Development of a Multi-Agent System for Describing the Logic of Behavior of Mobile Robots

Evgeniy G. Sysoletin
Ural Federal University
Ekaterinburg, Russia
unclesal@mail.ru

Konstatnin A. Aksyonov
Ural Federal University
Ekaterinburg, Russia
wiper99@mail.ru

Xenia D. Nisskhen
Ural Federal University
Ekaterinburg, Russia
xenia.nisskhen@ya.ru

Olga P. Aksyonova
Ural Federal University
Ekaterinburg, Russia
bpsim.dss@gmail.com

Abstract

This article provides a comparative analysis of CASE-tools, and also describes the rationale for architectural and technical solutions that formed the basis for the development of a universal CASE-tool for describing the behavior of mobile robots. The toolkit, which is being developed, is based on multi-agent technology and is intended for conducting various experiments in the field of artificial intelligence. As the basis for the development of the idea of combining within the framework of one, tool both real execution of the algorithm by the robot, and its modeling. This allows us to talk about testing partially implemented hardware. The development is carried out based on open source technology; all the texts of the programs are available at https://github.com/unclesal/tenguai.

1 Introduction

It is probably difficult to overestimate the relevance of the subjects of robots in general and mobile robots in particular. Robots perform new functions and become more intelligent and skillful. It is the consequence of the complexity both hardware and software-algorithmic part. The increasing complexity increases the time for program development and the cost of their maintenance. In addition, mobile robots actively cooperate with the environment. Accordingly, in addition to directly achieving the goals that set for the robot the software must take into account the aspects of interaction between robots, the sharing of common resources of the surrounding space, security and functioning in the same space as people, cars, buildings, etc.

The multi-agent approach to programming such systems has proven itself quite well. With this approach, both each of the robots, and the components of the surrounding space is a relatively independent, functionally complete agent. Moreover, the "agent" is not necessarily monolithic; in its turn, it can consist of other agents, the specific set of which is determined by the current state and can change over time. It can be noted that this approach makes it possible in the future to realize the concept of computational consciousness according to S. A. Pinekr [1]. This approach ensures reliability, because failure in the work of one of the agents is not fatal, does not lead to the inoperability of the system as a whole. In addition, the task of software development as a whole is somewhat simplified: it is simpler to write, debug, and accompany several smaller, functionally completed tasks than one large program that takes into account all the options for possible development of the situation.

The CASE-tools, in which the description of the logic of behavior of complex systems is made on a clear and intuitive level, in the form of various diagrams. However, the diagrams themselves are only a form of representation, they are not able to be performed. Accordingly, the question is concretized before the creation of a certain "engine" that allows you to translate the diagrams into machine codes that are executed on the mobile robot's board. Accordingly, the question is concretized before creating a certain "engine" that allows to translate the diagrams into machine codes on-board the mobile robot. Such "engines" certainly exist [2-4]. However, for a number of reasons, existing CASE-tools are intended, as a rule,

to work as systems of simulation modeling, but not for programming real robots. The system is one of all third-party experts working with abstract components that reflect reality in one way or another. At the same time, the system is in a sense abstract, external, has no direct connection with the sensors, just as it has no direct effect on the actuators. The very interaction with reality lies outside the system and is often carried out by a person - taking into account the proposed solutions offered by the system.

The following is a comparison of software product ARIS ToolSet, PowerDesigner, BorlandTogetherDesigner, IBM Rational, CA ERwinModelingSuite, BizAgi and Elma in the design (table 1.1). They are most common among users of software systems.

Table 1.1– Comparison of CASE-tools

Comparison criteria	IBM Rational tools	CA ERwinModelingSuite	ARIS ToolSet	PowerDesigner	BorlandTogetherDesigner	BizAgi	Elma
Support IDEF0, DFD	-	+	+	-	-	-	-
Support UML	+	+	+	+	+	-	-
Support BPMN	-	-	+	-	+	+	+
IM BP	-	-	+	-	-	+	-
Software design based on IM BP	-	-	-	-	-	+	+
Development of a model agent	-	-	-	-	-	+	+
Script description the decision-making agent	-	-	-	-	-	+	-
Design UI	-	-	-	-	-	+	-
Conversion charts	-	-	-	-	-	-	-
Designing of DB structure	+	+	+	+	+	-	-
Generation of technical documentation to be created the IS	+	+	+	+	+	+	-

