НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
июль–август 2023 Том 23 № 4 http://ntv.ifmo.ru/
SCIENTIFIC AND TECHNICAL JOURNAL OF INFORMATION TECHNOLOGIES, MECHANICS AND OPTICS
July–August 2023 Vol. 23 No 4 http://ntv.ifmo.ru/en/
ISSN 2226-1494 (print) ISSN 2500-0373 (online)

doi: 10.17586/2226-1494-2023-23-4-750-756

Verification of event-driven software systems using the specification language of cooperating automata objects

Irina V. Afanasieva[1], Fedor A. Novikov[2], Ludmila N. Fedorchenko[3]✉

[1] Special Astrophysical Observatory of the Russian Academy of Sciences (SAO RAS), Nizhny Arkhyz, 369167, Russian Federation
[2] Peter the Great St. Petersburg Polytechnic University (SPbPU), Saint Petersburg, 195251, Russian Federation
[2] Alferov Academic University, Saint Peterburg, 194021, Russian Federation
[3] St. Petersburg Federal Research Center of the Russian Academy of Sciences, Saint Petersburg, 199178, Russian Federation

[1] riv615@gmail.com, https://orcid.org/0000-0003-4225-4124
[2] fedornovikov51@gmail.com, https://orcid.org/0000-0003-4450-0173
[3] lnf@iias.spb.su✉, https://orcid.org/0000-0002-4008-9316

Abstract
The CIAO (Cooperative Interaction Automata Objects) specification language is intended to describe the behavior of distributed and parallel event-driven systems. This class of systems includes various software and hardware systems for control, monitoring, data collection, and processing. The ability to verify compliance with requirements is desirable competitive advantage for such systems. The CIAO language extends the concept of state machines of the UML (Unified Modeling Language) with the possibility of cooperative interaction of several automata through strictly defined interfaces. The cooperative interaction of automata objects is defined by a link scheme that defines how the provided and required interfaces of different automata objects are connected. Thus, the behavior of the system as a whole could be described as a set of execution protocols, each of which is a sequence of interface calls, possibly with guard conditions. We represent a set of protocols using a semantic graph in which all possible paths from the initial nodes to the final nodes define sequences of interface method calls. Because the interfaces are strictly defined in advance by the connection scheme, it is possible to construct a semantic graph automatically according to a given system of interacting automaton objects. To verify the system behavior, one only has to check if each path in the semantic graph does satisfy the requirements. System requirements are formally described using conditional regular expressions that define patterns of acceptable and forbidden behavior. This article proposes methods and algorithms that allow you to check the compliance of programs in the CIAO language with the requirements automatically and, thereby, check the semantics of the developed program. The proposed method narrows the specification formalism to the class of regular languages and the programming language to a language with a simple and predefined structure. In many practical cases, this is sufficient for effective verification.

Keywords
verification, event-driven programs, critical system development, regular expressions, elevator control system

Acknowledgments
The work of Irina Afanasieva was carried out within the framework of the SAO RAS State Assignment approved by the Ministry of Science and Higher Education of the Russian Federation no. 121092300060-6.

For citation: Afanasieva I.V., Novikov F.A., Fedorchenko L.N. Verification of event-driven software systems using the specification language of cooperating automata objects. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2023, vol. 23, no. 4, pp. 750–756. doi: 10.17586/2226-1494-2023-23-4-750-756

© Afanasieva I.V., Novikov F.A., Fedorchenko L.N., 2023

УДК 004.415.52, 004.434

Верификация событийно-управляемых программных систем с использованием языка спецификации взаимодействующих автоматных объектов

Ирина Викторовна Афанасьева[1], Федор Александрович Новиков[2], Людмила Николаевна Федорченко[3]✉

[1] Специальная астрофизическая обсерватория Российской академии наук, Нижний Архыз, 369167, Российская Федерация
[2] Санкт-Петербургский политехнический университет Петра Великого, Санкт-Петербург, 195251, Российская Федерация
[2] Санкт-Петербургский национальный исследовательский Академический университет имени Ж.И. Алфёрова Российской академии наук, Санкт-Петербург, 194021, Российская Федерация
[3] Санкт-Петербургский Федеральный исследовательский центр Российской академии наук, Санкт-Петербург, 199178, Российская Федерация

