Version of this article was published in "PC World/Russia". 2003, № 10

TECHNOLOGY OF AUTOMATA-BASED PROGRAMMING

Anatoly Shalyto

The winner of the contest of research projects in sphere of integral scheme design automation, that was organized by **Intel Corporation** in Commonwealth of Independent States in 2003

shalyto@mail.ifmo.ru

Saint-Petersburg State University of Information Technologies, Mechanics and Optics
Computer Technologies Department, http://is.ifmo.ru

Introduction

In recent years great attention has been paid to the development of the technology of programming for the embedded systems and real-time systems. These systems have special requirements for the quality of software. One of the most well known approaches for this field of tasks is synchronous programming [1].

Simultaneously with the progress of synchronous programming in Europe, in Russia an approach, called "**automata-based programming**" or "**state-based programming**" is being created [2–4]. This method could be considered as a type of synchronous programming.

This paper describes main properties of automata-based programming. It contains such phases of software developing as designing, implementing, debugging and documenting.

The term event in the programming has been used wider and wider in programming. It has become one of the main terms in software development. The offered approach is based on the term "**state**". After introduction of the term "**input action**", which could be an **input variable or an event**, the term "**automaton without outputs**" could be brought in. After addition of the term "**output action**", the term "**automaton**" could be brought in. It is the finite determined automaton.

That is why, the sort of programming, which is based on this term was called "automata-based programming" in paper [4]. So the process of software creation could be named "**automata software design**".

The feature of this approach is that automata, used for developing, are defined with the help of transition graphs. For distinguishing of the nodes of these graphs the term "**state coding**" is to be introduced. When using "**multiple state coding**" with the help of single variable it is possible to distinguish amount of states which is equal to the amount of variables values. This allows to introduce in programming the term "**program observability**".

Using the offered approach, programming is to be performed using the concept of "**state**", not the concept of "flag variables". It allows to understand and specify the task and its parts (subtasks) better.

It is necessary to note that it automata-based programming debug is performed with the help of drawing up the protocols (logging) in the terms of automata.

In this approach there is a formal and isomorphic method of transfer from the transition graph to the software source code. So when using programming languages of the high level, the most suitable way is to use construction which is similar to construction "**switch**" of the programming language C. That is why the technology of "automata-based programming" in paper [4] was called as "**Switch-technology**".

Nowadays this technology is been developed in several variants, for different types of task to be solved and for various type of computing devices.

Logical Control

In 1996 Russian Fund for Fundamental Investigations in the context of publishing project № 96-01-14066 supports the publishing of book [4]. Offered technology was described in this book, being applied to the systems of logical control, in which there are no events, input and output actions are binary variables and operating system is working in the scanning mode.

Systems of this class as usual are to be implemented on **programmable logical controllers**, which have relatively little memory and programming is to be performed using specialized languages (for example, the language of functional blocks) [5]. In work [4] formal methods of program developing for such kind of devices were offered. These methods are based on the giving specification for the developing project with the help of set of connected transition graphs. There were shown opportunities which are given by the language of transition graphs comparing with "Graphset" language.

State-Based Programming

Henceforth automata approach was spread to the **event-based (reactive) systems** [6]. In systems of this kind all limitations, mentioned above are taken away. It is obvious from the name of these systems that events are used among the input actions. Role of output actions could be played by arbitrary functions. Any real-time operating system could be used as an environment.

For programming for event-based systems with the help of automata a procedural approach to software developing was used. So this king of programming was called as "**state-based programming**" [7].

Using this method output actions are assigned to the arcs, loops or nodes of the transition graphs (mixed automata are to be used – Moore-Mealy automata). This allows to present sequences of actions, which are reactions to the corresponding input actions, in the compact form.

On of the features of programming for the reactive systems is that liquidation of logic in the event handlers and forming of system of interacting automata, which are called from these handlers, causes logic centralization [8]. Automata in such system can interact by nesting, by calling ability and with the help of state numbers interchange.

Last type of interaction is described in work [9], which declares that "this type of interaction may be used as powerful tool for program verification".

System of interconnected automata forms system-independent part of software. At the same time system-dependent part is formed by functions of input and output actions, event handlers and so on.

Another important feature of this approach is that automata in it are used **thrice: for specification, for implementation** (they stay in the source code) and **for drawing up the protocol**, which is performed, as said above, **in terms of automata**.