In this paper, the method for developing information systems (IS) for the subject area of multi-agent resource conversion processes (IPM) [12-13] is used as a basis and programmed in the BPsim complex, which consists of the following stages (Fig 1):

1. The development of IP begins with a survey of the subject area and the construction of an imitative model of the MAPP "as is."
2. Conducting simulation experiments with the "as is" model in order to identify "bottlenecks" in the organization of processes. Based on the results of the simulation, the model of the MAPP "will be constructed" as it will be.
3. At the third stage, the IP model is constructed on the basis of data from the MAPP model.
   3.1. The objects of the model of MDPR are transformed into objects of diagrams DFD, Use-case, Classes diagrams.
   3.2. Objects of the diagram Classes diagram are transformed into objects Sequence diagrams and elements of the structures of the DB and BS.
   3.3. Objects in the Sequence diagram are transformed into GUI elements.
4. Solution of the issue of placing instances of domain concepts on the knowledge bases of agents (depending on technical requirements).
5. Finalization of the system by developers, construction of sequence diagrams and modeling of the user interface.

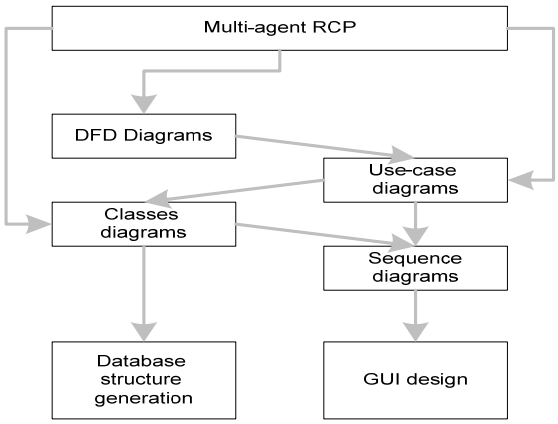

Figure 1. Basic stages of the design method.

In this case, a slightly different approach is proposed. Toolkit, the frame, the "engine" itself - he is one. Both for modeling and for real execution of algorithms. In a real implementation on board the mobile robot may not be specific parts of the engine, for example, GUI. However, the base it remains the same and still interacts with the rest of the system. As a result, in the process of real implementation, mobile robots can be co-existed between them - both physically realized and simulated, implemented only from a mathematical point of view. All the same applies to the composition of the robot itself: certain robot nodes can already be realized, while others exist only in the form of a mathematical model, but this circumstance is not an obstacle to launching the entire robot entirely.

Development is conducted with reference to the field of unmanned aerial vehicles, but in general, the system itself can be applied to any mobile robots.

2 Inter-agency data exchange

In the multi-agent approach to modeling and programming [14-16], the main component of the system is the mechanism for the exchange of certain data between participants (agents). The data itself can be divided into two groups:

- Messages. In other words, notifications of events that have occurred. Messages can be addressed, that is, they go from one agent to another. In addition, there may be thematic messages. In this case, the sender is one, and the recipients may be several agents "interested" in this topic. Finally, messages can be broadcast. For such messages, the term "message board" is often used.
- Conditional-constant data. In fact, this is a set of certain parameters. Sometimes it is also called "environment variables".

Since the nature of the data is different, often the sources of these data are also made differently. This approach is implemented, for example, in ROS [5]. To start any ROS process, two servers must be running: ROS-master and Parameter Server.

However, often these two groups of data overlap. That is, changing a parameter is simultaneously an event in response to which a certain reaction from already running agents can or must occur. The next agent that launches takes into consideration the parameter, because the change and the corresponding event occurred before the agent started.

Another problem that needs to be considered is the mechanism for processing events from the operating system. The occurred event is somehow processed, the registered handlers of the given event are called. The ambiguity of the situation is that there is no single standard for handling events. Moreover, in the operating system at the same time, there may well be several different event processing cycles that are not connected in any way.