[1] riv615@gmail.com, https://orcid.org/0000-0003-4225-4124
[2] fedornovikov51@gmail.com, https://orcid.org/0000-0003-4450-0173
[3] lnf@iias.spb.su✉, https://orcid.org/0000-0002-4008-9316

Аннотация

Введение. Язык спецификации Cooperative Interaction Automata Objects (CIAO) предназначен для описания поведения распределенных и параллельных систем, управляемых событиями. К этому классу систем относятся различные программно-аппаратные комплексы управления, контроля, сбора и обработки данных. Возможность автоматической проверки соответствия требованиям является желательным конкурентным преимуществом событийно-управляемых программных систем. Язык CIAO расширяет концепцию конечных автоматов (Unified Modeling Language, UML) возможностью кооперативного взаимодействия нескольких автоматов через строго определенные интерфейсы. Кооперативное взаимодействие автоматных объектов определяется схемой связи, которая связывает предоставленные и требуемые интерфейсы различных автоматных объектов. Таким образом, поведение системы в целом можно описать как набор протоколов выполнения, каждый из которых представляет собой последовательность вызовов интерфейса, возможно со сторожевыми условиями. **Метод.** Представлен набор протоколов с помощью семантического графа, в котором все возможные пути от начальных к конечным узлам определены последовательностью вызовов методов интерфейса. Благодаря тому, что интерфейсы заранее строго определены схемой связи, возможно автоматическое построение семантического графа по заданной системе взаимодействующих автоматных объектов. Для проверки поведения системы достаточно убедиться, что каждый путь в семантическом графе удовлетворяет требованиям. Системные требования формально описаны с помощью условных регулярных выражений, определяющих шаблоны допустимого и запрещенного поведения. **Основные результаты.** Предложены методы и алгоритмы, позволяющие автоматически проверить соответствие программ на языке CIAO заданным требованиям. **Обсуждение.** Разработанный метод сужает формализм для задания спецификаций до класса регулярных языков, а язык программирования — до языка с простой и предопределенной структурой. Во многих практических случаях этого достаточно для эффективной верификации.

Ключевые слова

верификация, событийно-ориентированные программы, разработка критических систем, регулярные выражения, система управления лифтом

Благодарности

Работа Ирины Афанасьевой выполнена в рамках государственного задания Специальной астрофизической обсерватории РАН № 121092300060-6, утвержденного Министерством науки и высшего образования Российской Федерации.

Ссылка для цитирования: Афанасьева И.В., Новиков Ф.А., Федорченко Л.Н. Верификация событийно-управляемых программных систем с использованием языка спецификации взаимодействующих автоматных объектов // Научно-технический вестник информационных технологий, механики и оптики. 2023. Т. 23, № 4. С. 750–756 (на англ. яз.). doi: 10.17586/2226-1494-2023-23-4-750-756

Introduction

Event-driven programs, otherwise called discrete reactive systems [1, 2], are often found in the tasks of control, monitoring, data collection, and processing. An event-driven system responds to emerging events (stimuli) by performing certain actions (reactions). Such systems are often classified as critical systems [3, 4] for which the formulation of requirements and verification of compliance with the requirements are nontrivial tasks. Ordinary verbal formulations and selective testing are not enough for critical systems, and the use of formal verification methods [2, 5] is necessary.

At the same time, for event-driven systems, it is not enough to specify a logical precondition that must be met before starting work and a logical postcondition that must be met because of the implementation of a certain sequence of events/actions, since the same set of actions can be performed both in permissible sequence and in an undesirable forbidden sequence.

Thus, the formal requirements for the systems of the class under consideration must be set in the form

of a description of both admissible and inadmissible sequences of events/actions. Various methods are known for the formal description of a set of sequences which vary depending on how diverse the elements of the sequences are and how complex the sequences are organized.

In this case, elementary events/actions can be interpreted as symbols of a finite alphabet, sequences of actions can be interpreted as words in this alphabet, and in this case, formal requirements put some language in a given alphabet [6].

The purpose of this article is to present methods and algorithms that allow for a certain class of reactive systems, namely for systems described in the Cooperative Interaction Automata Objects (CIAO) language [7, 8], to build a formal description of a set of sequences of actions (a set of possible execution protocols) in a form very close to conditional regular expressions [9] automatically.

