THE EXPERIENCE OF THE AGENT-BASED SIMULATION SYSTEM DEVELOPING

Elena Zamyatina, Danil Karimov, Artiem Mitrakov,

Abstract: *This paper discusses the problems of the agent-based simulation system design. It is well known that agent models extend the capabilities of simulation for solving some problems that can't be solved by the methods of system dynamics and discrete event simulation. Particular attention in the design and the implementation of agent-based simulation authors pay to the problems of distributed simulation and the problems of intelligent agent's implementation and the use of ontologies at all stages of the simulation experiments.*

Keywords: *simulation, agent-based model, distributed simulation, intelligent agent*

ACM Classification Keywords: *I.6 SIMULATION AND MODELING I.6.8 Types of Simulation - Distributed : I.2 ARTIFICIAL INTELLIGENCE I.2.5 Programming Languages and Software - Expert system tools and techniques*

Introduction

The problems encountered in everyday life (in industry, commerce, management, etc.) become more complex, so it is necessary to develop new and to improve existing methods for solving these problems. One of the methods for solving complex problems in various areas of activity is a simulation method.

It is well known that simulation is very important in the investigations of complex dynamical systems. Simulation methods application are rational if it is very difficult to formalize a problem or in the case when analytical and numerical methods require strong abstractions to avoid some sufficient details.

Two classes of simulation systems exist: continuous and discrete-event simulation (DES). But some problems can't be described by continuous and discrete-event simulation models. This is true with respect to the economic-mathematical models for example (these models describe the processes occurring in the, ecological and economic systems in the form of equations and inequalities). Modern models are dynamic and they have large dimensions. So designed simulation models may be valid but contradictory.

So the application of new techniques that allow us to solve the problem of managing the complex objects becomes relevant. One of the methods is the method of agent-based modeling. The essence of the agent-based modeling is that the local behavior of agents operating under their own rules defines the global behavior of the whole system (the concept of designing "bottom-up"). This differs from traditional approaches to a simulation model design. The traditional approach supposes «top-down» design. Investigator defines a set of global laws of the system behavior and the elements of this system operate on basis of these global laws. But global functioning of the agent-based simulation system is not known to the investigator. Two or three simple rules can already lead to very diverse forms of behavior

in a group of agents. An example is a boids-modeling and cellular automata theory [Macal & North, 2005; Macal & North, 2009].

The need for the development, analysis and a business process reengineering is another argument in favor of the agent-based simulation [Klyshinskii, 2000]. Some operations in a business process are executed by the objects that determine their own behavior (decision makers, automated and robotic systems, objects that are managed by humans). The behavior of the object is determined by a set of rules. The behavior of this object may differ due to particular situation: it should consider the impact of other objects and the impact of the external environment.

So let us entertain the hypothesis that an agent is an independent (autonomous) system having the opportunity to take effect from the outside world, to determine its reaction to this effect and to carry out its reaction, and an intelligent agent - an agent which has some knowledge about itself and the world around it, at that its reasonable behavior is determined by its knowledge. There are many works that show the relevance of the application of agent-based approach in the marketing of [Ivashkin, 2003], in the simulation of situations that occur in the auction [Gribanova, 2006], inventory management, supply chains, etc.

Currently there are many software systems (most often - the software libraries) that implement agent-based simulation. We'll discuss the various software systems below. These systems are both domestic and foreign, both paid and freeware. This paper considers the specific features of architecture of agent-based simulation systems and suggests some approaches to improve the adaptability of the software product, the effectiveness and reliability of the agent-based simulation experiment.

Review of existing agent-based simulation systems

Let us discuss the specific characteristics of agent-based simulation systems and special libraries for the creation these systems:

- ***Anylogic.*** AnyLogic [Borshchev, 2014, Borshchev et al, 2012] is one of the famous agent-based simulation systems in Russia, and it is well known all over the world. AnyLogic permits to use not only continuous modelling but discrete-oriented and agent-based modelling too. It has graphical interface and the set of standard libraries. These tools make the process of simulation model design simpler and allow designing simulation models for more wide class of problems. AnyLogic was implemented on the base of Java language (so it is possible to say that Anylogic is a cross-platform system), moreover the power of Anylogic may be extended if the investigator includes into a simulation model some modules developed with the help of Java.

  It is possible to develop Java-applets. These applets can be opened by various browsers.

