BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **18**, No 2
Sofia • 2018 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2018-0033

Using Agent-Based Methodologies in Healthcare Information Systems

Reem Abdalla[1], *Alok Mishra*[2]

[1]*Department of Modeling & Design of Engineering Systems, Atilim University, Ankara, Turkey*
[2]*Department of Software Engineering Atilim University, Ankara, Turkey*
E-mails: *reemelsaetii@yahoo.com* *alok.mishra@atilim.edu.tr*

Abstract**:** *This paper carries out a comparative analysis to determine the advantages and the stages of two agent-based methodologies: Multi-agent Systems Engineering* (*MaSE*) *methodology, which is designed specifically for an agent-based and complete lifecycle approach, while also being appropriate for understanding and developing complex open systems; Agent Systems Methodology* (*ASEME*) *suggests a modular Multi-Agent System* (*MAS*) *development approach and uses the concept of intra-agent control. We also examine the strengths and weaknesses of these methodologies and the dependencies between their models and their processes. Both methodologies are applied to develop The Guardian Angle: Patient-Centered Health Information System* (*GA: PCHIS*)*, which is an example of agent-based applications used to improve health care information systems.*

Keywords**:** *Agent-based Methodologies, MaSE, ASEME.*

1. Introduction

With the increasing importance of complex software systems in the software industry, the need for using agent technologies have increasing to facilitate in large-scale commercial and industrial software systems development. It is of crucial need for system modelling techniques to support reliable, maintainable and extensible designs. With these new challenges, software development has become increasingly difficult. Thus, it is important that software companies develop tailor made processes for constructing inherently complex and distributed software to run in such environments and be able to live up to the task. Compared to traditional software engineering paradigms, Agent Oriented Software Engineering (AOSE) allows for improved functionalities and security [1-3].

Multi-Agent Systems (MAS) consist of autonomous agents, which interact within a dynamic environment to accomplish their common individual goals. Achieving these goals usually requires effective coordination of all the activities assigned to the agents [20]. The primary advantages of using multi-agent technologies include: (1) individuals agents take into account the application-specific nature and environment; (2) local interactions between individuals can be modelled and investigated; and (3) difficulties in modelling and computation can be organized as sub layers and/or components. Therefore, MASs provide satisfactory solution to the issue of distributed control as a computational paradigm. In addition, Artificial Intelligence (AI) techniques can also be utilized in line with these efforts [22]. With the increasing importance of complex software systems in the industry, the need for using agent technologies has become even more paramount for developing in a sustainable way large-scale commercial and industrial software systems. In line with these priorities, it is of crucial need for system modelling techniques to support reliable, maintainable and extensible designs [21].

However, unlike Object-Oriented (OO) technology which uses Unified Modeling Language (UML), there is no unified method that can be used to develop agent systems. In this, regard, AOSE can also offer ways for comparing and assessing agent methodologies to help developers in choosing between the current and available methodologies [4].

Over the last two decades, health care systems have become increasingly computer-based [23], requiring the ability to save and organize large amounts of medical information pertaining to patients. Using agent technologies, this task will be faster and more reliable [14]. Health care information systems have become more and more computerized. A large amount of data in this sector needs to be stored and analyzed, and with the help of computer systems, this task can be done faster and more efficiently.

The present work is an attempt to analyze the Guardian Angel: Patient-Centered Health Information System **(**GA: PCHIS) [5] using two agent methodologies: Multi-agent Systems Engineering (MaSE) [6, 7] and Agent System Methodology (ASEME) [8, 9]. These two support the engineering of a large array of open system and allow for the complexity and knowledge presentation in the MAS to be scaled modularly to any level required. These methodologies are clearly agent-based, the design is widely regarded as agent-oriented, and both consider the social aspects associated with the tasks involved [10-12].

2. MaSE methodology

The MaSE process [6, 7] includes seven steps split into two phases: the analysis phase which in itself consists of three steps: capturing goals, applying use cases, and refining the roles; and the design phase that covers four steps: Creating agent classes, constructing conversations, assembling agent classes, and system design.

