2005 International Conference on "Integration of Knowledge Intensive Multiagent Systems. KIMAS '05: Modeling, Exploration, and Engineering". USA, MA: IEEE, 2005, pp. 449–453.

Automata-Based Programming of the Reactive Multi-Agent Control Systems

Boris Yartsev, yartsev@rain.ifmo.ru
George Korneev, kgeorgiy@rain.ifmo.ru
Anatoly Shalyto, shalyto@mail.ifmo.ru
Vladimir Kotov, kotov_v@rain.ifmo.ru
St. Petersburg State University of Information Technologies,
Mechanics and Optics
Computer Technologies Department
Sablinskaya street 14, St. Petersburg, Russia

Abstract—*Automata-based approach, proposed for the programming of the virtual autonomous reactive agents control systems used in the famous "Robocode" game is extended in this paper to the creation of the control systems of the reactive multi-agent real-life environment agents. We demonstrate the efficiency of the proposed approach by the example of creation of transportation system, which consists of two interacting robots and is assembled from the famous "Lego Mindstorms" kit. The advantage of this approach is the formalization of the implementation process and the simplification of testing and modification.*

1. INTRODUCTION

Recently a lot of attention has been devoted to the programming of the multi-agent systems [1-4]. One of the most important agent types are reactive agents [5-9].

It is appropriate to use finite automata to describe the behavior of the reactive agents [10-12]. Finite automata are also used in the description of "psychology of the behavior" in the book [13] and correspond to the model of Artificial Intelligence as behavior control system [14].

Despite of the theoretical research in the application of finite automata to the programming of reactive multi-agent systems [15] and single projects [6], automata are not widely used in the development of the software.

The aim of this paper is to describe the automata-based approach to the development of the reactive multi-agent systems. The approach is demonstrated on the example of the creation of a transportation system, which consists of two interactive robots, implemented with "Lego Mindstorms" kit.

There are many books about programming for "Lego Mindstorms" [16, 17]. But they don't describe the technology for control programs creation.

Some "Lego Mindstorms" projects are available in the internet [18-20]. Unfortunately such projects are usually poorly documented. Also we would like to notice, that in some papers finite automata [21-23] are used for the verification of the control systems for "Lego Mindstorms" robots [24, 25].

2. AUTOMATA-BASED APPROACH

The main ideas of the automata-based approach were formulated in [26-28]. These papers describe the process of the development of the virtual autonomous reactive agent control system, used in the famous "Robocode" game [29], created by the IBM Corporation.

This approach can be described in following steps.

1. Creation of the agent's interaction diagram.

2. Development of the agent's class-diagram.

3. Design of the classes structural diagram.

4. Creation of the automata interaction diagram if the agent behavior is represented by two or more automata. Automata can interact in three ways:
   - by nesting;
   - by invocation;
   - by exchanging the identifiers of their states.

5. Creation of four documents for each automaton.
   - Verbal description of the automaton behavior.
   - Automaton connections diagram. This diagram defines the interface of the automaton that is the mapping of symbolic names of the input variables and output actions used on other diagrams to the real variables and actions.
   - State transition diagram. This diagram formally describes behaivor of the automata. Variables and actions are marked by symbolic names defined on automata connections diagram. The states are also marked by the symbolic names. Even for the complicated behavior this diagram is quite compact.

- Source code of automata. Source code is generated or developed formally by the state-transition diagram and is isomorphous to it.

6. Development of the source code for each class according to its structural scheme.

7. Development of the environment emulator for the physical agents (robots).

8. Debugging of the control systems in environment emulator and environment emulator itself.

9. Debugging of the control systems in the real-life environment.

10. The release of the documentation.

We should notice, that input variables can be defined by quite complex functions.

For debug purposes each state has its numeric id that is logged when current state of the automaton is changed. So the debug on the 8^{th} and 9^{th} steps is preformed in the terms of the automata states [30].

The automata are implemented using automata variables. There is a corresponding variable for each automaton. These variables hold the id of the current state of the automata.

As it was mentioned before, automata can interact in three ways: by nesting, by invocation and by exchanging state ids. In one or several of its states one automaton can call the other automaton. This is called interaction by nesting. When one automaton calls the other in its output actions, it is called interaction by invocation. The third type of interaction is used in logical conditions—for example, the transition between two states can occur only if the automaton variable of the other automaton holds some specified value (the other automaton is in the specified state).