The application of the CIAO specification language for building event-driven systems

The practical application of the CIAO language in the field of creating control and data processing systems has shown good results, in particular, a high degree of reliability of software created using this language [10]. The CIAO language is based on the use of state transition graphs to describe the behavior of reactive systems, and Unified Modeling Language state machine diagrams [11] are used as transition graphs extended with additional constructions and conventions to increase the expressive power of the language. The most significant innovation of the CIAO language is, on the one hand, the multiplicity of interacting automata objects and, on the other hand, the strict typing of interaction interfaces.

In the article [8], we considered the use of the CIAO language for building control systems using the example of the elevator control problem described by D. Knuth [12]. In this problem, the actions that the elevator can perform, the conditions that the control algorithm can check, and the requirements that the control algorithm must satisfy are specified. Because of applying the technique described in the article [8], the specification of the elevator control algorithm in the CIAO language was obtained, as shown in Fig. 1.

Here x is the number of the starting floor, y is the number of the target floor, t_1 is the maximum waiting time for the passenger to start service, t_2 is the maximum waiting time for the elevator to enter the passenger

This specification contains states, events, actions, and guard conditions. Abbreviated identifiers according to the approach proposed in [13] accompanied by short descriptions are given in Table 1.

For a passenger, all events come from within, from free will, and only the *ez*5 event comes from the elevator. The elevator, on the other hand, has no free will, and all the events of the elevator are the actions of the passenger.

In article [8], the requirements for the control system are formulated as follows.

1. All requests to move to the floors inside the elevator must be serviced.
2. All elevator call requests from floors must be serviced.

In the article [8], it is also shown how the specification in Fig. 1 guaranties the fulfillment of requirements 1 and 2. Here we go further.

When developing critical systems, it is necessary to be able to verify compliance with requirements at all stages of development, including the ability to change requirements if necessary or to incorporate new requirements into an existing system. Consider, for example, the following additional requirement.

3. If there is a passenger in the elevator car, the light must be turned on.

In the following sections, we show how this new additional requirement can be verified.