2.1. Analysis stage

The first step in the analysis stage is capturing goals, which takes the primary system specifications and converts them into an organized collection of system goals. It is shown in Fig. 1.

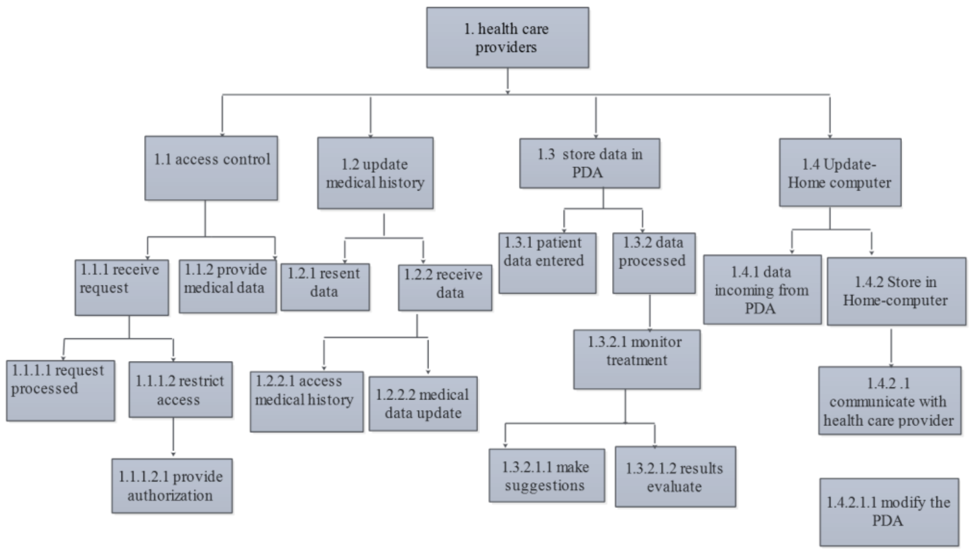
1. health care providers
1.1 access control
1.2 update medical history
1.3 store data in PDA
1.4 Update-Home computer
1.1.1 receive request
1.1.2 provide medical data
1.2.1 resent data
1.2.2 receive data
1.3.1 patient data entered
1.3.2 data processed
1.4.1 data incoming from PDA
1.4.2 Store in Home-computer
1.1.1.1 request processed
1.1.1.2 restrict access
1.2.2.1 access medical history
1.2.2.2 medical data update
1.3.2.1 monitor treatment
1.4.2 .1 communicate with health care provider
1.1.1.2.1 provide authorization
1.3.2.1.1 make suggestions
1.3.2.1.2 results evaluate
1.4.2.1.1 modify the PDA

Fig. 1. Goal hierarchy diagram

Applying use cases is an inter-agent conversation and establishes the true backbone of a MAS because it enables the formation of a distributed process as a powerful agent methodology. The use case models capture such cases from early system requirements and restructure them as a sequence diagram (Fig. 2). A sequence diagram describes a sequence of messages exchanged among multiple agent roles.

However, to assign the relationship between multiple roles and one agent is not trivial. Many design quality factors need to be considered [24]. Software quality has long been a critical issue for software developers [25, 26]. First, use cases are extracted from the system requirements in the analysis stage to describe a series of events that determine the behavior of the desired system.

A sequence diagram is used to define a minimum set of messages that should be passed among the roles. If a message is exchanged between two roles, then there should be a corresponding connection path between the two. The communication path created among the roles by independent agent classes imply that there must be a conversation among agent classes to pass the message.

Refining roles converts the goals in the goal hierarchy diagram into more useful forms for building MAS roles.

Roles are constructing blocks used to capture system goals and determine the agents' classes in the design stage. The role is an abstract description of the expected function of the enterprise and covers the goals of the target system to which it has been assigned [16, 17]. The general shift of goals into roles is one-to-one, and every goal maps to one role [10].