  So the program tools of AnyLogic allow designing, implementing and documenting the simulation models, it is possible to collect a set of statistical data during the simulation run, to analyze a simulation model and to optimize it. But intellectual agents not implemented in AnyLogic and it isn't a distributed one (we must notice that some publications [Borshchev et al, 2012] concerning the developing of program tools supporting distributed simulation in AnyLogic appeared).

- ***BPSim***. BPSIM – it is another agent-based simulation system [Aksienov, 2013]. This simulation system is dedicated to the simulation of business processes. BPSIM is the program realization of the resource consumption multi-agent conversion model. The agents control the process objects of resource conversion. The agents analyze a current situation, refer to the knowledge base, find a solution, control the purpose gaining and exchange messages.

  The program tools BPSIM allow to develop a conceptual model, dynamic model, to carry out a simulation experiment and to export the results to EXCEL. The system contains (a) reactive agents, (b) reactive-intelligent agents, (c) intelligent agents (their behavior is described by the planning system, the knowledge is stored in the knowledge frame base, applied for construction of complicated advising expert system and so on), (d) hybrid agents (construction of complicated planning systems).

  But one can hardly speak that BPSim is cross-platform software (code is generated in Delphi). Moreover BPSim software does not support a distributed (parallel) simulation.

- ***REPAST:*** REcursive Porous Agent Simulation Toolkit (Repast) [Repast, 2015] – it is an open source and freely available libraries for large-scale agent-based modeling. Repast supports the development of flexible agent-based models and it may be used in modeling of social processes, in marketing and logistics. A modeler builds simulation model including components from open source libraries into his program. One can use Visual Repast. There are three versions of Repast: Repast for Python (Repast Py), Repast for Java (Repast J) and Repast for the Microsoft.NET framework (Repast .NET). Repast use MatLab, SQL, Excel for the processing of the results of simulation experiments. Repast is very powerful program tool but user must be skilled programmer.

- ***NetLogo:***NetLogo [NetLogo, 2015] **-**simulation environment designed to create models describing the natural and social phenomena. This system allows the models to operate "on the fly" dynamically changing the behavior of the system under the influence of various conditions. The system is quite simple and it allows the users without programming skill to open and run models saved in libraries. These models can be used again, or modified. Moreover users may build their own simulation models. NetLogo modeling environment is cross-platform.

- ***Mason:*** Agent-oriented simulation system Mason [Mason, 2015] - is a multi-agent simulation environment and it is represented as a set of libraries in Java (cross-platform). Mason supports discrete-oriented modeling paradigm (released under Academic Free License, version 3.0). Mason includes powerful visualization software (2D, 3D). MASON system is protected from cyber-attacks. However Mason does not support distributed simulation (but currently developers try to create a distributed version). Moreover users have to know programming language Java rather seriously, so Mason is not convenient for the inexperienced users.

- ***Ascape:*** Simulation system Ascape [Ascape, 2015] is another cross-platform agent-based system, written in Java and it is freely available. Users without programming skills may learn many aspects of modeling.

- ***Swarm:*** Swarm [Swarm, 2015] was first program tool for the creation of agent-based applications and was implemented in 1994. Moreover Swarm attempts to create distributed platform.

The requirement for the agent-based simulation systems

So the review being mentioned above suggests that not only users skilled in programming can create the simulation model but the ordinary users skilled in specific domains may do it too. Due to this fact it is advisable to develop special software and visual languages to create and change a simulation model. Moreover it is advisable that agent-based simulation system should have the following characteristics: (a) agent-based simulation system should be cross-platform (the developers of simulation software use Java and C languages in order to achieve this property), (b) agents are developed as the objects sending messages from one to another, (c) users have a possibility to change adjusting parameters of a simulation model during a simulation run [Vlasov, 2013; Zamyatina, 2012].

