CIMSA 2006 – IEEE International Conference on
Computational Intelligence for Measurement Systems and Applications
La Coruna - Spain, 12-14 July 2006

Modeling Technology for One Class of Multi-Agent Systems with Automata Based Programming

Dmitry Paraschenko, Anatoly Shalyto, Fedor Tsarev
St. Petersburg State University of Information Technologies,
Mechanics and Optics
Computer Technologies Department
Sablinskaya street 14, St. Petersburg, Russia
{parashchenko,tsarev}@rain.ifmo.ru, shalyto@mail.ifmo.ru

Abstract *– The technology of modeling of one class of multi-agent systems with automata based programming is proposed in the paper. The technology is illustrated on the example of system of interacting drone flying objects creation. UniMod – a tool intended for supporting automata based programming is used for building a control system for each flying object. UniMod also supports the concept of "executable UML".*

Keywords *– automata based programming, UniMod, reactive agents, hybrid agents, multi-agent systems, SWITCH-technology*

I. INTRODUCTION

Reflective agents [1] were studied by psychologists [2], who used "input-output" representation. This representation corresponds to the model of automata without memory – combinational circuit.

As the result of progress in psychology, comprehension of need to take agent's internal state into consideration [3] aroused. In the majority of papers model of reflective agents with internal states (reactive agents) was supposed to be too simple to be a foundation of a serious system. Researches described in [4, 5] cast doubt on this statement [1].

The usage of this approach allows avoiding creating the complex model of environment – instead of it one can use reactive system (system, which reacts on external events), description of which is described by finite automata.

It is important to note, that *NASA* uses the similar approach [6] in creation of software from project *Mars Science Laboratory*.

Later on, the automata based programming was proposed [7], and researches on its application to implementation of different types of agents:

- logical control agents [8, 9];
- reactive agents [10];
- object-oriented reactive agents [11]

were made.

Automata based programming technology was verified in the context of *Foundation for Open Project Documentation* [12]. Several student projects in which multi-agent systems were created were made. An example of creation of reactive system with complicated behavior is oriented agents' implementation are described in papers [13–17]. In those projects agents' implementation was made manually – without any programming tools.

Publication of project documentation is the necessary requirement in the context of *Foundation for Open Project Documentation*. Project documentation should be designed in such way, that any programmer could understand and modify the program, and the modification will be made easier than in the case program is designed and documented in traditional way. In the article [6] need of detailed project documentation for software creation is mentioned too.

This work was made in the context of *Foundation for Open Project Documentation*. As a consequence, program described is supplied with detailed project documentation. It is available for download on site http://is.ifmo.ru in *UniMod-projects* section.

The goal of this paper is to describe the technology of modeling of one class of multi-agent systems on the basis of automata based programming. This technology is based on using *UniMod* tool [18, 19] which is intended for supporting automata based programming technology, also called *SWITCH*-technology [8].

II. AUTOMATA BASED PROGRAMMING

In the context of automata based programming it is recommended to build programs like automated systems, which consist of control system (system of cooperating automata), controlled objects and feedback loops between objects and control system. Automata transits between states using input actions (events and input variables) and forms output actions which correspond to controlled objects' methods. Such a view on programming is natural while solving different control problems including several types of multi-agent systems.

Two types of diagrams should be used: connectivity schema, which describes connections between automata, event providers, controlled objects and transition graph which describes automaton behavior.

1-4244-0245-X/06/$20.00 ©2006 IEEE

III. UNIMOD TOOL

UniMod [18] is the open tool based on three open components: unified modeling language *UML*, *SWITCH*-technology and integrated development environment *Eclipse* [20]. *UniMod* tool is a plug-in for this development environment.

UniMod allows editing connectivity schemas (for example, figure 1) and state charts, verifying them, debugging diagrams in graphical mode etc.

Content of diagrams is automatically converted into the *XML*-description after verification. Additional fragments of code should be written on *Java* programming language: for event providers – their definitions, initializing, and conversion of system events to automata events; for controlled objects – methods implementing input actions and input variables.