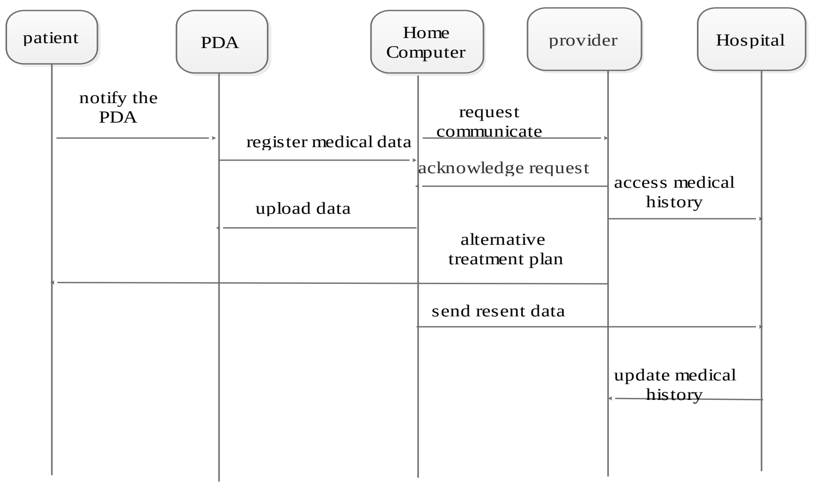

Fig. 2. GA:PCHIS sequence diagram

Refining roles converts the goals in the goal hierarchy diagram into more useful forms for building MAS roles.

Roles are constructing blocks used to capture system goals and determine the agents' classes in the design stage. The role is an abstract description of the expected function of the enterprise and covers the goals of the target system to which it has been assigned [16, 17]. The general shift of goals into roles is one-to-one, and every goal maps to one role [10].

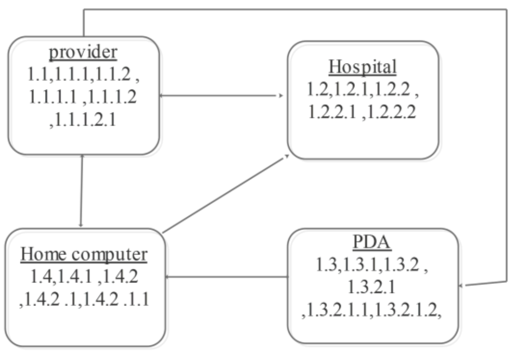

Fig. 3. Traditional role model for the GA:PCHIS

Fig. 3 shows a traditional role model for the GA:PCHIS. The lines among the roles in this model refer to possible connection paths. These paths are derived from the sequence diagrams developed in the previous step. Fig. 4 shows clearer and more complete version of the role model, which contains information on the interactions among role tasks. The goals related with each role are represented under the role name. Also illustrated are the tasks related to each role, used to determine each role’s behavior.

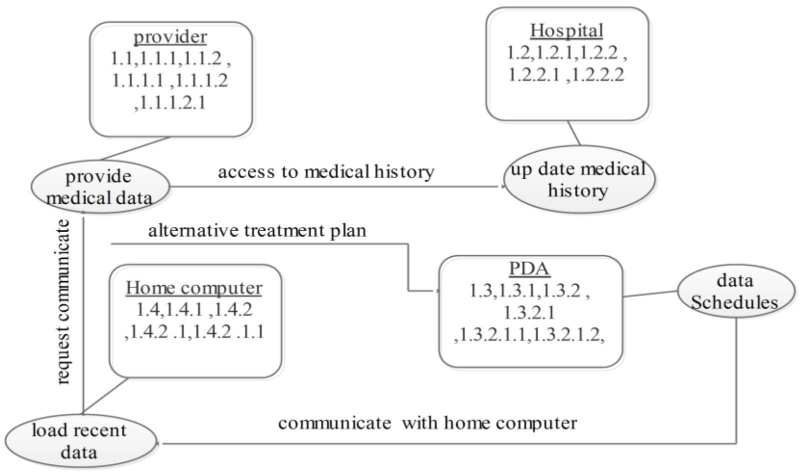

Fig. 4. GA:PCHIS role model

2.2. The design stage

The design stage of the MaSE translates the agent classes to actual agents. It describes the numbers, types, and locations of agents within a system by using a deployment diagram. Since most of the work has been done in the previous stage, the design of the system, in essence, is the simplest stage within MaSE.