In view of the above factors, the LoRedis event mechanism was developed based on the in-memory repository Redis.io (https://redis.io/). Redis.io allows you to capture both the event-driven nature of the data and the conditionally permanent data. However, in the "boxed" version, Redis.io only provides a C-based API. LoRedis is a C ++ wrapper for calling the Redis.io functions at a higher level. In addition, LoRedis allows asynchronous processing of events in two ways: through the libevent library and through the QtCoreApplication event loop. Thus, LoRedis provides a more flexible way of working with the in-memory data storage and event provider.

Separately, it is necessary to touch on the topic of safety and reliability of the system. As already mentioned, in the general case, the multi-agent system is built on the principles of decentralization and distribution. It would be inadvisable to reduce the processing of all events to one node because:

- Such an approach would lead to a single and central component of the whole system, which is contrary to the principle of decentralization.
- This build reduces the security of the system, since any external element must at least have the ability to connect to the central event-processing node. At the same time, it is far from obvious that the client that is picked up will behave correctly.
- It is not a fact that this particular mobile robot can connect to the central event-processing node at any time. The robot is able to move in space, so situations are not excluded when there is simply no communication between the board and the central event node.

One of the possible ways to resolve this situation is to introduce two additional agents into the system. The task of one of them is the transportation of events between agents; the second is for notification, broadcast messages about the presence of this robot and the possibility of establishing a connection with it (Fig. 2).

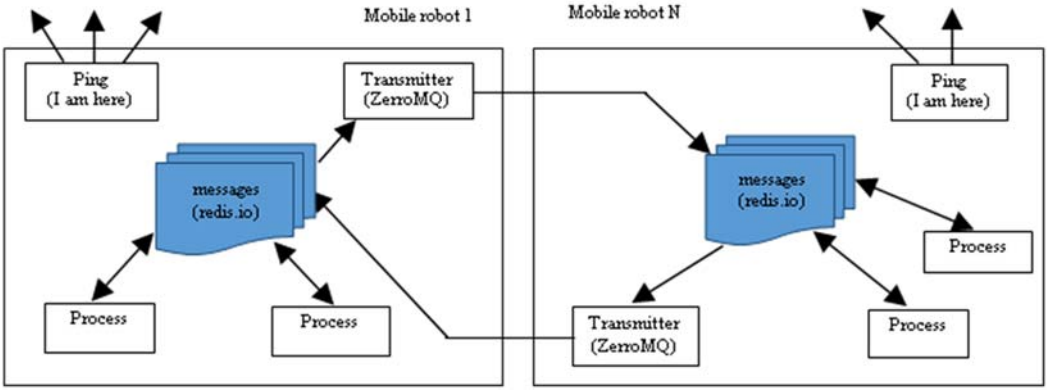

Figure 2. Organization of communication between the agents entering into the system.

With such a construction scheme, each of the robots has its own event processing mechanism, located directly on its board. Exchange between the boards is based on the use of the technology of queues (RabbitMQ, ZerroMQ, etc.). However, the recipient of the remote message is not a specific agent, but again the LoRedisbased event mechanism. In other words, a remote message is "published" in a local event mechanism - after a point-to-point connection has been established between the robots. As a result of this approach:

- The structure of the mobile robot does not change over time; it does not depend on the number of participants in the system. There is no difference between messages from local and remote agents; they are all handled in exactly the same way.
- There was an opportunity of authentication. The robot "gives or does not give its consent" to communicate with a certain external source of messages. What improves work safety, unknown third party components cannot be the source of events for this robot.

3 Simulation

Simulation in the field of robotics is a necessary thing. Playing the situation allows you to debug hardware and software components without conducting "full-scale experiments", which ultimately reduces the cost of development. Of particular relevance is the simulation in the field of aircraft, since a collision with the planet, as a rule, leads to fatal consequences. Thus, modeling is understood in a natural way, as a necessary part of the system.

Proceeding from the above construction principles, the conclusion that the simulation process should be an agent interacting organically with the already existing mechanism of events logically arises. Naturally, such agents do not exist in the finished form. Therefore, there is a need for their adaptation by writing some additional software products.