Interpretation and compilation approaches can be used. In the first case *XML*-description is being interpreted and methods of *Java*-classes written manually are invoked. In the second case program code which implements automata system on *Java* language is built from *XML*-diagrams using *Velocity* [21] templates. This code and program fragments written manually are compiled jointly. In both cases the log of program execution is kept in terms of automata.

It is important to emphasize that level of abstraction rises, because diagrams become a sort of programs, because they are not pictures, but strict mathematical models. So program source code consists of two different-type parts: *UML*-diagrams and texts on *Java* programming language.

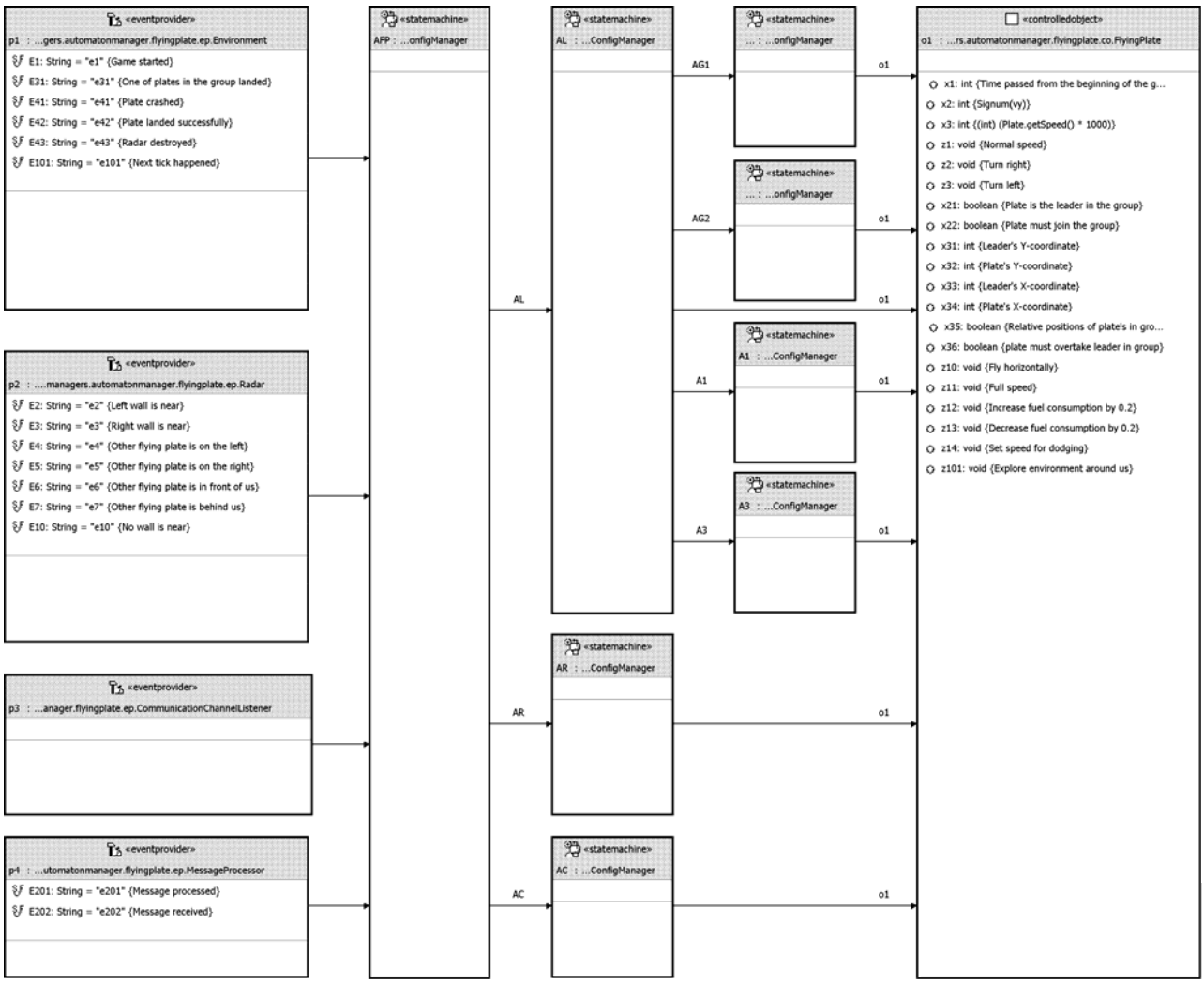

Fig. 1. Connectivity schema

Each *UniMod*-model consists of one connectivity schema and several state charts, whereas global system model can consist of several *UniMod*-models. The connection between *UniMod*-models is performed using event providers and controlled objects.