The main objective of this step is to construct a number of agent classes determined from the component roles; constructing conversations using communication task diagrams as a type of finite state machine, where every conversation is depicted by two communication class diagrams; one is for the initiator of the conversation and the other for the responder. Each diagram displays the sequence of the messages exchanged between the initiator and the responder.

3. ASEME methodology

The ASEME methodology [8, 9, 15] consists of three steps: requirements, design, and implementation. It also covers a modular agent development method and offers the concepts of intra-agent control to determine the agent's behavior by formatting the various modules which execute this tasks. In addition, the Inter-Agent Control (IAC) defines the protocols that control the coordination of the agents' society.

3.1. Requirements analysis

At this stage, the participating actors are identified along with their respective goals. Furthermore, information is collected about the specific requirements that dictate the expected system functions. Initially, the analyst determines which actors will act in the system. Next these actors are related with their goals. As a final stage, the specific requirements associated with each goal of each actor are defined [15]. Fig. 5 describes the actor diagram for the target system.

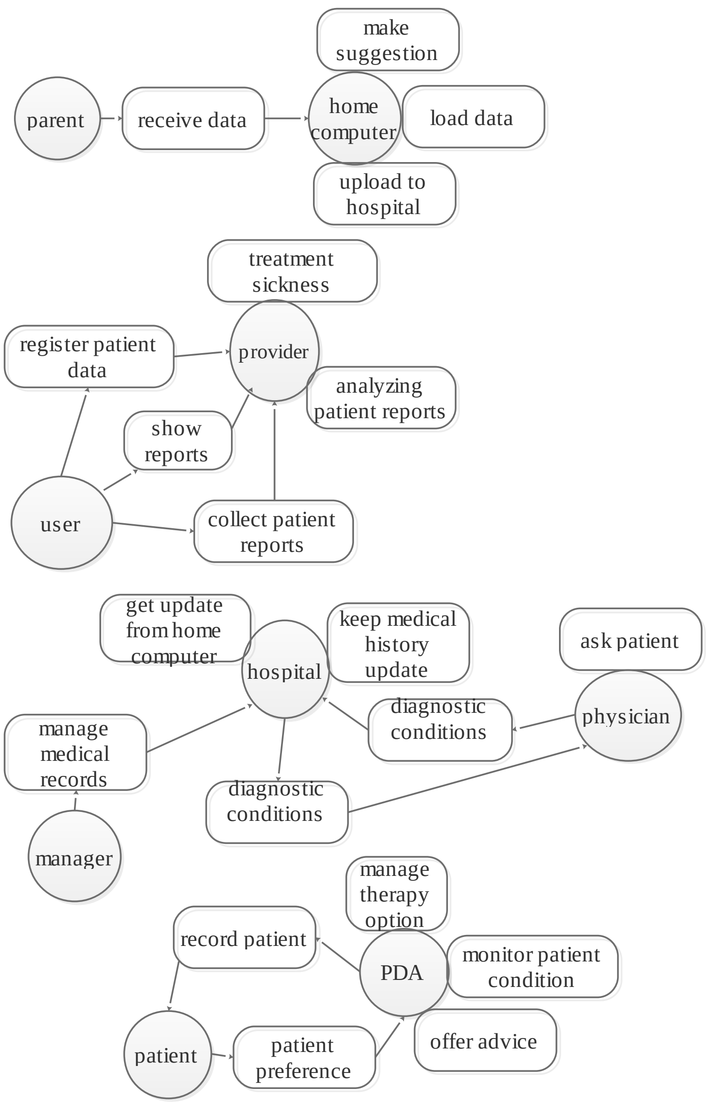

Fig. 5. GA:PCHIS actor diagram

3.2. Analysis phase

There are certain concepts involved in an analysis [9, 15]. In the following these concepts are introduced and explained in brief:

Roles. Human and agent roles can correspond to the actors of the requirements analysis phase as concrete roles. These roles are initially represented in the use case diagram and, then, moved to the roles model, which includes the role name, the inter-agent protocols which the role participates in, and its liveness model as shown in Fig. 6.

