi-criteria auction protocol presented in (Lavendelis and Grundspenkis, 2014) are used.
3. Data output. The data about the winning offer is presented in the right part of the user interface that is denoted by 4 in the Figure 6. If the client is not satisfied with the offer, he/she can change some input data and go through the steps 2 and 3 again.

One of the significant advantages of the extended MASITS methodology compared to other methodologies, is the fact that it allowed to implement the openness of the system in the following way. The insurance company agents can join or leave the system at any time, because the list of companies working in the insurance market may change dynamically. Thus the higher level holon is open for new agents of this type. Any agent that is capable to follow auction protocols may register at the directory facilitator agent of the JADE platform (Bellifemine et al, 2007) and become the member of the system.

6 CONCLUSIONS

The paper presents an extension of the MASITS agent oriented software engineering methodology that enables to use it for other systems than ITSs. As a result, the extended methodology is not any more specific to ITS development, but for larger class of agent oriented software. It was proved by the development of the market simulation tool.



Vārds: John
Uzvārds: Tester
Datumi No: 2014.28.7
Līdz: 2014.5.8
Personas kods: 050484 - 12345
○ Skolēns ○ Students

Galamērķis: Eiropa

Vēlams
Cik svarīgs? (1-mazsvarīgs, 5-kritiskis)
Cena: Zemākā 1

Angļu izsole
Atrast piemērotāko piedāvājumu

Apdrošināšanas kompānija: "CompanyABC"

Cena: 5.89 LVL

Klients: John Tester, p.k. 050484-12345

Ceļojums: 2014.28.7 - 2014.5.8 ar galamērķi: Eiropa

Apdrošināšātie riski:

Medicīna: 80000 EUR
Repatriācija: 10000 EUR
Med. evakuācija: 50000 EUR
Nelaimes gadījumi: 20000 EUR
Lidojumu aizkavēšanās: 200 EUR
Bagāžas nozaudēšana: 500 EUR
Vasaras sports: Jā
Klienta aizstāšana: 2000 EUR

Figure 6: The user interface of the developed insurance market simulation tool.

The paper provides a new approach to develop new AOSE methodology for development of systems with certain characteristics. It is done not by specializing a general purpose methodology and adapting the techniques used there, but by taking more specific methodology and identifying the steps that are too specific and generalizing them. The advantage of this approach is the fact that the specific methodology has been validated in the development of systems with similar characteristics.

It was chosen to leave the methodology implementation platform specific, so limiting the use of the methodology, because as it was concluded in (Padgham and Winikoff, 2004) and (Lavendelis and Grundspenkis, 2009a) the methodologies that are not implementation platform specific have very weak support of the implementation phase, because it is not possible to support transition from the design time concepts into all possible implementations. An example of such methodology is Gaia (Zambonelli et al, 2005). Contrary, the choice of the implementation platform at the design time enables easy transition to the implementation. In the MASITS methodology this transition is supported by code generation algorithms and MASITS tool implementing them.

One of the directions of the future work is to develop more case studies of the extended version of the methodology. The case study in the transportation and logistics domain is currently under development at Riga Technical University.

ACKNOWLEDGEMENTS

The work has been partly supported by Latvian National Research Programmes IMIS and SOPHIS.

REFERENCES

Bellifemine F. L., Caire G., Greenwood D., 2007. Developing Multi-Agent Systems with JADE. Wiley, 300 p.

Cortese, E. et al., 2005. JADE Test Suite – USER Guide. Available online: http://jade.tilab.com/doc/tutorials/JADE_TestSuite.pdf (Last visited: 24.07.2014).

DeLoah S., 2014. O-MaSE An Extensible Methodology for Multi-Agent Systems. In *Agent Oriented Software Engineering*, pp 173-192.

Fischer K., Schillo M., Siekmann J., 2003. Holonic Multiagent Systems: A Foundation for the Organisation of Multiagent Systems, *Lecture Notes in Computer Science* 2744, Springer.

FIPA, 2014. FIPA interaction protocol specifications. Available online: http://www.fipa.org/repository /ips.php3 (Last visited: 27.07.2014).

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

Huget M.P., Odell J., 2005. Representing Agent Interaction Protocols with Agent UML. *Agent-Oriented Software Engineering V. Lecture Notes in Computer Science*. Volume 3382, 2005, pp 16-30.

Kendall, E.A., Malkoun, M.T., Jiang, C.A., 1995. Methodology for Developing Agent Based Systems for Enterprise Integration. In *IFIP Working Conference of TC5 Special Interest Group on Architectures for Enterprise Integration*, Queensland, Australia, November 1995.

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

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

Lavendelis E., Grundspenkis J., 2009b. MASITS - A Tool for Multi-Agent Based Intelligent Tutoring System Development. *Advances in Intelligent and Soft Computing* Vol. 55. Springer, pp. 490-500.

Lavendelis E., Grundspeņķis J., 2010. MIPITS - An Agent based Intelligent Tutoring System. *Proceedings of 2nd International Conference on Agents and Artificial Intelligence (ICAART 2010)*. Vol. 2., Spain, Valencia, January 22-24, 2010. pp. 5-13.

Lavendelis E., Grundspenkis J., 2011. MASITS Methodology Supported Development of Agent Based Intelligent Tutoring System MIPITS In *Communications in Computer and Information Science (CCIS)*, Vol. 129. Springer, 2011, pp. 119-132.

Lavendelis, E., Grundspenkis, J. 2014. Multi-Agent Auction Based Simulation Tool for an Insurance Policy Market. *Applied Computer Systems*. Vol.15, 2014, pp.5-13.

Padgham L. and Winikoff M., 2004. Developing intelligent agent systems. A practical guide. New York. John Wiley and Sons. 240 p.

Varga, L.Z., Jennings, N.R., Cockburn, D., 1994. Integrating intelligent systems into a cooperating community for electricity distribution management. In *International Journal of Expert Systems with Applications* 7 (4), pp. 563-579.

Zambonelli, F., et al., 2005. Multi-Agent Systems as Computational Organisations: The Gaia Methodology. *Agent-Oriented Methodologies*, Idea Group Publishing, London, pp. 136-171.