Velocity templates usage allows adapting compilation approach for programming languages different from *Java.*

IV. MULTI-AGENT SYSTEM DESCRIPTION

In this paper the technology of modeling of one class of multi-agent system using *UniMod* tool is illustrated on the example of the problem of *VI Open All-Siberian Olympiad in Informatics* [22]. The author of this problem is Dmitry Irtegov.

Flying objects in this problem are called flying plates. Each of them has a round form of a specified radius. A flying plate has reactive engine with fuel tank of specified volume and aerodynamic ailerons which allow rotating up to 25°.

The computer which can change fuel consumption and configuration of ailerons is placed on the flying plate. Flying plate can move with speed which is at least specified limit. Flying plate which speed is less than this limit falls on the ground. If its fuel tank is empty at that moment, it finishes the game normally; otherwise, it crashes.

Flying plate moves accordingly to Newton's Second Law. Its movement depends on two main forces: air resistance $\boldsymbol{F}$ and engine tractive force $\boldsymbol{T}$. Air resistance satisfies the equation $F = c_1 + c_2 v$, where $\boldsymbol{v}$ is a plate's speed, coefficients $\boldsymbol{c_1}$ и $\boldsymbol{c_2}$ are determined by plate's aerodynamic characteristics. Engine traction force satisfies the equation $T = c_4 q$, where q is fuel consumption. Fuel consumption is controlled by plate's computer and can change in specified limits. Coefficient $\boldsymbol{c_4}$ depends on engine characteristics.

Plates (agents) are grouped into teams of size N and "compete". Each "competition match" will be called "game" later on. There are two teams competing in each game. In the beginning of the game agents of the first team are randomly placed in the left part of the field on the first 25 meters from the start line. Agents of the second team are placed symmetrically in the right part of the field. The width of the field is specified and its length is infinite. Initial speed and direction are specified for each agent. In the simplest case, all initial speeds are equal and every initial direction is strictly forward. After "Start" command all agents start moving with the goal to move on maximal distance away from the start line.

Agents may collide with each other, because they can change fuel consumption and movement direction. Collisions are considered to be elastic. If relative speed of agents exceeds specified value, each collided agent crashes. Moreover, agents crash if they leave the field.

Aerodynamic mutual effects exist between agents. Each agent changes the environment state around it on the distance not exceeding specified aerodynamic interaction radius $\boldsymbol{L}$. Measurement unit for $\boldsymbol{L}$ is agent radius $\boldsymbol{R}$. Figure 2 shows how the environment changes around an agent.

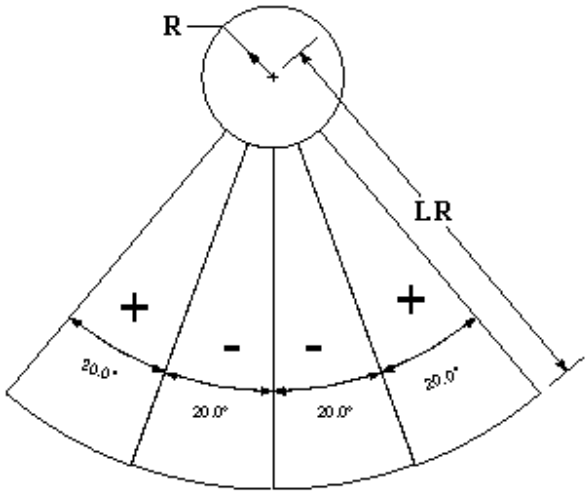

Fig. 2. Environment change around an agent

In areas marked with minus sign ("—") on figure 2 the air resistance increases by 50%. In addition, in areas marked with plus sign ("+") on figure 2 the air resistance decreases by 50%. Naturally, in the first case more fuel is needed than in the second case. While in the area of high air resistance, agents are trying to leave it. In other cases agents continue moving forward.