Since the subject area was chosen unmanned aerial vehicles, the simulator should be aimed precisely at them. There are many open and commercial simulators: Lockheed Martin Preparation3D, Microsoft Flight Simulator, FlightGear Flight

Simulator, RealFlight, MicroFlight, FMS, Atmosphere Velocity and others. As a result of the comparative analysis of existing simulators, the choice fell on X-Plane, because:

- It has an assembly including Linux too. The development of the described CASE-system of robots is carried out in the Linux operating system; it is also used on most boards. Having a simulator in the same operating system is not a prerequisite, but it is convenient.
- It is essentially open. Provides a plug-in mechanism that allows you to access virtually any internal variables of the simulator.
- It provides, perhaps, the most elaborated mathematical model of flight. Here it is worth mentioning that this simulator received the status of "game" only from version 5, published in 2001. For 9 years before that, he was - precisely a specialized flight simulator for the Piper Archer. Such a long term of development causes a sufficiently high proximity of simulation and real flight. From the program point of view, for the reading, and often for the change, during the simulation, the internal variables of this mathematical model are available, for example, the arising forces and moments.
- It is used for modeling by commercial firms developing real aircraft. As an example of such use, the company CarterCopter can be named. At least in 2013-2015 was used by NASA to visualize the behavior of their own models, later usage references were not found. Used by the US Agency for Advanced Development DARPA. He has a certificate from the US Federal Aviation Administration (FAA), which recognizes the right to train and train civilian pilots on its basis.
- It allows completely to disconnect the internal logic of control of the aircraft, stabilization system, autopilot, etc. At the same time, the control of the aircraft in the simulator is completely transferred to the side of the debugged external (in relation to the simulator) program. In addition to regular behavior, this circumstance makes it possible to model the occurrence of malfunctions in the hardware.
- It takes into account this environment (wind, clouds, thermal flows, etc.).
- It has a special graphical toolkit (PlaneMaker) for creating arbitrary models of aircraft with sufficiently wide ranges of weighted characteristics (Fig. 3). In other words, it is possible to create an aircraft similar in its characteristics to the actual UAV being developed. Thus, achieving the maximum possible adequacy of the simulation process.

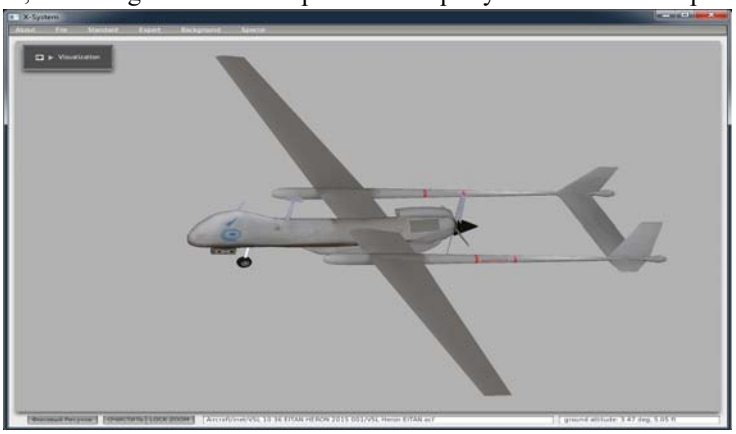

Figure 3. Screen Utilities PlaneMaker simulator X-Plane

For other subject areas, in another simulator will have been needed. Perhaps it could become Gazebo (http://gazebosim.org/). However, the approach to the organization of communication with the simulator does not depend on the specifics and can be used with any external programs. When using another simulator, it is necessary to implement additionally only one block, which is responsible for transferring events from a particular simulator to the local LoRedis and back. In other words, the adapter for the desired simulator will have been needed to write.

At the core of the software kernel communication with both the simulator and the real hardware is the idea of the so-called Sprout, an abstraction for representing an external device (sensor or actuator). The developed algorithm of the robot does not interact with the hardware directly. Instead, he refers to Sprout, which, depending on the situation, can be connected either to a real device or to a mathematical model of a real device (Fig. 4). Combinations in one process between "real" and "virtual" can be any, depending on the specific needs and interests of developers.

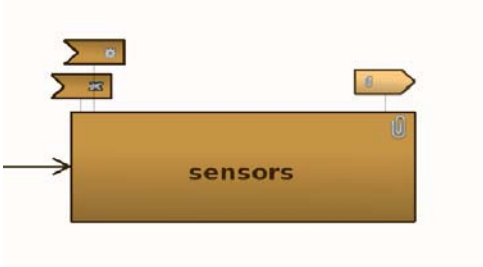

Figure 4. Using Sprout.