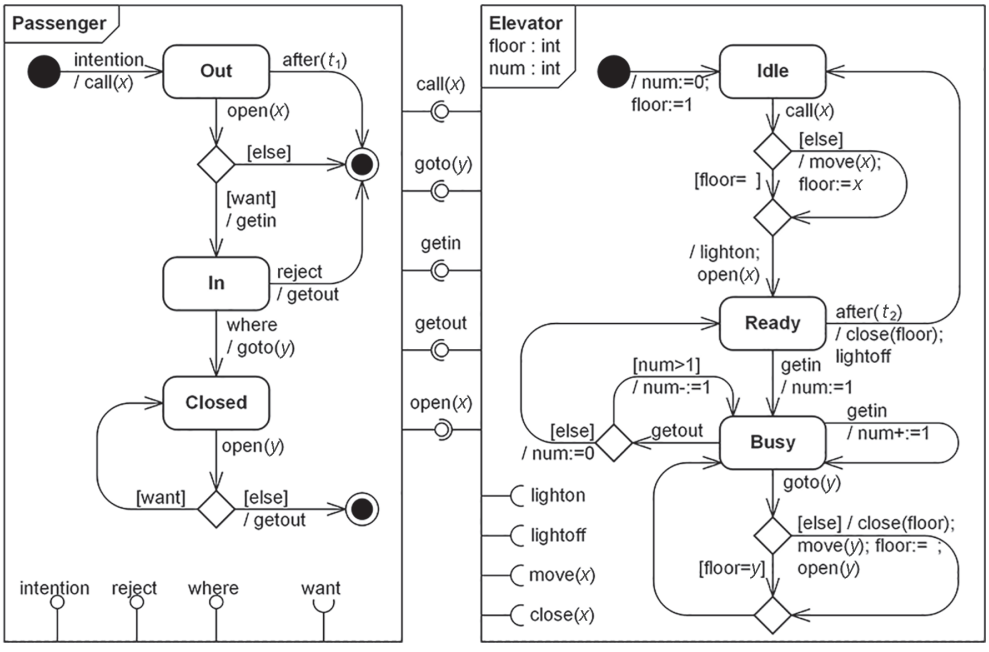

Fig. 1. Specification of elevator control system in CIAO language

Table 1. States, Events, Actions and Guard Conditions IDs and descriptions

ID	Name	Kind	Description
*e*1	intention	Event	The passenger had the idea to take the elevator from floor *x* to floor *y*
*e*2	reject	Event	The passenger changed his mind about taking the elevator
*e*3	where	Event	The passenger has decided which floor he should get to
*ez*1	call (*x*)	Event/Action	The passenger presses the elevator call button on floor *x*
*ez*2	goto (*y*)	Event/Action	The passenger in the elevator cabin presses the departure button to floor *y*
*ez*3	getin	Event/Action	The passenger enters the elevator
*ez*4	getout	Event/Action	The passenger exits the elevator
*ez*5	open (*x*)	Event/Action	Open elevator and shaft doors on floor *x*
*z*1	move	Action	Move the elevator to floor *x*
*z*2	lighton	Action	Switch the lights on
*z*3	lightoff	Action	Switch the lights off
*z*4	close	Action	Close the doors of the elevator car and the doors of the shaft on floor *x*
*c*1	want	Guard	The passenger wants to use the elevator
*c*2	floor = *x*	Guard	The elevator is on floor *x*
*s*1	Out	State	The passenger is outside the elevator car
*s*2	In	State	The passenger is in the elevator car, the doors are open, and the elevator is stopped
*s*3	Closed	State	The passenger is in the elevator car, the doors are closed, and the elevator is moving
*s*4	Idle	State	The elevator is idle, the doors are closed, and the lights are off
*s*5	Ready	State	The elevator is ready for service, the doors are open, and the lights are on
*s*6	Busy	State	The elevator is serving a passenger, the doors are closed, the lights are on, and the elevator is moving

The description of the semantics for reactive systems

The basis of the proposed verification methods is the use of the technique of formal description of languages.

In these notations and conventions, elementary actions are symbols of some alphabet (in this example, the symbols of the alphabet are listed in the first column ID of Table 1). Thus, the program execution protocol is a word in a given alphabet, and the entire set of protocols, that is, the semantics of the program, is a language over this alphabet.

In terms of formal languages, the requirements for the system are expressed as some statements about the structure of the words of the language that define the semantics. It is most convenient to set the structure of words using a regular expression, and then checking compliance with the requirements is reduced to the problem of parsing [14].

Let us explain what has been said with an example. In the notation of Table 1, we introduce the event/action class *zz*, which includes all actions except *z*2, *z*3, *z*4, and *ez*5. Then, requirement 3 is written as the following regular expression (where * denotes the Kleene iteration, as usual).

$$(zz)^*, z2, ez5, ((zz)^*, z4, (zz)^*, ez5, (zz)^*)^*, z4, z3. \quad (1)$$

In fact, requirement 3 means that the actions of turning on/off the light and entering/exiting the passenger must always be performed strictly in the specified sequence relative to each other and nothing else.

The fulfillment of the requirements described in this way is easily established by the algorithm for checking that each instance of the program execution protocol conforms to the regular language specified by the regular expression (1). Therefore, we assume that the requirements for the system are specified by a set of regular expressions, possibly with guard conditions, and the system itself is specified by a set of transition graphs in the CIAO language. The arcs of this transition graph are marked by events/actions which are considered as symbols of regular expressions.

Algorithm for automatic verification of semantics

Verification of compliance with the requirements for a given system in the CIAO language is carried out in three stages.

At the first stage, according to the given transition graphs of interacting automata objects and link scheme, a single graph is built, which is similar to a language source graph [15]. This source graph actually defines the semantics of the program, so we call it the semantic graph. The nodes in this graph correspond to events/actions, and the arcs can be labeled with guard conditions.

The graph turns out to be single since the action in one graph of transitions through the corresponding interface is unified with the event in another graph of transitions. The paths in the constructed graph correspond to the system execution protocols, that is, the words of the language of the semantics being verified. Fig. 2 shows a semantic graph for an elevator control system.