Several agents can simultaneously change the state of the same domain of the environment. In this case, for the sake of the simplicity, their summary influence is computed in the following way. Let $\boldsymbol{N_+}$ be the number of agents, which decrease air resistance in this domain, and $\boldsymbol{N_-}$ – number of agents, which increase air resistance. Let $\boldsymbol{\Delta N} = \boldsymbol{N_+} - \boldsymbol{N_-}$. If $\boldsymbol{\Delta N = 0}$, then air resistance in this domain is normal, if $\boldsymbol{\Delta N = 1}$ or $\boldsymbol{\Delta N = 2}$, then air resistance decreases by $\boldsymbol{50\Delta N}$ per cent. If $\boldsymbol{\Delta N > 2}$ then there is no air resistance in this domain. Air resistance in the area increases by $\boldsymbol{50|\Delta N|}$ per cent if $\boldsymbol{\Delta N}$ is negative.

Agent can either continue game or have finished it. Game continues while there is at least one agent that hasn't finished it. The game is considered to be finished after all agents finish it. To sum up game results only agents that have normally finished the game are taken into account. Team result is the maximal distance on which this team agent that has normally finished the game has moved away from the start line.

It is important to note, that since modeling of the whole system (which includes environment and agents) is made using computer, agent's control program affects it not continuously. Program can change agent movement parameters regularly, with constant period Δt. This period will also be named modeling step. Such situation is close to real situation, because even if opportunity of continuous agent control exists, it takes some time to gather full information about environment.

Figure 3 shows a screenshot of a situation which can happen during a competition. There one can see two agents (numbered 1 and 2) which have finished the competition due to their collision. Also one can see agents' trajectories and zones of their aerodynamic influence. Circles around agents designate their current speed.

To solve this problem you should develop and implement a winning strategy for your team.

V. AGENTS' BEHAVIOR DESCRIPTION

In this paper it is assumed that the winning strategy should be the following. A half of agents move forward and avoid other agents and field boundary, other agents in the same time gather in pairs. Agents in pairs are moving preserving their relative positions to use aerodynamic effects efficiently. This strategy usually wins in competition against strategy implemented for opponent – one agent is moving forward while others are attacking another team's agents.

Connectivity schema with eight automata is shown on figure 1. Event providers are used to implement agents' senses, finite automata – to describe agents' behavior, controlled objects – to effect on the environment.

In this version all "our" team agents' behavior is described by the same system of interacting automata.

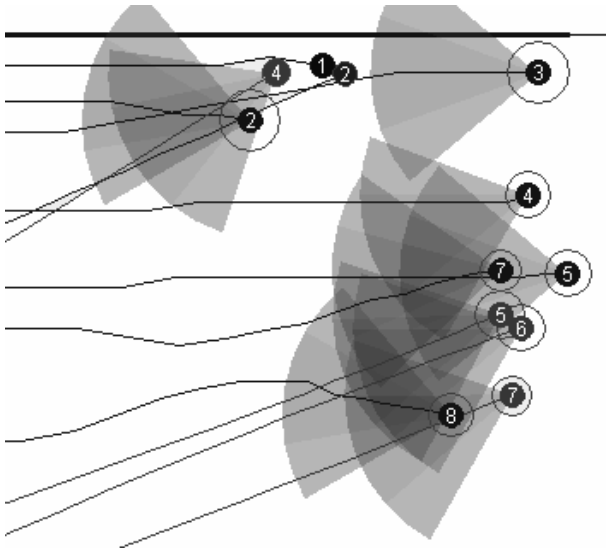

Fig. 3. Situation which can happen during a competition

Automaton *Agent state* has three states: "Flying", "Landed", "Crashed". Automaton *Flight mode* (figure 4) and automaton *Radar* are included in the state "Flying" of automaton *Agent state*.