4 RAD technology and programming language

The Rapid Application Development (RAD) technology has existed for a long time. In fact, from the whole approach of RAD, two factors can be identified, which over time began to play a key role in the "rapid" creation of applications:

• The object-oriented graphical interface that allows you to work with the created software, including the non-programmer, but simply a specialist in the subject area. In the case of the universality of the toolkit, the term "specialization in the domain" is blurred, expanding to the usual common sense. The interface should be such that the algorithm of the program is understandable to the widest possible range of users. Diagrams for such purposes fit as well as possible.

• Partitioning into separate blocks, "bricks", with the possibility of as wide a reuse as possible, is already assembling blocks. Ideally - just as purely a visual way, without having to write additional code. If you have something like "programming", then you can do it, if you want, the storage of such blocks should be open, allowing the creation of custom components.

With the classical approach to software development, talking about a certain rigidly defined algorithm of behavior. There was an obstacle on the left - the robot should be turned to the right side. There was quite a lot of effort on the sensor - more likely, the manipulator encountered an unplanned obstacle (possibly with a human) and further manipulator movement should be stopped. There can be very many conditions, that is, pre-envisaged situations, but in the final analysis, the combination of conditions uniquely determines the logic of the behavior of the mobile robot. There is a set of external signs, "incentives" available to the robot through sensors and determining the situation that has arisen. This set corresponds to the reaction of the robot. For all the seeming simplicity of the approach, the resultant behavior of the robot may prove to be sufficiently "reasonable", since the number of stipulated conditions is significant. If it is add, here some choice of one of the possible reactions envisaged for this combination of stimuli and based, let us assume, on certain priorities, on the "degree of desirability" of this outcome of events, then result will be to get one of the practical realizations of behaviorism. That is, the resulting behavior of the robot is rather complicated. Such a robot could already be fully called "smart", that is, capable of acting according to the circumstances.

This concept combines well with RAD and is implemented in some CASE-systems. "Programming" is to write conditions of the form IF ... THEN ... ELSE ... with possible indication of the priority of these conditions and, possibly, the calculation of the values of some elementary functions.

For writing the GUI, the Qt library was initially used (https://www.qt.io/). Therefore, it would be quite reasonable not to complicate the system by using an external interpreter, but use Qt's QML (Qt Modeling Language) as part of Qt. Assuming that the main algorithms will still be implemented in C / C ++, the role of QML actually amounts to describing non-standard reactions to events and implementing small user procedures. The using QML has several advantages:

- Language allows the use of alphabetic characters (via UTF encoding). In other words, during the programming process, the user can use the procedures and functions that are native to him.
- There is a close link between QML and Qt itself, since QML is an integral part of the Qt library. It is not difficult to gain access from QML to variables and methods implemented in C ++, and vice versa. The user accesses from the modeling language to almost any structure as the elements (tasks) of the individual process, and the core of the system as a whole.

In this case, there is no need to insert QML into each block of the system, it is enough to talk about one universal block "task", which is used in cases when the logic of available ready blocks is not enough (Fig. 5). The remaining blocks are configured in the usual way, by changing their properties (parameters).

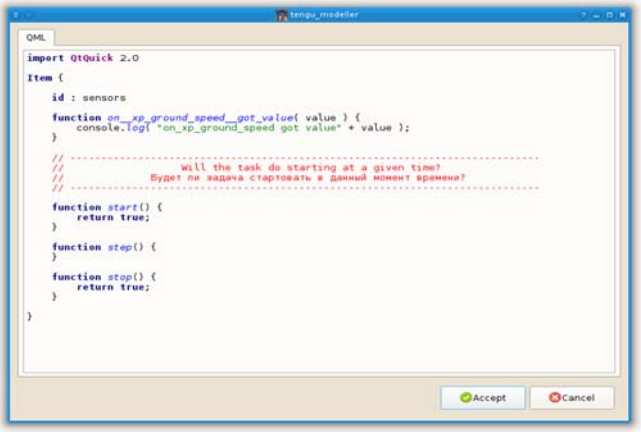

Figure 5. Extending the task algorithm using QML.

5 Branching of the algorithm and the concept of the "focus of the problem"

The most common branching representation on the diagrams is shown in Figure 6. It is intuitive, if some condition is satisfied, then the algorithm continues to execute one branch. If the condition is not met, then the other. The very same task (block) is usually considered in two states: it can be in the execution state, or inactive.