Last property allows to verify the propriety of automata system functioning. Logging is performed automatically, it is based on the created program. This mechanism could be also used for large scale tasks and for task with difficult, smeared software logic.

Any drawn up protocol could be considered as the action script. Note that for the large tasks it is impossible to use sequence diagrams, cooperation diagrams, which are parts of UML language [10]. That is because usage of UML language listed diagrams are suggested to be drawn up manually on the projection phase of developing. In automata-based programming protocols will be build automatically at the run-time.

Protocols allow to keep an eye on program execution and demonstrate that the fact is that automata are not just the "pictures", but they are actually functioning entities.

Automata approach is offered to be used not only for controlling system developing, but also for objects of control modeling.

This method was approved on the task of developing of controlling system for ship diesel generator [11]. Mentioned system was specified with help of more than thirteen interacting automata. For describing of model of diesel generator automata were also used. While designing each automaton was provided with four following documents:

- verbal description ("expectancies declaration ");
- bonds scheme, which explains on the human language all symbols which are involved in automatons interface;
- state transition graph with symbolic indication of events, input and output actions;
- source code of program module, which realizes state transition graph (also without usage of meaningful identifiers and comments).

These documents replace self documenting programs, which contains meaningful identifiers and comments. These standard facilities cannot ensure the understandability and clearness of the source code for further developing, modifying and enlarging [12]. This problem for difficult logic cannot be solved also with the help of self documenting state transition graphs [10].

Realized project had proved expediency of protocols usage for verifying propriety of interaction of such great amount of automata and each automaton separately.

This project had been implemented for computing system with architecture ix86. Henceforward described approach was evolved by N. I. Tukkel for developing systems for microcontrollers. Advantage of automata technology is that all design may be done on personal computer using Switch-technology and only on last phase of developing software could be ported to microcontroller.

State-Based Object-Oriented Programming

The composite approach may be rather useful for solving task from a very large spectrum, it is based on **object-oriented and automata-based programming paradigms**. In work [13] this method was called as "**state-based object-oriented programming**".

Feature of this approach is that, like in Turing machines [14], here controlling (automata) states are explicitly singled out.

The amount of these states is noticeably fewer than amount of all other objects states (for example, run-time states).

The term "**states space**" was introduced in programming. This term means the set of object **controlling states**.

So this approach provides more understandable behavior in comparison with case when such space is absent.

The minimal set of documents, which are necessary for the visually and clearly, but strictly describe structural (static) and behavioral (dynamic) sides of the software.

As when using any other approach, this approach needs the set of heuristics, back-steps, accurate definitions and concurrent tasks. But after program has been developed offered approach may be named an ideal technology for storing of decisions which had been made.

After analysis of the knowledge domain the classes could be defined and class diagram may be drawn up.

For each class the verbal description needs to be created at least in the form of list of tasks to be solved.

For each class the structure scheme is to be developed. This scheme describes its interface and structure. Important note is that classes attributes and member functions are to be separated to automaton and others.

If there are several automata in the same class then there is a need in drawing up their interaction scheme.

For each automaton verbal definition is to be developed with the bond scheme and state transition graph.

Each class implemented with the separate program module. Its structure is isomorphic to classes structure. Member functions, which have corresponding automata, are realizable according to the template, described in paper [8].

For system debugging and for confirmation of propriety of its work protocols are automatically built during run-time. These protocols describe the functioning of objects, which contains automata, in terms of states, transitions, events, input and output actions.

Project documentation is to be made. Its important part is the program documentation.

From stated above it can concluded that application of automata makes programs behavior clearer as objects using makes programs structure clearer.

Described approach was used for developing tank controlling system for "Robocode" system [15].

In contrast to controlling systems of hundreds of other tanks, there is **a detailed project documentation** for this tank. Documentation contains state transition graphs, bond schemes and others.

Detailed protocols of tanks behavior allow to observe the history of battle. Method of protocols drawing up may form a base of new paradigm of blackbox functioning.

In the described technology automata were implemented inside classes' member functions. But there could be another approaches to the object realization of automata, which were described, for example in papers [16–18]. Automata may stand out, in particular, and classes, which are descendants of the class that realizes automata basic functionality. This functionality is declared by the semantics of Switch-technology.