The authors of this paper discuss the developing of agent-based simulation systems. The authors attempt to take into account the experience of the other developers. Their agent-based simulation system has to meet the following requirements:

- ***Operations with a simulation model.*** It is advisable to have a possibility to change a set of interacting agents and to change links between agents.
- ***Hierarchical representation of simulation model.*** Simulation model should be hierarchical because it is necessary to consider model in details or in contrary to change the group of agents by one agent. This new agent must simulate the behavior of a group of agents [19,20,21].
- ***The highlighting of the structure of collaborating agents.*** It is advisable to develop software for highlighting structure of agent's connections in the simulation model. One can carry out the analysis of highlighting structure of the agents, their relationships, examining its structural characteristics (it is possible to use the methods of graph theory) [Mikov, 1995; Zamyatina & Mikov, 2012].
- ***Data collection and data processing.*** The statistical data collection and data processing must be carried out by special agents that act as sensors, monitors and form "the researching algorithm". This algorithm should be separated from most of the simulation models [Mikov, 1995].
- ***Verification and validation of simulation models.*** Most of simulation systems either do not have the special program components for the verification and validation of simulation models or the functionality of these components is limited. So it is advisable to develop special software for debugging and testing the simulation models in order to obtain a reliable and valid model [Mikov et al, 2013]
- ***Adaptability.*** Simulation systems solve the problems dealt with various specific domains. Moreover simulation model may be described in various «terms». Indeed, let us consider computer networks. One may investigate computer network using the theory of graphs, or the

theory of Petri Nets, or theory of queueing network. So it is advisable to describe computer network as a queuing network or as a graph (a set of nodes and a set of arcs connecting these nodes) or as a Petri Net (bipartite multigraph: a set of nodes (the nodes of two types: positions and transitions) and a set of arcs connecting these nodes (the nodes of different types). It is useful to carry out the description of simulation model using appropriate domain specific language (DSL). The designers of simulation systems apply ontological approach to adjust it to the particular specific domain. One may read the related papers in [Mikov, 2009; Mikov et al.,2009; Sukhov, 2013; Zamyatina et al, 2013].

- ***Reducing the time of the simulation experiment.*** It is well known that simulation systems deal with complicate time consuming problems, so it is necessary to reduce the time of simulation experiment and to use a set of compute nodes of supercomputer or cluster or computer network, so it is useful to provide parallel (distributed) simulation, hence optimistic or conservative algorithms providing causality of events of distributed simulation experiments must be implemented [Fujimoto, 2003]. Load balancing is the second method to reduce the time of distributed simulation experiments [Wilson, 1998].
- ***Safety and fault tolerance of simulation systems.*** Distributed simulation experiment demands additional software. This software provides safety and fault tolerance of a simulation system. It is assumed that software have to find failed compute node and to transfer agents to alive compute node.
- ***Remote access.*** It is advisable to provide the remote access to the simulation system by developing the appropriate Web-services. Remote access will not only allow users to get quickly simulation results and create models using graphical or text editor, but also collaborates users who are geographically located at a remote distance from each other.
- ***Processing of simulation results.*** The user receives a large number of unstructured information as a result of simulation experiment. It is advisable in this case to perform additional processing of the simulation results, using the methods of Data Mining and Knowledge Discovery. Additional processing of the simulation results will optimize simulation experiment, identifying dependencies and relationships between the input parameters [Kolevatov & Zamyatina, 2012].
- ***The intelligent agents.*** In order to reflect more adequately the simulated processes in economics, marketing, logistics it is very important to implement the intelligent agents. Most of the above systems provide the user with the tools to describe the reactive behavior or quite simple one. But it is essential to teach the intelligent agents following the changes in external environments, to pursue goals, to choose a particular strategy depending on the role these agents.

  In this version of the agent-based simulation system the authors focused on the development of tools that support distributed modeling (distributed simulation experiment) and on the development of intelligent agents.

An implementation of agent-based simulation system

The current version of the multi-agent simulation system is made in the language Scala [Odersky, 2015]. Scala language was chosen as the programming language to implement agent-based modeling platform. Scala language is object-functional programming language that combines functional programming and OOP (object-oriented programming) and specializes in building easily scalable component-oriented software. Scala was founded by Martin Odersky in the University EPFL (Lausanne, Switzerland) in 2004. It is currently available for the Java platform and the .NET Framework.

The Scala language includes the following key features: a single object model, the presence of traits, the method of pattern matching, lambda-calculus, type views bounds, type linearization, parametric and functional polymorphism, variation of types, case classes and etc. Moreover there are numerous tools to create new language constructs and list processing. Generalized block diagram of the simulator is shown in Figure 1.

Thus the architecture of the simulator is not a multi-component but multi-layered. This is achieved due to the fact that a module that provides some basic functionality may be "mixed" with other components extending the structure, behavior and semantics.