3. EXAMPLE

Problem Definition

Develop using the "Lego Mindstorms" kit the transportation system, which delivers the specified number of items from one place of the room or table to another.

The Results

Using the freeware program "MLCAD" [31], drafts of the robots were designed, and after that, robots were assembled.

The developed transportation system consists of two robots: the delivery robot and the vending robot, which interact via the infrared ports.

Vending robot stands on the vending place. It receives messages from the delivery robot via infrared port. The message encodes the number of the objects to give (one, two or three).

The delivery robot starts at the place of delivery. It receives the number of the objects to deliver via the remote control, operated by the user. When it receives a

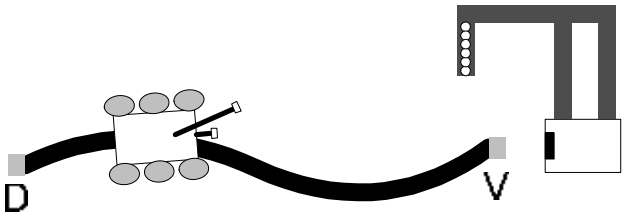

Figure 1—Scheme of operation

signal, it moves to the vending place and sends a message with the number of items to the vending robot.

Delivery robot takes the desired number of items and transports them to the place of delivery. The schematic plan of operation is shown on the figure 1. The delivery place is marked with "D" and the vending place is marked with "V". The delivery robot is searching for the path using the rod.

The place of the delivery and vending place are marked by pieces of the foil. The path between these places is marked by the black stripe on the white background. There are no obstacles on the path. The colors of the path and of the background are assumed to be uniform.

Each robot has a microcontroller. An operating system and the control program are loaded to the memory of the microcontroller via the infrared port. The microcontroller has 32Kb of RAM. In this project "leJOS" [32] operating system is used, that takes about 17Kb of RAM so the size of the memory left for the control program is 15Kb.

The control programs are developed in Java programming language [33] and runs under "leJOS". Only static methods are used because of lack of functionality in "leJOS". So for each automaton there was a corresponding static method.

Delivery Robot

The control program of the delivery robot consists of four automata.

1. The transport robot control automaton is responsible for message sending and rotation of the robot before moving back.

2. The path tracing automaton is responsible for moving along the path.

3. The line search automaton searches the path using the light sensor on the rod.

4. Rod centering automaton centers the rod.

The first two automata contain eight states. The third and the fourth automaton contain seven and four states respectively.

Vending Robot

The control program of the vending robot is implemented by one automaton named “Vending robot control”. This automaton contains four states.