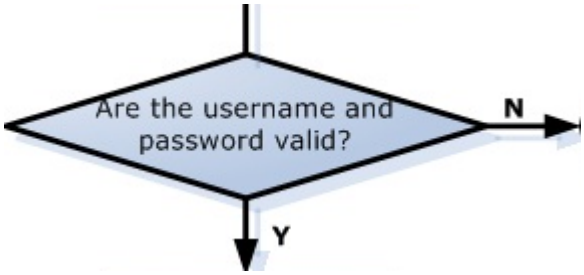

Figure 6. The classical representation of algorithm branching.

But in practice this approach is clearly not enough. Several robotic flows can be performed simultaneously on the robot, each of which generally has its own start and stop conditions. And these streams are completed - not necessarily synchronously at the same time. Accordingly, with the continuation of the algorithm, the question arises whether it is necessary to expect the completion of all the running flow of the previous stage, or it is sufficient to verify the completion of some "key" task.

In order to achieve the greatest universality, a third state, the "focus," was introduced into the problem. The task can be in the running state, in the standby state (no execution) and in the "focus" state (readiness for execution). In the latter version, the task is ready to start, but before running it pre-checks the execution of some of its own conditions. If they are executed, the "focus" from this task is removed and the task goes into the execution state, if not, it remains in the "focus" state until this state is removed from the outside, or until a certain timer occurs. All changes in the status of the task cause signals that inform other agents of the occurrence of the event.

To ensure the branching of the algorithm, two blocks-ANDor and ORer - are introduced into the system (Fig. 7). Both of them can have an arbitrary number of incoming and outgoing connections. The triggering logic is present in the name, ANDor implements the logical AND, ORer-the logical OR. The logic is applied to the input and output separately. That is, in the case of ANDor'a, in order for this node to become active, it is necessary to have a signal on all its inputs. In this case, the fact of activity will be transmitted simultaneously to all its outputs and their activity will not change during the operation of the algorithm. Thus, all the blocks shown on the diagram after ANDor'a will be executed in parallel. In the case of ORer, there is enough signal to activate the node on at least one of its inputs. The focus is transferred to all outputs, but launching

at least one outgoing task results in the focus being removed from all other outputs. In other words, ORer'u enough to run at least one outgoing task, of all the presented parallel outputs will work only one branch of the algorithm.

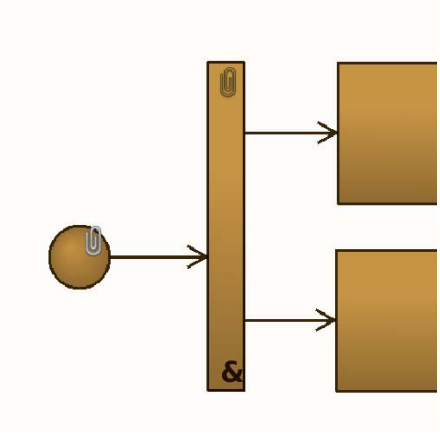

Figure 7. ANDor block: organization of parallel execution of tasks.

This approach differs from the one adopted in the standard schemes, but at the same time it allows to provide maximum flexibility of the graphical representation of the algorithm.

6 Conclusion

The article considers the main technical solutions underlying the development of the CASE toolkit for describing the behavior of a mobile robot (or a group of mobile robots). The article considers the main technical solutions underlying the development of the CASE toolkit for describing the behavior of a mobile robot (or a group of mobile robots). This toolkit combines both modeling and algorithm work on a real mobile robot. Development is carried out in stages, with the linking of the results of each stage to the hardware implementation. The purpose of the development is to provide the possibility of conducting experiments in the field of artificial intelligence of autonomous devices, in particular, the introduction of "feelings" that affect the motivation for the further behavior of the mobile robot.is toolkit combines both modeling and algorithm work on a real mobile robot. Development is carried out in stages, with the linking of the results of each stage to the hardware implementation. The purpose of the development is to provide the possibility of conducting experiments in the field of artificial intelligence of autonomous devices, in particular, the introduction of "feelings" that affect the motivation for the further behavior of the mobile robot.