ROLE: PDA
Protocols: get patient data involves Store patient's information.
Liveness:
agent = (data schedules)$^{\omega}$ //(monitor patient condition$^{)}$$^{\omega}$
data schedules =downloaded data. store data.
monitor patient condition = get patient data. data schedules. monitors progress*, data update.

Fig. 6. The role model, including three liveness formulas

Use-Cases. The use case model shows the relation between the system and its environment while also defining the system functionality. Use cases are derived from the goals of the actor diagram in the previous phase (Fig. 7).

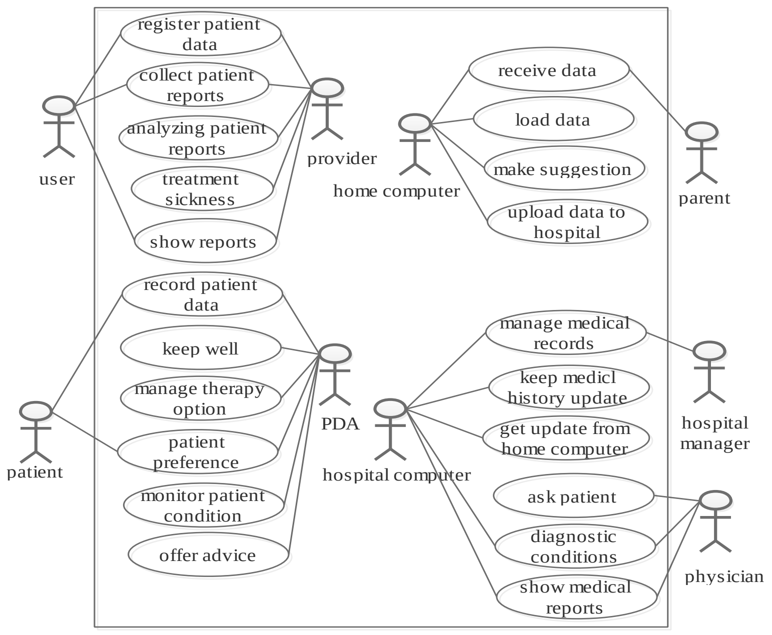

Fig. 7. GA:PCHIS use case diagram

Capabilities. In general, an agent’s capabilities describe what the agent can actually achieve. The capabilities derived from the use case diagram are related to the tasks and can be done by the role itself or through interaction with other roles.

Functionalities. A functionality is associated with the various types of technologies used to implement reasoning mechanisms.

Activities. Each capability is broken down into simple activities.

Agent Interaction Protocols. A protocol is implemented as a capability.

Liveness model. This is a process model which shows the dynamic behavior of the roles in the system.

4. Results and discussion

4.1. MaSE

MaSE is the first methodology used in this paper, in its latest form, and updated as it appears in [6, 7]. This methodology is constructed based on the application of current techniques for the OO. As a software engineering approach, agents are described as finite state machines. The focus is on communication modeling, and the control flow is also presented within a very clear plan.

In the MaSE methodology, every agent acts as a software process that interacts with another agent or software process to accomplish a common overall goal. Even though some agents are intelligent and designed efficiently while others are not, all are treated equally; that is, they all work in the same manner to achieve the required goal. MaSE can also be regarded as an environmental and deployment model.

In MaSE, several software agent applications are encapsulated and all their external interfaces are closed by an agent involved in the system communication protocol. However, one limitation of MaSE is that the open systems are not taken into consideration in the process, as a result of which agents cannot be created, deleted or moved during implementation. Also, in exchange for multicast, the interactions among the agents of a target system are to be on a one-to-one basis. For this reason, systems employing MaSE are not considered as extensive.

Goal analysis, initially performed in the MaSE process, enhances goal preservation through the stages of analysis and design. The role and modeling of the agent class is facilitated by focusing upon assigning a clear goal, each of which is intended to be fulfilled. Also, there are tasks that belong to the custom goals of the roles.