Automaton *Avoiding field boundary and agents on the right and on the left* and automaton *Avoiding agents in front and behind* are included in the state "Flying alone" of automaton *Flight mode*. Automaton *The first in the pair* is included in the state "The first in the pair" of automaton *Flight mode*. Automaton *The second in the pair* is included in the state "The second in the pair" of automaton *Flight mode*. Furthermore, the automaton *Receiving and processing messages from other agents* is included in each of three states described above.

VI. MODELING

Multi-agent system modeling was carried out using *UniMod* tool according to the technology described in section III.

The development of multi-agent system under discussion was made in untraditional way, on the base of models. On the contrary, models are used in program design. So, program design is made in the framework of *MDA* (*Model Driven Architecture*) [23] which allows making the modeling platform-independent.

VII. SOURCE CODE STATISTICS

As mentioned in section III, *UML*-diagrams created by programmer are converted into text on *Java* programming language when compilation approach is used. Figure 5 shows the proportion of sizes of source code written manually and generated automatically from *UniMod*-model.

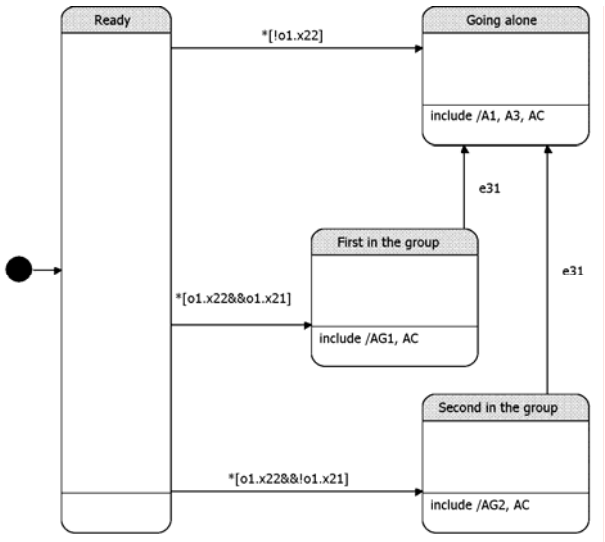

Fig. 4. Flight mode automaton

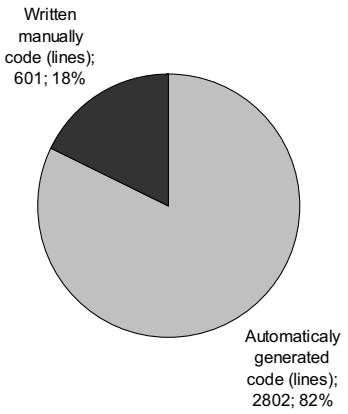

Fig. 5. Proportion of sizes of source code written manually and generated automatically

So, we can conclude that the main functionality of agent control system is implemented on the basis of finite automata. The size of source code generated automatically from *UniMod*-model exceeds more than four times the size of source code written manually. As a consequence, the programs reliability appreciably increases, particularly in connection with the fact that correctness of diagrams is automatically checked by many parameters during their creation.

VIII. EXPERIMENTAL RESULTS

Strategies' testing was held with the following values of agents' and modeling parameters (section IV):

$$c_1 = 0.625; \quad (1)$$
$$c_2 = 0.025; \quad (2)$$
$$c_4 = 3.125; \quad (3)$$
$$\Delta t = 0.3 \text{ second}; \quad (4)$$
$$L = 7. \quad (5)$$

Each team consisted of eight agents, each of them had 15 units of fuel at the start of the competition, and diameter of agent was equal to one meter, width of field – 40 meters. Initial speeds of agents were four m/s, initial directions were strictly forward. Initial positions of agents were randomly selected on first 25 meters of the field. Let us note that agents in the beginning of competition can be also placed deterministically if needed.

Let's note that for these parameters values it was experimentally set that one agent if it is alone on the field can show the result 200-205 meters. Result of group of agents can mount to 210-220 meters thanks to "positive" aerodynamic interaction between agents.

Two strategies were used as opponents' strategies. One of them (named simple later on) was such: all agents move strictly forward. In the second (aggressive) one all agents except one moved in the direction of "our" team agents to destroy them or pull away from the field and the last one moves strictly forward.

"Our" strategy and simple one competed in 30 competitions. Their results statistics are represented in table 1.