It is also possible to use classes, which realizes concepts of "state" or "group of states".

For automata programs design pattern "state" [19] or other patterns may be also used.

Peculiarities of automata implementation of parallel processes, based on messages interchange, are considered in paper [20].

Existence of high quality project documentation makes further program refactoring (changing of programs structure, keeping its functionality the same) much easier. Last thesis was confirmed by refactoring of tank controlling system, which was mentioned above. Refactoring was complete for rising programs "objectness".

Computational Algorithms

Automata approach is used now for computational algorithms implementation [21–23].

It was shown that arbitrary iterative algorithm can be implemented with the help of construction, that is equivalent to loop operator "do ... while", inside which there is single "switch" operator.

New state-based approach to algorithms animation programs creation was offered. Such visualization programs are widely used on Computer Science Department of Saint-Petersburg State University of Information Technologies, Mechanics and Optics for students teaching of programming and discrete mathematics [24].

Approach allows to represent visualization programs logic as a system of interacting finite automata. Systems consists of pairs of automata, each of them contains "forward" and

"back forward" automata, which provides step-by-step algorithms execution forwards and back forwards respectively.

One of the aims of this paper if to show that automata can be used not only for language recognition [25] and washing machines control. That they are not only one of mathematical models of discrete mathematics, but can be used for implementation of any programs, which have complicated behavior.

Foundation for Open Project Documentation

At the opening of semifinal competitions of world team championship on programming ACM (Association for Computing Machinery) in north-western region 27 of November 2002 the "Foundation for Open Project Documentation" was declared. In the context of this foundation on site **http://is.ifmo.ru** section "Projects" was created. In this section soon will be placed over 50 projects, which are examples of software developing, using automata approach. Lets list some of them:

- automata realization of scripts for educational animation using *Macromedia Flash*;
- XML-format for video players appearance description (www.crystalplayer.com);
- combined usage of compilers developing theory and Switch-technology;
- tank controlling system for the *Robocode* game (sponsored by IBM) [15]. Tank *Cynical*, that had been developed, was decided to be one of the best in the world. There also will be placed its refactored version, which, as stated above, was developed for rising programs "objectness";
- controlling systems for lift (like Knuth's, but better), cars, percolators, turnstile, semaphores and many others;
- possible, automata-based solutions of classical tasks:
  - task of synchronization of the chain of the shooters;
  - task of "philosophers dinner";
  - task of Hanoi-towers [21];
  - task of knights move [22];
- many well-known classical algorithms (for example, *QuickSort*, search of substrings and many others);
- illustrations of different approaches to state-based object-oriented programming:
  - automata as classes;
  - automata as member functions;
  - formal method of conversion of object-oriented automata programs to programs, written using classical Switch-technology. It allows to develop models of system using personal computer and then formally port it to microcontrollers platform;
  - "state" pattern;
  - classes, which realize concepts of "state" and "group of states";
  - it was showed that automata have properties of objects, used in object-oriented paradigm;
- automata based graphical users interface toolkit;
- windows manager, with supports all necessary functionality (dragging, groping, resizing, overlapping, minimizing/maximizing, closing and so on);
- realization of protocol SMTP (Simple Mail Transport Protocol);
- a lot of computer games:
  - *Lines*;
  - *Automatic Bomber*;

- *Sea wars*;
- *Zavalinka* (bank) and others.

In summary lets note that usage of automata makes formalization of specification of program simpler. It defines programs behavior and plays "key role in matter of inhibition of errors" [26].

It is important to note that "state-oriented programming" style is widely used for solving tasks of logical control (classification of programming styles was offered in book [27]). *State based programming stands upon two styles: "state-oriented programming" and "event-based programming*". Object-oriented state based programming takes up object-oriented style and two styles, mentioned above.

The work was carried out with the support of Russian Fund for Fundamental Investigations according to grant №02-07-90114 "Technology of automata-based programming development".