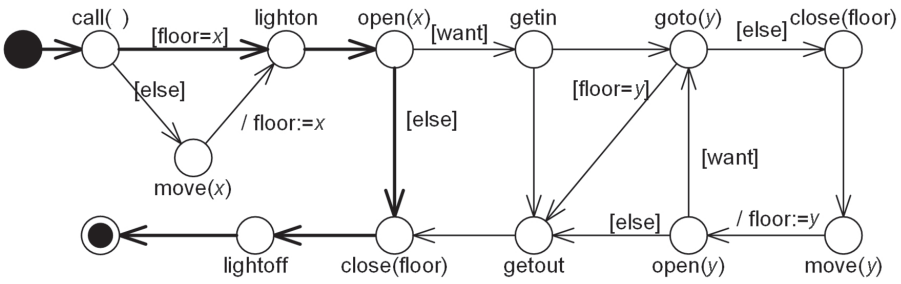

Fig. 2. Semantic graph of the elevator control system

In the second stage, conditional regular expressions from the graph that describe the language of the semantics of the system being verified are constructed. If the language of semantics turns out to be regular (which is often encountered in practice, for example, in communication protocols [16]), then the proposed methods turn out to be universal and allow one to extract a regular expression describing the semantics of a program directly from a given program in the CIAO language automatically.

If the semantic language of the program being checked is not regular, then it is impossible to obtain a single regular expression, but it is possible to obtain a certain set of expressions provided by guard conditions. We can say that the behavior of the program as a whole can be described as a set of several descriptions of the behavior of the program in special "modes", each mode being characterized by its own guard condition. The division of the program into different modes of operation is not automatic, but is determined by the programmer in the CIAO language when setting guard conditions. In this example, the following conditional regular expression is obtained, matching the bold path in the semantic graph in Fig. 2.

$$ez1, [c2], z2, ez5, [!c1], z4, z3. \qquad (2)$$

At the third stage, it is necessary to match the requirements given by the templates with the regular expressions obtained at the second stage. The matching consists of checking that the received regular expressions describing the semantics (paths from the initial node to the final one in the semantic graph) really have the structure prescribed by the requirements templates. For example, it is easy to check that the resulting regular expression (2) is indeed a special case of pattern (1), which means that the introduced requirement 3 is satisfied.

Discussion and conclusion

Thus, if the terms of reference for a responsible reactive system are described in terms of acceptable and unacceptable sequences of elementary actions using conditional regular expressions, i.e., the desired and undesirable behavior is indicated, then the proposed methods allow, without testing, a mathematically strict check of the compliance of the developed system in the CIAO language to the requirements specified automatically.

Last but not least, this is the positioning of the proposed semantic verification method by constructing a semantic graph among other verification methods. When comparing verification methods, it is necessary to consider two significant factors that affect the theoretical significance and practical applicability of the methods. First, what is the expressive power of the formalism used for the specification? The more powerful the formalism used; more complex requirements may be specified in the specification. Second, what is the expressive power of the programming language being used? The more powerful the language used, the wider the class of automatically checked programs.

In his outstanding book [17], Dijkstra developed the method of predicate transformers which is in fact a full-scale verification. This method is, in a certain sense, extremely general, since Dijkstra's Guarded Command Language is Turing complete, and the first-order predicate calculus language is sufficient in all reasonable cases, according to David Hilbert. However, subsequent studies have shown that the method of predicate transformers cannot be completely automated because the verification problem turns out to be algorithmically unsolvable if the power of the specification language is not limited and the power of the programming language is not limited.

The next step was the development of a family of methods for model checking [2], among which, as applied to automaton programs, work [5] stands out. Model checking allows automation by using temporal logic languages for the specification and application of automaton models as programs to be verified. These restrictions make it possible to narrow the verification problem to an algorithmically solvable one. However, in the model checking method, both the class of specifications and the class of automaton models are still very wide, and therefore, the automatic verification methods turn out to be computationally laborious.

In the proposed verification method, the specification formalism is narrowed down to a well-studied class of regular languages, and the programming language is narrowed down to the CIAO language, programs in which have a very simple and predefined structure. Due to this narrowing, it was possible to construct an efficient algorithm for automatic verification. In fact, this algorithm automatically checks whether the program performs some actions in a given order or does not perform some other actions in a forbidden order. No more, but no less. If we carefully structure the required behavior into elementary actions and translate informal requirements into conditional

regular expressions, then in many practical cases the proposed limited means will be sufficient for effective verification. The main intellectual effort in automatic programming according to this method is required in the formalization of informal specifications, in full accordance with a recent letter from Prof. A.A. Shalyto [18].