Table 1. Results of competitions against simple strategy

	N_{OK}	R_{our}	R_{op}	Δ
Max	8	225.6843	187.2180	39.4271
Min	5	205.1690	179.2453	20.1622
Mean	7.3667	214.6608	185.0916	29.5691

In table 1 N_{OK} means number of "our" team agents which successfully finished the competition, R_{our} – "our" team result, R_{op} – opponent's result, $\Delta = R_{our} - R_{op}$.

On the base of these experimental results a conclusion that "our" strategy is much better than the simple one can be made. In one of the competitions result of "our" team was 39 meters more and usually the advantage was about 30 meters. More than seven agents successfully end the competition usually, but there are initial arrangements in which two or three agents collide with each other or go out off the field.

Thirty competitions between "our" strategy and an aggressive one were held. Their results are shown in table 2.

Table 2. Results of competitions against aggressive strategy

	N_{OK}	R_{our}	R_{op}	Δ
Max	7	217.7150	195.2719	36.4381
Min	2	197.4160	179.0959	7.4978
Mean	5.2333	208.5202	187.8240	20.6961

Designations of N_{OK}, R_{our}, R_{op} and Δ in table 2 are the same as in table 1.

More detailed information about "our" team results in competitions can be obtained from figure 6, which shows the distribution of "our" team results in competitions against aggressive strategy.

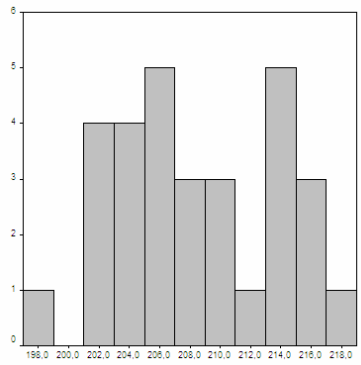

Fig. 6. Distribution of "our" team results in competitions against aggressive strategy

Another type of useful statistical information can be obtained from figure 7, which shows the distribution of number of "our" team agents successfully finished the competition against aggressive strategy.

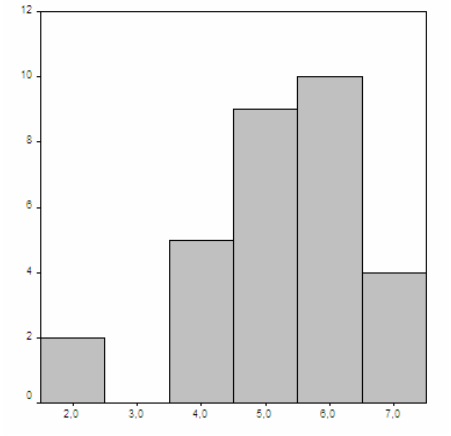

Fig. 7. Distribution of number of "our" team agents successfully landed in competitions against aggressive strategy

From these experimental results the following conclusion can be made. "Our" strategy successfully resists the aggressive one. Its average advantage is about 20 meters, and "our" rather successfully avoid collisions with agents of "aggressive" team (average number of agents successfully ended the competition is between five and six from eight).

IX. CONCLUSION

It is usually considered that novelty of the work in this domain of science should be in inventing some "good" strategy. Authors think that this paper's novelty is the formalization of process of chosen strategy implementation.

Simple automata model described in this paper allows to describe complex enough system behavior, and *UniMod* tool helps to efficiently implement it. Attention should be paid to the fact that *UniMod* checks diagrams' correctness during their creation. Researches on verification automata programs using methods based on *model checking* [24] in which authors take part started recently.

As a conclusion, we list features of approach proposed in this work and *NASA* approach [6]:

- finite automata usage;
- usage of tools for generating source code fragments from automata (in our work the question about generating the whole program from automata is solved);
- *model checking* usage.

This work was created in the *Programming technologies* laboratory organized by *SPbSU ITMO* and research and development center of *Borland* corporation.