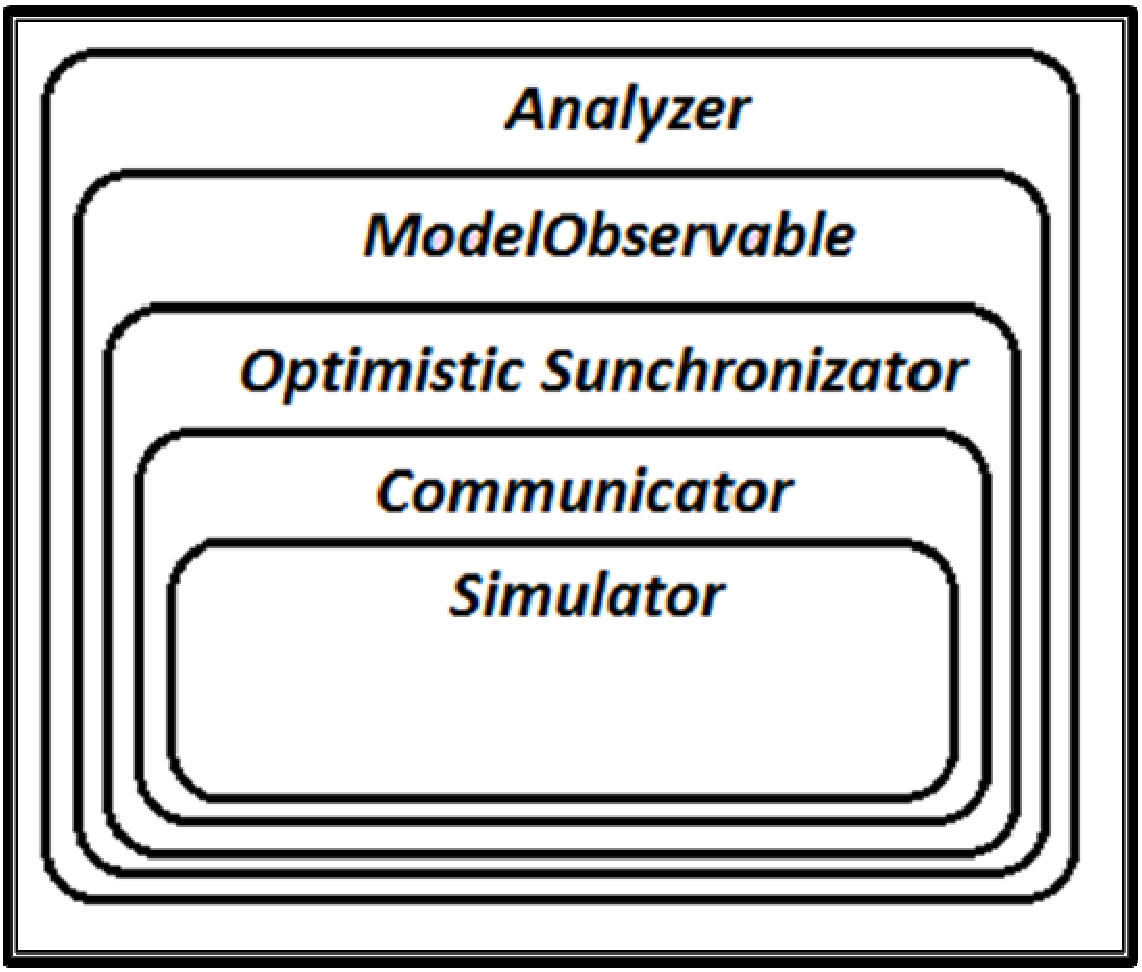


Figure 1. A Multilayer architecture of the simulator

Trait Simulator is the base module, which is interfered with the other elements, specifying it's behavior. Actually, Simulator, - is a frame for a logical process. Traits Communicator is a module for applying the system of actors. It contains a copy of the actor engaged in sending/receiving messages. Traits OptimisticSynchronizator is a key element for the organization of parallel (distributed) discrete-event simulation (PDES). It implements an optimistic synchronization algorithm Time Warp.

Trait ModelObservable is intended to implement "information" procedures – program unit for data collection during a simulation experiment.

Трейт Analyzer is intended to make the synchronization algorithm more "intellectual".