References

1. Harel D., Pnueli D. On the development of reactive systems. *Logics and Models of Concurrent Systems*. Berlin, Heidelberg, Springer, 1985, pp. 477–498. https://doi.org/10.1007/978-3-642-82453-1_17
2. Karpov Iu.G. *Model Checking. Verification of Parallel and Distributed Software Systems*. St. Petersburg, BHV-Petersburg Publ., 2010, 560 p. (in Russian)
3. Sommerville I. *Software Engineering*. 10th ed. Boston, Pearson, 2020.
4. Hinchey M., Coyle L. Evolving critical systems: a research agenda for computer-based systems. *Proc. of the 17th IEEE International Conference and Workshops on Engineering of Computer Based Systems*, 2010, pp. 430–435. https://doi.org/10.1109/ECBS.2010.56
5. Velder S.E., Lukin M.A., Shalyto A.A., Iaminov B.R. *Automata-Based Program Verification*. St. Petersburg, Nauka Publ., 2011, 244 p. (in Russian)
6. Fedorchenko L., Baranov S. Equivalent transformations and regularization in context-free grammars. *Cybernetics and Information Technologies*, 2015, vol. 14, no. 4, pp. 29–44. https://doi.org/10.1515/cait-2014-0003
7. Novikov F.A., Afanasieva I.V. Cooperative interaction of automata objects. *Information and Control Systems*, 2016, no. 6, pp. 50–64. (in Russian). https://doi.org/10.15217/issn1684-8853.2016.6.50
8. Afanasieva I., Novikov F., Fedorchenko L. Methodology for development of event-driven software systems using ciao specification language. *SPIIRAS Proceedings*, 2020, no. 19, no. 3, pp. 481–514. (in Russian). https://doi.org/10.15622/sp.2020.19.3.1
9. Aho A.V., Lam M.S., Sethi R., Ullman J.D. *Compilers: Principles, Techniques, and Tools*. 2nd ed. Boston, Pearson/Addison-Wesley, 2007, 1009 p.
10. Afanasieva I.V., Novikov F.A. Software architecture for optical detector systems. *Information and Control Systems*, 2016, no. 3, pp. 51–63. (in Russian). https://doi.org/10.15217/issn1684-8853.2016.3.51
11. Novikov F.A., Ivanov D.Iu. *UML Modeling. Theory, Practice, Video Course*. St. Petersburg, Professional'naja literature Publ., 2010, 649 p.
12. Knuth D.E. *The Art of Computer Programming, V. 1. Fundamental Algorithms*. 3rd ed. Addison-Wesley Professional, 1997, 672 p.
13. Polikarpova N.I., Shalyto A.A. *Automata-Based Programming*. St. Petersburg, Piter Publ., 2011, 176 p. (in Russian)
14. Fan W., Li J., Ma S., Tang N., Wu Y., Wu Y. Graph pattern matching: From intractable to polynomial time. *Proceedings of the VLDB Endowment*, 2010, vol. 3, no. 1–2, pp. 264–275. https://doi.org/10.14778/1920841.1920878
15. Avdoshin S.M., Nabebin A.A. *Discrete Mathematics. Formal Logic Systems and Languages*. Moscow, DMK Press Publ., 2018, 390 p. (in Russian)
16. Levonevskiy D., Novikov F., Fedorchenko L., Afanasieva I. Verification of internet protocol properties using cooperating automaton objects. *Proc. of the 12th International Conference on Security of Information and Networks (SIN'19)*, 2019, pp. 1–4. https://doi.org/10.1145/3357613.3357639
17. Dijkstra E.W. A *Discipline of Programming*. 3rd ed. Englewood Cliffs, N.J., Prentice Hall, 1976, 217 p.
18. Shalyto A.A. Validation of state machine specifications. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2023, vol. 23, no. 2, pp. 436–438. (in Russian). https://doi.org/10.17586/2226-1494-2023-23-2-436-438