References

1. *Benveniste A., Caspi P., Edwards S. et al*. The Synchronous Languages 12 Years Later // Proceedings of the IEEE. Vol. 91. 2003. № 1.
2. *Shalyto A.A.* Logic control and "reactive" systems: algorithmization and programming // Automation and Remote Control (Avtomatika i Telemekhanika). 2001. № 1. http://is.ifmo.ru.
3. *Shalyto A.A*. Logical control. Methods of hardware and software algorithms implementation. SPb.: Nauka (Science), 2000.
4. *Shalyto A.A*. Switch - technology. Algorithmization and programming of tasks of logical control. SPb.: Nauka (Science), 1998.
5. *Shalyto A.A.* Realization of algorithms of logical control with the help of programs written using language of functional blocks // Industrial automatic control system and controllers. 2000. № 4. http://is.ifmo.ru.
6. *Harel D., Politi M*. Modeling Reactive Systems with Statecharts. NY: McGraw-Hill, 1998.
7. *Shalyto A., Tukkel N.* State-Based Programming // World of PC. 2001. № 8, 9. http://is.ifmo.ru.
8. *Shalyto A.A., Tukkel N.I.* Switch - technology – automata approach to software developing for "reactive" systems // Programming and Computer Software (Programmirovanie). 2001. № 5. http://is.ifmo.ru.
9. *Dijkstra E.* Interaction of consequent processes // Programming languages. M.: Mir (World), 1972.
10. *Booch G., Rambo D., Jacobson A*. UML Language. Users manual. M.: DMK, 2000.
11. *Shalyto A.A., Tukkel N.I.* Software design for diesel generator controlling system based on automata approach //System for control and information processing. SPb.: NPO "Avrora", 2003, № 5. http://is.ifmo.ru.
12. *Bezrukov N.* Repeated look at "council" and "bazaar" //BYTE/Russia. 2000. № 8.
13. *Shalyto A.A., Tukkel N.I.* State-Based Object-Oriented Programming // Proceedings of International Scientific and Technical Conference "Artificial Intelligence – 2002". V.1. Taganrog - Donetck: TGRU - DIPII, 2002.
14. *Shalyto A., Tukkel N.* From Turing programming to automata //World of PC. 2002. № 2. http://is.ifmo.ru.
15. *Shalyto A.A., Tukkel N.I.* Tanks and automata //BYTE/Russia. 2003. № 2. http://is.ifmo.ru.

16. *Gurov V.S., Narvsky A.S., Shalyto A.A.* Automatization of projecting of event-based state-based object-oriented programs // Proceedings of X Scientific and Technical Conference "Telematica–2003". V.1. SPb.: SPbIFMO (TU), 2003. http://tm.ifmo.ru.
17. *Shopyrin D.G., Shalyto A.A.* Usage of class "STATE" in state-based object-oriented programming // Proceedings of X Scientific and Technical Conference "Telematica–2003". V.1. SPb.: SPbIFMO (TU), 2003. http://tm.ifmo.ru.
18. *Korneev G.A., Shalyto A.A.* Realization of finite automata with the help of object-oriented programming // Proceedings of X Scientific and Technical Conference "Telematica–2003". V.1. SPb.: SPbIFMO (TU), 2003. http://tm.ifmo.ru.
19. *Gamma E., Helm R., Johnson R., Vlissides J.* Methods of object-oriented programming. Design patterns. SPb.: Piter, 2001.
20. *Guisov M.I., Kuznetsov A.B., Shalyto A.A.* Integration of mechanism of messages interchange into Switch-technology. http://is.ifmo.ru.
21. *Shalyto A., Tukkel N., Shamgunov N.* Hanoi-towers and automata //Programmer. 2002. № 8. http://is.ifmo.ru.
22. *Shalyto A., Tukkel N., Shamgunov N.* Knights move task //World of PC. 2003. № 1. http://is.ifmo.ru.
23. *Shalyto A.A., Tukkel N.I.* Transformation of iterative algorithms to automata // Programming and Computer Software (Programmirovanie). 2002. №5. http://is.ifmo.ru.
24. *Korneev G.A., Kazakov M.A., Shalyto A.A.* Construction of functional logic of algorithms visualization programs based on automata approach // Proceedings of X Scientific and Technical Conference "Telematica–2003". V.2. SPb.: SPbIFMO (TU), 2003. http://tm.ifmo.ru.
25. *Hopcraft D., Motvani R., Ulman D.* Introduction to theories of automata, languages and calculations. M.: Williams, 2002.
26. *Allen E.* Typical design errors. SPb.: Piter, 2003.
27. *Nepeyvoda N.N., Skopin I.N.* Programming Basis. Moscow-Ijevsk: Institute of Computer Research, 2003.