REFERENCES

[1] Russel S., Norvig P., *Artificial Intelligence. A Modern Approach.* Prentice Hall. 2003.
[2] Skinner B.F., *Science and Human behavior.* London: Macmillan, 1953.
[3] Putnam H., Mind and machines. *Dimensions of Mind.* London: Macmillan, 1960, p.138–164.
[4] Rosenstein S.J., Formal Theories of Knowledge in AI and Robotics *New Generation Computing.* 1985. 3(4), p.345–357.
[5] Brooks R.A., A Robust Layered Control System for a Mobil Robot. *IEEE Journal of Robotics and Automation.* 1986. 2, p.14–23.
[6] Regan P., Hamilton S., NASA's Mission Reliable. *Computer*. 2004, January. p. 59–68.
[7] Shalyto A., *Technology of automata-based programming* http://is.ifmo.ru/technology/_tech_aut_prog.pdf
[8] Shalyto A., Switch-Technology. Algorithmization and Programming of Logic Control, SPb.: Science (Nauka), 1998. (in Russian). LC Control Number: 2001425055. http:/is.ifmo.ru/books/switch/1
[9] Shalyto A., Naumov L. Automata Theory for Multi-Agent Systems implementation. *Proceedings of International Conference Integration of Knowledge Intensive Multi-Agent Systems: Modeling, Exploration and Engineering.* KIMAS-03. Boston: IEEE Boston Section. 2003. http:/is.ifmo.ru/english/_aut_th.pdf
[10] Shalyto A., Tukkel N. Switch-Technology – Automata Approach to "Reactive" Systems Software Development. *Programming and Computer Software*. 2001. 27(5), pp. 260–276.
[11] Shalyto A., Naumov L., Korneev G. Methods of Object-Oriented Reactive Agents Implementation on the Basis of Finite Automata. Proceedings of International Conference Integration of Knowledge Intensive Multi-Agent Systems: Modeling, Exploration and Engineering. KIMAS-05. Boston: IEEE Boston Section. 2005. p.460–465. http://is.ifmo.ru/articles_en/_kimas05-1.pdf
[12] Shalyto A. New Initiative in Programming. Foundation for Open Project Documentation. http://is.ifmo.ru/download/SOPD.pdf
[13] Tukkel N., Shalyto A. Diesel-generator control system (fragment). State-based programming. Project documentation in Russian. http://is.ifmo.ru/projects/dg/
[14] Yartsev B., Korneev G., Shalyto A., Kotov V. Automata-Based Programming of the Reactive Multi-Agent Control Systems. *Proceedings of International Conference Integration of Knowledge Intensive Multi-Agent Systems: Modeling, Exploration and Engineering.* KIMAS-05. Boston: IEEE Boston Section. 2005. p.449–453. http://is.ifmo.ru/articles_en/_kimas05-2.pdf
[15] Yartsev B., Shalyto A., Software Development for Lego Mindstorms Using Automata-Based Approach (Project «Isenguard»). Project documentation in Russian. http://is.ifmo.ru/projects/lego/
[16] Tukkel N., Shalyto A., Robot Control System for "Robocode" Game. Variant 1. 2002. Project documentation in Russian. http://is.ifmo.ru/projects/dg/
[17] Kuznetsov D., Shalyto A., Robot Control System for "Robocode" Game. Variant 2. Project documentation in Russian. http://is.ifmo.ru/projects_en/robocode2/
[18] Gurov V.S., Mazin M.A., Narvsky A.S., Shalyto A.A. UniMod: Method and Tool for Development of Reactive Object-Oriented Programs with Explicit States Emphasis. *Proceedings of St. Petersburg IEEE Chapters. Year 2005. International Conference "110 Anniversary of Radio Invention".* SPb ETU "LETI". 2005. V. 2, pp. 106–110. http://is.ifmo.ru/articles_en/_unimod.pdf
[19] UniMod project. http://unimod.sourceforge.net
[20] Eclipse project. http://eclipse.org
[21] Velocity project. http://jakarta.apache.org/velocity/
[22] VI Open All-Siberian Olympiad in Informatics. Website in Russian. http://olimpic.nsu.ru/widesiberia/archive/wso6/2005/rus/index.shtml
[23] Model Driven Architecture. http://www.omg.org/mda/
[24] Clarke E. M., Grumberg O., Peled D. A., *Model checking.* The MIT Press. 2000.