Particular attention is paid to the synchronization algorithm of the developed distributed simulation system. It is known that the distributed model is a collection of logical processes executing on the different computing nodes of high-performance computing system. Logic processes exchange messages among each other. It is important to maintain the causality of events in the distributed systems. The causality of events is supported by classic distributed algorithms. These algorithms are divided into two groups: the conservative and optimistic. Conservative algorithms involve the promotion time only after it becomes apparent that the next event is safe. The event is safe, if you are sure that no other event will be with less "time stamp" [Fujimoto, 2003; Bryant, 1977,Chandy & Misra, 1979]. Optimistic algorithm is as follows: the process is performed by moving from one event to another for as long as it does not receive a message from another process with a smaller time stamp than the time stamp of the next event [Jefferson, 1985].

An optimistic synchronization algorithm based on knowledge of the simulation model is implemented in present simulation system (KBASA). There is a number of works [Zamyatina & Ermakov, 2011], which use knowledge of the simulation model in order to reduce the execution time of distributed simulation experiment.

The authors implemented efficient synchronization algorithm controlled by expert system based on rules in order to reduce the runtime of distributed algorithm. On the other hand the reduction of time is obtained through the load balancing [Wilson, ;Zheng, 2015; Mikov & Zamyatina, 2010].

Classic algorithms also use "knowledge" about the model: *lookahead* in conservative algorithms, *lookback* in the algorithm Work Flow, etc. Using the knowledge of the agent-based simulation model derived from ontologies, have yielded good results in the implementation of synchronization algorithms. Experiments were performed with the same model many times having the same parameters. The total duration of the simulation was 300 (± 15) units of modeling time.

The results of the experiments have shown that the metrics of the algorithm KBASA, based on knowledge of the model are substantially reduced in comparison with the metrics of the optimistic algorithm Time Warp. In fact, (a) the number of rollbacks decreased from 71.2 to 3.8 (Fig. 2). (b) The total number of not deep rollbacks reduced almost to zero (on average 0.1-0.2 vs. 7-11). (c) The number of deep rollbacks also decreased from 19.4 to 3.2. (d) The number of antimessages decreased from 108.4 to 12.8.

The implementation of intelligent agent

It is well known that the creation of intelligent agents is a very complex task that requires a theoretical foundation. There are various models of intelligent agents. These models describe knowledge, the methods of reasoning, the strategy of the behavior and actions of agents in various ways. One may consider these models from two points of view: in terms of the analysis of the properties and behavior of agents during their functioning; from the point of view of the design of the properties of the agents and their internal processes (acquisition of knowledge, goals definition, decision making, etc.).

There are three types of architectures: (a) deliberative architecture and model; (b) reactive architecture and model; (c) hybrid architecture and model.

Reactive approach allows the application of a set of rather simple scenarios. We mean the scenario of the agents behavior. These scenarios are a reaction to the appearance of an event in the external environment.

Deliberative model allows the application of rigorous formal methods and well-proven technologies of traditional artificial intelligence, makes it easy to represent knowledge in a symbolic form and use them in the agent-based systems. The hybrid type of architecture combines the advantages of the architectures mentioned above. Thus, the intelligent agent has a high-level inference engine and low-level reactive abilities.

Thus the development of agent-based simulation system involves the development of a set of base classes to represent the intelligent agents. It was decided to present the architecture of the agents in the form of hybrid schemes and deliberative one.

The inference engine is implemented with the help of production systems and neural networks. It is known that neural networks have the ability to self-learning, which is important for the implementation of intelligent agents, which have to adapt to changes in the external environment and change their behavior in order to make a decision. Three types of neural networks were used: multilayer perceptron, Hopfield network and the Hamming network [Khaikin, 2006; Kruglov, 2002]. A genetic algorithm is proposed for training the neural network.

The developed programming tools were tested. The task of searching of attractions in the park [Zamyatina & Chudinov, 2010; Dubiel & Thimsini, 2005] and "Artificial Life" were chosen in order to test implemented agents based on deliberate and hybrid schemes.

The problem of searching the attractions may be formulated as following: a person is looking for some attractions in the park and trying to get information about his location with the help of maps and informers. Having received information from an informer, a person will go to tram station, rather than solely on foot to get to the sights of interest. When the source data card is used, the person gets to the destination on foot, etc. The problem of "artificial life" is well known.

The results of the test tasks "Searching of attractions" are in the table below.
It has been made 100 test runs of the model. The simulation results were obtained indicating the possibility of using this type of agents (see Table 1).