4.2. ASEME

The second methodology used in this study was ASEME. To build MAS, this methodology uses model-driven techniques. The different models included are represented using meta-models with the Eclipse Modeling Framework (EMF). ASEME uses the Agent Modelling Language (AMOLA), which describes both syntax and semantics for building MAS models, in this way supporting the analysis and design phases of the software development process. AMOLA also copes with the individual and societal side of the agents, describing how protocols and capabilities can be used in the agents' design. In ASEME the guidelines are missing because the authors depend only on automated model transformations. The same is true for tasks to be done at each stage of development.

4.3. MaSE & ASME

MaSE illustrates, for the first time, both inter and intra-agent communications that should be integrated. However, MaSE models alone cannot provide a modeling technique for analyzing systems and allowing for model transformation in between the analysis and design stages. The concurrent tasks model is determined from the goal hierarchy tree and from sequence diagrams in a manner that cannot be automated. In ASEME the model transformation process is typical [18, 19].

For obtaining the design stage intra-agent control from the analysis phase liveness model, MaSE offers simple rules but the guidelines are missing because the authors rely only on the paradigm shifts [18]. Agents in MaSE are associated with the target system goals, while ASEME defines agent types which arise from the actors of the requirements stage. Finally, ASEME allows for greater implementation possibilities, while in MaSE agents are executed using AgentTool as in [19].

5. Conclusion

The main aim of this case study was to achieve greater functionality and flexibility in such type of complex open systems as health care information systems. The successful use of the two selected agent methodologies in designing the GA: PCHIS indicates that they are practical and usable.

The notation of the two methodologies is generally acceptable and both have a strong modeling language in terms of satisfying different criteria such as those of systematic transitions, modularity and ease of comprehension.

Regarding the processes, from the software development lifecycle point of view, both of the methodologies include the specification and analysis design as well as detailed design to some extent. Additionally, the selected methodologies provide techniques and adequate support for abstractions, allowing for the complexity and knowledge in the MAS to scale modularly to any arbitrary level required.

Moreover, there are distinctive features, such as Model-Driven Engineering (MDE), documentation of non-functional requirements phases in ASEME and deployment model in MaSE.

The evaluation of the AOSE method shows that most effort has been dedicated to requirements, design, and implementation phases. However, advances are still needed in all stages of the software lifecycle. With regard to the future work, it includes increasing the number of agent-oriented methodologies and adding other evaluation methods towards the assessment. In doing so, the work can be improved by involving supplementary models and techniques.

References

1. B r a d S h a w, J. Introduction to Software Agents. – In: Software Agents, J. BradShaw, Ed. Menlo Park, California, AAAI Press, 1997, pp. 3-46.
2. W o o l d r i d g e, M., N. R. J e n n i n g s. Intelligent Agents: Theory and Practice. – The Knowledge Engineering Review, Vol. **10**, 1995, No 2, pp. 115-152.
3. H o a, K. D., W. M i c h a e l. Towards a Next-Generation AOSE Methodology. – Science of Computer Programming, Vol. **78**, 2013, No 6, pp. 684-694.
4. M a n s u r a, H. Metrics for Evaluating Agent Oriented Software Engineering Model. – In: IEEE/OSA/IAPR International Conference on Informatics, Electronic &Vision, Dhaka, Bangladesh, 4 October 2012. DOI: 10.1109/ICIEV.2012.6317459.
5. E r i c s s o n, N i k o l a T e s l a d. d. Technical Description. – Integrated Health Care Information System. Croatia, April 2004.

6. D e L o a c h, S. A., M. F. W o o d. Multiagent Systems Engineering: The Analysis Phase. – Technical Report, Air Force Institute of Technology, AFIT/EN-TR-00-02, June 2000.

7.W o o d, M. F. Multiagent Systems Engineering: A Methodology for Analysis and Design of Multiagent Systems. MS Thesis, AFIT/GCS/ENG/00M-26. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, Ohio, USA, 2000.

8. S p a n o u d a k i s, N. The Agent Systems Engineering Methodology (ASEME). Ph. D. Thesis, Paris Descartes University, 2009.