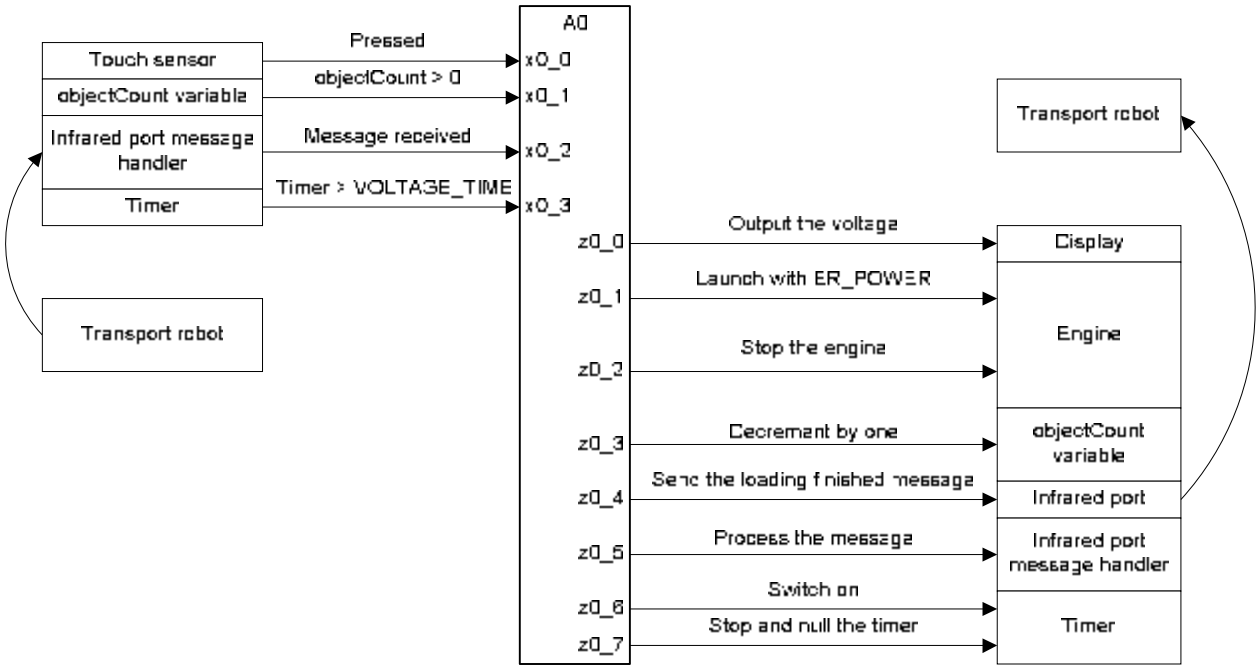

Figure 2 — Vending robot automaton connections diagram

4. AUTOMATA DOCUMENTATION EXAMPLE

Verbal Description

The initial state for the automaton “Vending robot control” is the “Start” state. Then the transition to the state “Voltage output” occurs, and the voltage of the batteries is displayed. In the “Voltage output” state, the automaton stays for the specified time, and than the transition to the state “Waiting for message” is performed. In this state, automaton stays unless the message from delivery robot is received. The number of items to give is extracted from the message, the engine is launched with the corresponding power and the transition to the “Item vending” state is performed. The transition between “Item vending” state and “Preparation for item vending” state is performed, when the object is given out. When desired number of items is given, automaton goes from “Preparation” to “Waiting for message”.

Connection Diagram

Connection diagram is shown on the figure 2. It contains short definitions of input variables and output actions.

Symbolic names of the input variables are prefixed by “x”, of the output actions—by “z” and symbolic names of the automata variables—by “y”.

In this example automata uses four input variables (x0_0, x0_1, x0_2 and x0_3) and eight output actions (z0_i for i from 0 to 7). Here the first index stands for automaton index and second index stands for id of the object within automaton.

Automaton variables (not shown on the figure 2), usually have a form of yi where i is an index of the automaton.

Transition Diagram

State transition diagram is shown on the figure 3.

Each transition has a guard condition that is specified on it. Guard condition consists of the input variables, automata variables and logical operators. The transition occurs, when its guard condition is satisfied. Some transitions have actions. States have an input (marked with “in:” prefix) and output (“out:”) actions.

When the transition occurs, the actions are executed. Actions are executed in the following order:

- output actions of the old state
- actions of the transition
- input actions of the new state

On the loop transition (see state 4 on the figure 3) neither “in” nor “out” output actions are executed.

5. CONCLUSIONS

The advantage of the proposed approach can be seen during in the process of the debugging of the control programs. At first, it started to work correctly very soon, and all the bugs were fixed fast and easily. There was also an opportunity to output the index of the active state on

the small LCD display of the microcontroller [12]. It is possible to output only four symbols on the display; with different approach debugging with such visualization could be more complicated.