Agent model based on neural networks was implemented as a 2-layer perceptron having 6 neurons of input level and 2 neurons of output layer. Following input data for the input layer of the network was

presented: (a) the fact that an informer is in sight of person looking for attractions (info); (b) the distance to the informer (when there are not only one informer it is necessary to take into account the distance to the nearest one) (dinf); (c) the distance to the target (df) and etc.

100 test runs were carried out during verification and validation of the agent-based simulation model. The simulation results were obtained indicating the possibility of using this type of agents (see Table 2).

Table 1. The results of simulation experiment “searching the attractions in the park” (agents with production rules”

The Number of agents-informers	Average time to search the target (in seconds)
1	29.6
3	27.9
5	19.5

Table 2. The results of simulation experiment “searching the attractions in the park” (agent - neural network)

The Number of agents-informers	Average time to search the target (in seconds)
1	37.6
3	33.5
5	25.1

Conclusion

So, authors designed and implemented the prototype of an agent-oriented simulation system. Authors tried to follow all the recommendations for the designers of the agent-based simulation systems (cross-platform, hierarchical representation of simulation model, adaptability, operations with simulation model

(add new agent, delete agent, add new links connecting agents), distributed simulation experiment). Developed simulation system is cross-platform because authors used Scala language as a base language.

Particular attention was paid to the development of the synchronization algorithm, which supports distributed simulation. It is known that the distributed model is a collection of logical processes located on different computing nodes. Logic processes must be synchronized to ensure the causality of the events in a distributed model. Authors developed synchronization algorithm based on classical optimistic algorithm. The synchronization algorithm is knowledge-based because it uses the knowledge about simulation model. The investigations have shown that the use of knowledge about the model significantly increases the efficiency of the algorithm. Authors used a model of actors as a formal model of an agent-based simulation system, because it is the most appropriate model showing the behavior of agents in a distributed (parallel) environment.

Another important element in building an agent-based simulation system is the development of intelligent agents. Most agent-based simulation systems do not have the modeling tools that support the functioning of intelligent agents (these system support only the functioning of reactive agent which can't adapt to the changing external environment. The presence of intelligent agents in the simulation system makes it possible to build more appropriate models of real processes and situations.

Intelligent agents which were implemented in agent-based system are based on the production rules and artificial neural networks. Two test models were built (searching of attractions in the park and well known "artificial life") and almost 100 simulation runs were carried out in order to test the functioning of each model. The results of experiments showed the correctness of the developed tools and the decisions taken in their design.

Implementation of works on the design of agent-based simulation system is relevant, because now the question of a wide practical application of simulation expertise [Vlasov et al, 2013] is discussed. It is believed that any design work on the establishment and modernization of any systems (objects) must be preceded by simulation studies "to give practical conclusions and guidelines regarding the appropriateness of existence, building, functioning and modernization of the system."

Bibliography

[Macal & North, 2005] Macal, C. M., and M. J. North. 2005. Tutorial on agent-based modeling and simulation. //Proceedings of the 2005 Winter Simulation Conference. eds. M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines. 2-15. Piscataway, New Jersy: Institute of Electrical and Electronics Engineers

[Macal & North, 2009] Macal, C. M., and M. J. North. 2009. Agent-based modeling and simulation. //Proceedings of the 2009 Winter Simulation Conference. eds. M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, and R.G. Ingalls. 86-98. Piscataway, New Jersy: Institute of Electrical and Electronics Engineers.

[Klyshinskii, 2000] Клышинский Э.С. Проектирование элементов принятия решений в имитационных моделях бизнес-систем. Автореферат на соискание канд. диссертации. [Электронный ресурс] [Режим доступа: http://www.dissercat.com] [Проверено:25.10.2015]

[Ivashkin, 2003] Ивашкин А.Ю. Мультиагентное имитационное моделирование маркетинговых ситуаций. Автореферат на соискание канд. диссертации. [Электронный ресурс] [Режим доступа: http://www.dissercat.com] [Проверено: 25.10.2015]

[Gribanova, 2006] Грибанова Е.Б. Алгоритмы и комплекс программ для решения задач имитационного моделирования объектов прикладной экономики. Автореферат на соискание канд. диссертации. [Электронный ресурс] [Режим доступа: http://www.dissercat.com] [Проверено: 25.10.2015]

[Borshchev, 2015] Борщёв А. От системной динамики и традиционного имитационного моделирования к практическим агентным моделям: причины, технология, инструменты. [Электронный ресурс] [Режим доступа: http://www.gpss.ru/paper/borshevarc.pdf] [Проверено: 25.10.2015]

[Borshchev, 2002] Borshchev A., Karpov Y., Kharitonov V. Distributed Simulation of Hybrid Systems with AnyLogic and HLA. // Parallel computing technologies. – 2002. № 18(6). – P.829-839.