9. S p a n o u d a k i s, N. I., P. M o r a i t i s. The Agent Modeling Language (Amola). – In: AIMSA’2008, LNCS (LNAI). Vol. **5253**. D. Dochev, M. Pistore, P. Traverso, Eds. Heidelberg, Springer, 2008, pp. 32-44.

10. I g l e s i a s, C. A., M. G a r i j o, J. C. G o n z a l e z. A Survey of Agent-Oriented Methodologies. – In: Intelligent Agents V (Atal’98), LNAI 1555. J. P. Muller, M. P. Singh, A. Rao, Eds. Berlin, Springer-Verlag, 1999.

11. T v e i t, A. A Survey of Agent-Oriented Software Engineering. First NTNU CSGS, 2001.

12. W e i s s, G. Agent Oriented Software Engineering. Knowledge Engineering Review. – In: Second International Workshop on Agent-Oriented Software Engineering (AOSE’02), Vol. **16**, 2001, No 4, pp. 101-108.

13. L a w, D. Methods for Comparing Methods: Techniques in Software Development. NCC Publications, 1988.

14. S z o l o v i t s, P., J. D o y l e, W. J. L o n g. Guardian Angel: Patient-Centered Health Information Systems. Tech Report MIT/LCS/TR-604, 2004.

15. S p a n o u d a k i s, N. I., P. M o r a i t i s. Using ASEME Methodology for Model-Driven Agent Systems Development. – In: AOSE XI, LNCS. Vol. **6788**. D. Weyns, M. P. Gleizes, Eds. Springer, 2011, pp. 106-127.

16. K e n d a l l, E. A. Agent Software Engineering with Role Modelling. – In: LNCS. Vol. **1957**, 18 June 2002.
https://link.springer.com/bookseries/558

17. K i n n y, D., M. G e o r g e f f, A. R a o. A Methodology and Modelling Technique for Systemsof BDI Agents. Agents Breaking Away. – In: Proc. of 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW’96). Lecture Notes in Artificial Intelligence. Vol. **1038**. Berlin Heidelberg, Springer-Verlag, 1996. pp. 56-71.

18. G o m e z-S a n z, J. J., F. R u b e n. Understanding Agent-Oriented Software Engineering Methodologies. – The Knowledge Engineering Review, Vol. **30**, 2015, No 4, pp. 375-393.

19. S p a n o u d a k i s, N., P. M o r a i t i s. The Agent Systems Methodology (ASEME): A Preliminary Report. –In: Proc of 5th European Workshop on Multi-Agent Systems, Hammamet, Tunisia, December. (EUMAS’07), 13 December 2007, pp. 13-14.

20. J u a n, A., S. C a r l e s, A. J o s e p l i , L. M a i t e, R. I n m a c u l a d a. Towards Next Generation Coordination Infrastructures. – The Knowledge Engineering Review, Vol. **30**, 2015, No 4, pp. 435-453.

21. M e h d i, D. Programming Multi-Agent Systems. –The Knowledge Engineering Review, Vol. **30**, 2015, No 4, Cambridge University Press, pp. 394-418.

22. J i n g, X i e, C h e n-C h i n g L i u. Multi-Agent Systems and Their Applications. – Journal of International Council on Electrical Engineering, Vol. **7**, 2017, No 1, pp. 188-197.

23. A b d a l l a, R., A. M i s h r a. Application of Agent Methodology in Health Care Information System in TEM Journal (Technology Education, Management). – Informatics, Vol. **6**, 2017, No 1, pp. 147-152.

24. Y u, L., A. M i s h r a. Multiple-Role Agent Based Distributed Computing. – Technics Technologies Education Management, Vol. **8**, 2013, No 1, pp. 238-243.

25. M i s h r a, D., A. M i s h r a. Simplified Software Inspection Process in Compliance with International Standards. – Computer Standards & Interfaces, Vol. **31**, 2009, No 4, pp. 763-771.

26. M i s h r a, D., A. M i s h r a. Software Quality Assurance Models in SME – A Comparison. – International Journal of Information Technology and Management**,** Vol. **5**, 2006, No 1, pp. 4-20.

Received 19.01.2018; Second Version 09.04.2018; Accepted 20.04.2018