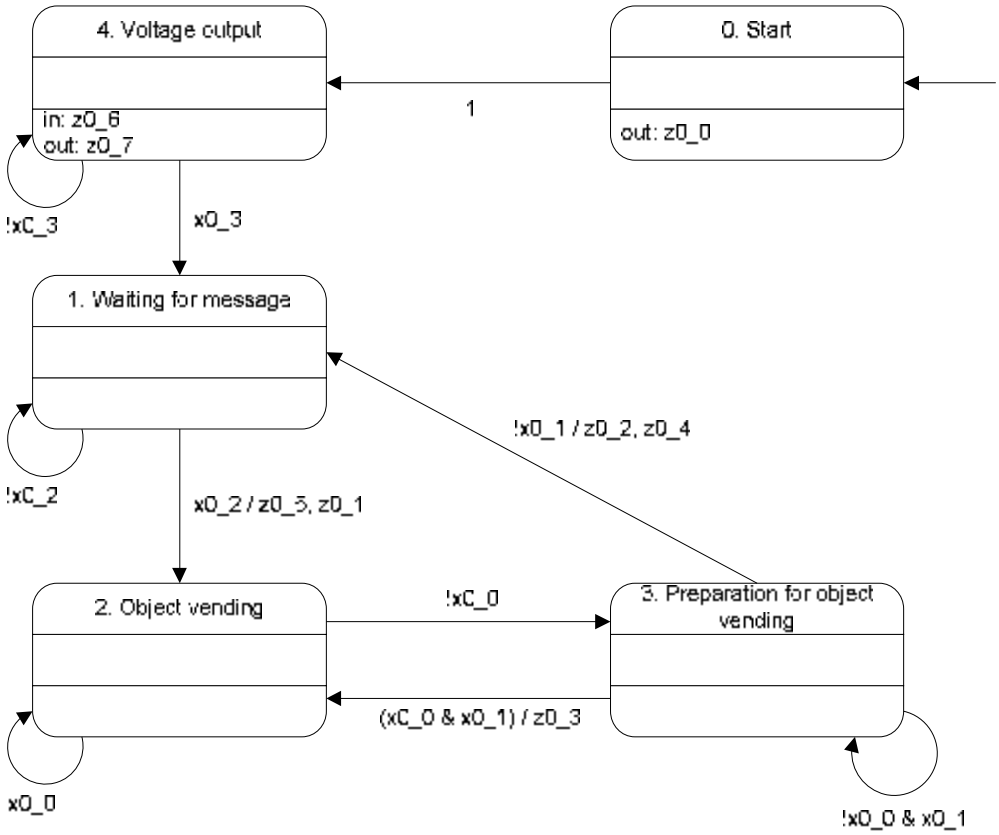

Figure 3 — Vending robot automaton state-transition graph

We would like to notice, that the debugging process could be really simplified, if the control programs at first are debugged in the environment emulator and only after that loaded in the appropriate agents (robots) and debugged in real-life environment. Environment emulation is also designed using state-based approach.

It follows from the example that all the states in the proposed approach can be of two types—control states and calculation states. The number of the control states is small, and the number of calculation states can be unbounded [34]. Being in the control state, automaton can pass many calculation states, which allows (as in hybrid automata [35]) development of the complicated behavior of each of the agents. This idea was used during the development of the software for virtual autonomous reactive agent in project [36].

From review of the documentation for the project [37], which was developed in the context of "Foundation for open project documentation" [38], it follows, that the proposed approach separates the design of the agents from writing the source code for their control systems [39, 40] and allows to create high-quality software, which can be easily modified.

At the end we would like to notice, that in this paper, in contrast to other works, two types of interactions are used—between the agents and between the automata in each of the agents. In both cases the interactions can be nested.

Project was carried out with the support of the Arcadia Inc. (Russia, Saint-Petersburg). Authors really appreciate the help from the Arcadia Inc. CEO Arcady Khotin.

REFERENCES

[1] Luger G., Artificial Intelligence, *Structures and Strategies for Complex Problem Solving*, MA: Addison Wesley, 2002.

[2] Shoham Y., Agent Oriented Programming, *Journal of Artificial Intelligence*, Vol.60, № 1, 1993.

[3] Genesereth M.R., Ketchpel S.P. Software agents, *Communications of the ACM*, Vol.37, № 7, 1994.

[4] Bradshaw J., *Software Agents*, MA: AAAI Press, 1997.

[5] Tarasov V.B., *From multi-agent systems to intellectual organizations: philosophy, psychology and informatics*, M.: Editorial URSS, 2002.

[6] Brooks R., Intelligence Without Representation, *Artificial Intelligence*, 47, 1991.

[7] Seel N., Intentional Description of Reactive Systems, *Decentralized Artificial Intelligence II*, Amsterdam: Elsevier North-Holland, 1991.

[8] Ferber J., *Les systems multi-agents vers une intelligence collective*, Paris: InterEditions. 1995.

[9] Murray D., *Developing Reactive Software Agents, AI Expert*, № 3, 1995.

[10] Shalyto A., Cognitive Properties of Hierarchical Representations of Complex Logical Structure, *10th IEEE International Symposium on Intelligent Control*, Monterey, California, 1995.

[11] Harel D., Politi M., *Modeling reactive systems with statecharts*, NY: McGraw-Hill, 1998.