[AnyLogic, 2015] Система моделирования “AnyLogic” [Электронный ресурс] [Режим доступа: http://www.xjtek.ru] [Проверено: 25.10.2015]

[Repast, 2015] Система моделирования “Repast” [Электронный ресурс] [Режим доступа: http://repast.sourceforge.net] [Проверено:25.04.2014]

[NetLogo, 2015] Система моделирования “NetLogo” [Электронный ресурс] [Проверено:25.04.2014]

[Mason, 2015] Система моделирования “MASON” [Электронный ресурс] [Режим доступа: http://cs.gmu.edu/~eclab/projects/mason] [Проверено:25.04.2014]

[Ascope, 2015] Система моделирования “Ascape” [Электронный ресурс] [Режим доступа: http://ascape.sourceforge.net/index.html#Contact] [Проверено:25.04.2014]

[Swarm, 2015] Система моделирования SWARM. [Электронный ресурс][режим доступа:www.swarm.org] [Проверено:25.04.2014]

[Vlasov et al, 2013] Власов С.А., Девятков В.В., Назмеев М.М. Имитационная экспертиза: опыт применения и перспективы // Сборник докладов шестой всероссийской научно-практической конференции «Имитационное моделирование. Теория и практика» (ИММОД-2013). Том 1. // ISBN 978-5-9690-0221-0 // Издательство «ФЭН» Академии наук РТ, Казань, 2013, с. 54-63.

[Zamyatina & Mikov, 2012] Замятина Е.Б., Миков А.И. Программные средства системы имитации Triad.Net для обеспечения ее адаптируемости и открытости. Информатизация и связь. №5, 2012, АНО «Редакция журнала «Информатизация и связь», ISSN 2078-8320, С.130-133.

[Mikov, 1995] Mikov A.I. Simulation and Design of Hardware and Software with Triad// Proc.2nd Intl.Conf. on Electronic Hardware Description Languages. Las Vegas, USA, 1995. P. 15 20.

[Mikov et al, 2013] Mikov A.,Zamyatina E.,Mikheev. Linguistic and Program Tools For Debugging and Testing Of Simulation Models Of Computer Networks. International Journal “Information Models and Analyses, V.2., N 1,2013, ITHEA, ISSN 1314-6416, Sofia., 1000., P.O.B. 775, Bulgaria, pp. 70-80

[Mikov et al, 2009] Mikov A., Zamyatina E., Kubrak E. An Ontology-based Approach to the Incomplete Simulation Model Analisis and its Automatic Completion. International Journal “Information Technologies & Knowledge”, 2009, Volume 3, Number 2, pp. 169-186.

[Sukhov, 2013] Сухов А.О. Трансформация визуальных моделей в системе MetaLanguage / Современные проблемы математики и ее прикладные аспекты – 2013. Сборник тезисов конференции. – Пермь: Пермский государственный национальный исследовательский университет, 2013. – С. 44.

[Zamyatina et al, 2013] Замятина Е.Б, Лядова Л.Н., Сухов А.О.. Мультиязыковое моделирование с использованием DSM платформы MetaLanguage. Информатизация и связь. №5, 2013, АНО «Редакция журнала «Информатизация и связь», ISSN 2078-8320, С.11-15.

[**Fujimoto, 2003]** Fujimoto R.M. Distributed Simulation Systems. In Proceedings of the 2003 Winter Simulation Conference S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, eds. The 2003 Winter Simulation Conference 7-10 December 2003. The Fairmont New Orleans, New Orleans, LA, pp. 124-134

[Bryant, 1977] Bryant R.E. Simulation of packet communication architecture computer systems. Technical Report MIT-LCS-TR-188, Massachusetts Institute of Technology, 1977.

[Chandy & Misra, 1979] Chandy K.M, Misra J. Distributed simulation: A case study in design and verification of distributed programs. IEEE Transactions on Software Engineering, SE-5(5): p.440-452, September, 1979.

[Jefferson, 1985] Jefferson D.R. Virtual Time. ACM Transactions on Programming Languages and Systems, 1985. 7(3): p.404-425.