Acknowledgements

The work was supported by Act 211 Government of the Russian Federation, contract № 02.A03.21.0006.

References

1. Steven Pinker. How the Mind Works. – W. W. Norton & Company, 1997 – 660 p.
2. Aksyonov K., Bykov E., Sysoletin E., Aksyonova O., Nevolina A. Integration of the Real-time Simulation Systems with the Automated Control System of an Enterprise // International Conference on Social Science, Management and Economics (SSME 2015) Guangzhou, China, May 09-10, 2015, P. 871-875. WOS:000361112900160.
3. Aksyonov K., Bykov E., Sysoletin E., Aksyonova O., Goncharova N. Perspectives of Modeling in Metallurgical Production // International Conference on Social Science, Management and Economics (SSME 2015) Guangzhou, China, May 09-10, 2015, P. 876-880. WOS:000361112900161.
4. Aksyonov, K., Bykov, E., Aksyonova, O., Nevolina, A., Goncharova, N.Architecture of the Multi-agent Resource Conversion Processes Extended with Agent Coalitions. IEEE International Symposium on Robotics and Intelligent Sensors, IRIS 2016; Hosei UniversityTokyo; Japan; 17 December 2016 through 20 December 2016; Code 134518. Procedia Computer Science 105, pp. 221-226. DOI: 10.1016/j.procs.2017.01.214 WOS:000398830900036.

5. Lentin Joseph. Mastering ROS for Robotics Programming: design, build and simulate complex robots using Robot Operating System and master its out-of-the-box functionalities. Packt Publishing Ltd. Birmingham – Mumbai, 2015 – 481 p.
6. Bruce Williams. Scenario-Based Training with X-Plane and Microsoft Flight Simulator: Using PC-Based Flight Simulations Based on FAA-Industry Training Standards – John Wiley & Sons, January 2012 - 624 p.
7. Jeff Van West, Kevin Lane-Cummings. Microsoft Flight Simulator X For Pilots: Real World Training - John Wiley & Sons, July 2007 – 744p.
8. David Ohrvall, Crack the Case System: Complete Case Interview Prep - Turtle Hare Media, October 2011 - 548 p/
9. James Martin, Rapid Application Development - Macmillan Publishing Company, 1991 – 788 p.
10. Marianne A. Silva, Quality Toolkit and Case Studies: Solutions for Common Problems - AABB Press, October 2011 – 192 p.
11. Koray Okar, Evaluation and selection of case tools - LAP Lambert Academic Publishing, 2010 – 224 p.
12. Aksyonov K. A., Spitsina I. A., Sysoletin E. G., Aksyonova O. P., Smoliy E. F. Multi-agent approach for the metallurgical enterprise information system development // 24th Int. Crimean Conference “Microwave & Telecommunication Technology” (CriMiCo’2014). 7—13 September, Sevastopol. Vol. 1. P.437-438. DOI: 10.1109/CRMICO.2014.6959467
13. Aksyonov K.A., Bykov E.A., Aksyonova O.P. Development and application of software engineering solution BPsim.SD // Proceedings - UKSim-AMSS 7th European Modelling Symposium on Computer Modelling and Simulation, EMS 2013. Manchester; United Kingdom; 20 November 2013 through 22 November 2013. Article number 6779866, Pages 321-325. WOS:000350449700054, DOI: 10.1109/EMS.2013.55
14. Kuleshov, S, V., Aksenov, A. J., Zaytseva A. A.: Software-Defined Data Formats in Telecommunication Systems. Series: Advances in Intelligent Systems and Computing. 575 , 326–330 (2017)
15. Alexandrov, V. V., Kuleshov, S. V., Zaytseva, A. A.: Active Data in Digital Software Defined Systems Based on SEMS Structures. Logical Analysis of Data and Knowledge with Uncertainties in SEMS – Smart Electromechanical Systems, Studies in Systems, Decision and Control. Gorodetskiy, A. E. (ed.). 49, 61–69 (2016)
16. Borodin, A., Kiselev, Y., Mirvoda, S., Porshnev, S.: On design of domain-specific query language for the metallurgical industry. In: Proceedings of 11th International Conference BDAS: Beyond Databases, Architectures and Structures: Communications in Computer and Information Science, pp. 505–515 (2015)