[12] Shalyto A.A., Logic Control and "Reactive" Systems: Algorithmization and Programming, *Automation and Remote Control*, Vol.62, № 1, 2001. http://is.ifmo.ru/english/log_control/

[13] Piaget J., *Structuralism*, NY: Basic Books, 1970.

[14] Miller G., Gallanter E., Pribram K., *Plans and the structure of behavior*, NY: Holt, 1960.

[15] Naumov L., Shalito A., 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.

[16] Baum D., *Definitive Guide to Lego Mindstorms*, NY: Appress, 2000.

[17] Baum D., et al. *Extreme Mindstroms, An advanced guide to Lego Mindstorms*, NY: Appress, 2000.

[18] Kiwi B., Do it youself, *Computerra* 2002. №5.

[19] Brown JP., *Serious Lego.* http://jpbrown.i8.com/cubesolver.html

[20] Crosby M., *Lego projects.* http://www.mastincrosbie.com/mark/lego/lego.html

[21] Alur R., Kannan S., Yannakakis M., Communicating hierarchical state machines, *Proceedings of the 26th International Colloquium on Automata, Languages, and Programming*, LNCS 1644, 169–178, 1999. http://www.cis.upenn.edu/~alur/Icalp99chsm.html

[22] Alur R. Timed Automata., *Summer School on Verification of Digital and Hybrid Systems*, NATO-ASI, 1998. http://www.cis.upenn.edu/~alur/Nato97.ps.gz

[23] Alur R., Dill D., Automata-theoretic Verification of Real-Time Systems, *Proceedings of the 33rd IEEE Symposium on Foundations of Computer Science*, 177-186, 1992.

[24] Iversen T., Kristoffersen K., Larsen K., et al. Model-Checking Real-Time Control Programs. Verifying Lego Mindstorms Systems Using UPPAAL, *12 Euromicro Conference on Real-Time Systems*, 2000.

[25] Laursen M., Madsen R., Mortensen S. *Verifying Distributed LEGO RCX Programs Using UPPAAL*. http://citeseer.ist.psu.edu/laursen99verifying.html

[26] Shalyto A., Tukkel N., Tanks and automata, *Byte/Russia*, № 2, 2003. (article in Russian). http://is.ifmo.ru/download/tanks_new.pdf

[27] Tukkel N., Shalyto A., *Tank control system for "Robocode" game. Version 1. (project documentation in Russian)*, http://is.ifmo.ru/projects/tanks/

[28] Kouznetsov. D., Shalyto A., *Tank control system for "Robocode" game*. Version 2. http://is.ifmo.ru/projects_en/robocode2/

[29] "Robocode" game homepage http://robocode.alphaworks.ibm.com/home/home.html

[30] Shalyto A.A., *SWITCH-technology. Algorithmic and programming methods in solution of the logic control problems.* St. Petersburg: Nauka (Science), 1998. LC Control Number: 2001425055

[31] "Lego" CAD program MLCAD http://www.lm-software.com/mlcad/

[32] "leJOS" operating system project http://lejos.sourceforge.net/

[33] Java programming language http://java.sun.com/

[34] Shalyto A.A., Tukkel H.I., From Turing programming to automata-based programming, *PC World (Russia)*, № 2, 2002.

[35] *The Hybrid Systems Project*, http://control.ee.ethz.ch/~hybrid/

[36] Belyaev A.V., Suyasov D.I., Shalyto A.A., *Computer Game "Cosmonaut"*, http://is.ifmo.ru/projects_en/cosmo/

[37] Yartsev B.M., Korneev G.A., Shalyto A.A., *Software Development for Lego Mindstorms Using Automata-Based Approach (Project "Isenguard")*, http://is.ifmo.ru/projects_en/lego/

[38] Shalyto A., *New Initiative in Programming. Foundation for Open Project Documentation*, http://www.linuxsummit.org/archive2004/shalyto_foundation.pdf

[39] Shalyto A.A., Tukkel N.I., SWITCH-Technology: An Automated Approach to Developing Software for Reactive Systems, Programming and Computer Software, *Programming and Computer Software*, 27(5), 2001. http://www.kluweronline.com/issn/0361-7688

[40] Auslander D.M., Ridgely J.R., Ringgenberg J.D., *Object-Oriented Design in a Real-Time World,*. NJ: Prentice Hall, 2002.