[Wilson, 1998] Wilson L. F., Shen W. Experiments in load migration and dynamic load balancing in Speedes // Proc. of the Winter simulation conf. /Ed. by D. J. Medeiros, E. F.Watson, J. S. Carson, M. S. Manivannan. Piscataway (New Jersey): Inst. of Electric. and Electron. Engrs, 1998. P. 487–490.

[Zheng, 2005] Zheng G. Achieving high performance on extremely large parallel machines: Performance prediction and load balancing: Ph.D. Thesis. Department Comput. Sci., Univ. of Illinois at Urbana-Champaign, 2005. 165 p. [Electron. resource]. http://charm.cs.uiuc.edu / [Дата обращения: 24.10.2015] .

[Mikov & Zamyatina, 2010] Миков А.И., Замятина Е.Б. Проблемы повышения эффективности и гибкости систем имитационного моделирования. Проблемы информатики,№4(8), Новосибирск, Институт вычислительной математики и математической геофизики СО РАН, 2010, стр.49-64

[Kolevatov & Zamyatina, 2012] Kolevatov G.A., Zamyatina E.B. Simulation Analysis Framework Based on Triad.Net. Proceedings of the 6-th Spring/Summer Young Reseachers' Colloquium on Software Engineering. SYRCoSE 2012, Perm, May 30-31, 2012-Perm, Russia,pp.160-163.

[Odersky, 2015] Odersky M. The Scala Programming Language [Электронный ресурс]. URL: http://www.scala-lang.org/node/25 [Дата обращения: 24.10.2015]

[Zamyatina & Ermakov, 2011] Замятина Е., Ермаков С. Алгоритм синхронизации объектов распределенной имитационной модели в TRIAD.Net. Applicable Information Models. ITHEA, Sofia, Bulgaria, 2011, ISBN: 978-954-16-0050-4, стр.211-220

[**Mikov & Zamyatina, 2012]** Миков А.И., Замятина Е.Б. Использование GPU для повышения производительности симулятора компьютерных сетей. Высокопроизводительные вычисления на графических прцессорах: тезисы докл. Науч.-практ.конф. с междунар. Участием с элементами науч. шк. для молодежи, 21-25 мая 2012 г. Перм. гос. нац. иссл. ун-т..-Пермь, 2012, стр.53-57.

[Zamyatina & Chudinov, 2010] Замятина Е.Б., Чудинов Г.В. Разработка и использование программных средств для построения и исследования агентных имитационных моделей. / Е.Б. Замятина, Г.В. Чудинов // Вестник пермского университета. Математика, механика, информатика, 2010, №2(2), С. 80 – 84.

[Dubiel & Thimsoni, 2005] Dubiel B., Tsimhoni O. Integrating agent based modelling into a discrete event simulation . In the Proceedings of the 2005 Winter Simulation Conference, ed. Kuhl M. E., Steiger N. M., Armstrong F. B., and Joines J. A., 2005, pp. 1029 1037. URL: http://www.informs-cs.org/wsc05papers/123.pdf. [Проверено:25.04.2014]

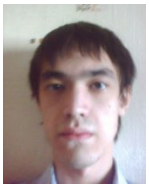
[Khaikin, 2006] Хайкин С. Нейронные сети: Полный курс. – Москва: Вильямс, 2006. – 31-89 с.

[Kruglov, 2002] Круглов В.В. Искусственные нейронные сети. – Москва: Телеком, 2002. – 63-79 с.

Authors' Information

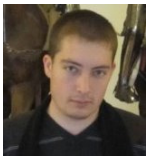
Elena Zamyatina *– Assoc. Prof. PhD, National Research University Higher School of Economics, Department of Business Informatics; Russia, Perm, 614070, Studencheskaya st., 38;e-mail: e_zamyatina@mail.ru*

Fields of Scientific Research: simulation, artificial intelligence, ontology, domain specific languages (DSL)



Danil Karimov – *Programmer, "PROGNOZ", Stakhanovskaya St, 54, Perm, e-mail: googlicus@gmail.com*

Fields of Scientific Research: information systems, neural networks, Web-programming



Mitrakov Artem Andreevich, - *Lecturer, Perm National Research University (PGNIU), Bukireva,15, e-mail: mitrar@yandex.ru*

Fields of Scientific Research: information systems, agent-based simulation, ontology