Литература

1. Harel D., Pnueli D. On the development of reactive systems // Logics and Models of Concurrent Systems. Berlin, Heidelberg: Springer, 1985. P. 477–498. https://doi.org/10.1007/978-3-642-82453-1_17
2. Карпов Ю.Г. Model Checking. Верификация параллельных и распределенных программных систем. СПб.: БХВ-Петербург, 2010. 560 с.
3. Sommerville I. Software Engineering / 10th ed. Boston: Pearson, 2020.
4. Hinchey M., Coyle L. Evolving critical systems: a research agenda for computer-based systems // Proc. of the 17th IEEE International Conference and Workshops on Engineering of Computer Based Systems. 2010. P. 430–435. https://doi.org/10.1109/ECBS.2010.56
5. Вельдер С.Э., Лукин М.А., Шалыто А.А., Яминов Б.Р. Верификация автоматных программ. СПб.: Наука, 2011. 244 с.
6. Fedorchenko L., Baranov S. Equivalent transformations and regularization in context-free grammars // Cybernetics and Information Technologies. 2015. V. 14. N 4. P. 29–44. https://doi.org/10.1515/cait-2014-0003
7. Новиков Ф.А., Афанасьева И.В. Кооперативное взаимодействие автоматных объектов // Информационно-управляющие системы. 2016. № 6. С. 50–64. https://doi.org/10.15217/issn1684-8853.2016.6.50
8. Афанасьева И.В., Новиков Ф.А., Федорченко Л.Н. Методика построения событийно-управляемых программных систем с использованием языка спецификации CIAO // Труды СПИИРАН. 2020. Т. 19. № 3. С. 481–514. https://doi.org/10.15622/sp.2020.19.3.1
9. Aho A.V., Lam M.S., Sethi R., Ullman J.D. Compilers: Principles, Techniques, and Tools / 2nd ed. Boston: Pearson/Addison-Wesley, 2007. 1009 p.
10. Афанасьева И.В., Новиков Ф.А. Архитектура программного обеспечения систем оптической регистрации // Информационно-управляющие системы. 2016. № 3. С. 51–63. https://doi.org/10.15217/issn1684-8853.2016.3.51
11. Новиков Ф.А., Иванов Д.Ю. Моделирование на UML. Теория, практика, видеокурс. СПб.: Профессиональная литература, 2010. 640 с.
12. Knuth D.E. The Art of Computer Programming, V. 1. Fundamental Algorithms / 3rd ed. Addison-Wesley Professional, 1997. 672 p.
13. Поликарпова Н.И., Шалыто А.А. Автоматное программирование. СПб.: Питер, 2011. 176 с.
14. Fan W., Li J., Ma S., Tang N., Wu Y., Wu Y. Graph pattern matching: From intractable to polynomial time // Proceedings of the VLDB Endowment. 2010. V. 3. N 1–2. P. 264–275. https://doi.org/10.14778/1920841.1920878
15. Авдошин С.М., Набебин А.А. Дискретная математика. Формально-логические системы и языки. М.: ДМК Пресс, 2018. 390 с.
16. Levonevskiy D., Novikov F., Fedorchenko L., Afanasieva I. Verification of internet protocol properties using cooperating automaton objects // Proc. of the 12th International Conference on Security of Information and Networks (SIN'19). 2019. P. 1–4. https://doi.org/10.1145/3357613.3357639
17. Dijkstra E.W. A Discipline of Programming / 3rd ed. Englewood Cliffs, N.J.: Prentice Hall, 1976. 217 p.
18. Шалыто А.А. Валидация автоматных спецификаций // Научно-технический вестник информационных технологий, механики и оптики. 2023. Т. 23. № 2. С. 436–438. https://doi.org/10.17586/2226-1494-2023-23-2-436-438

Authors

Irina V. Afanasieva — PhD, Head of Laboratory, Special Astrophysical Observatory of the Russian Academy of Sciences (SAO RAS), Nizhny Arkhyz, 369167, Russian Federation, sc 57210431774, https://orcid.org/0000-0003-4225-4124, riv615@gmail.com

Fedor A. Novikov — D.Sc., Senior Researcher, Professor, Peter the Great St. Petersburg Polytechnic University (SPbPU), Saint Petersburg, 195251, Russian Federation; Professor, Alferov Academic University, Saint Peterburg, 194021, Russian Federation, sc 16441904500, https://orcid.org/0000-0003-4450-0173, fedornovikov51@gmail.com

Ludmila N. Fedorchenko — PhD, Senior Researcher, St. Petersburg Federal Research Center of the Russian Academy of Sciences, Saint Petersburg, 199178, Russian Federation, sc 36561350100, https://orcid.org/0000-0002-4008-9316, lnf@iias.spb.su

Received 02.05.2023
Approved after reviewing 16.06.2023
Accepted 26.07.2023

Авторы

Афанасьева Ирина Викторовна — кандидат технических наук, заведующий лабораторией, Специальная астрофизическая обсерватория Российской академии наук, Нижний Архыз, 369167, Российская Федерация, sc 57210431774, https://orcid.org/0000-0003-4225-4124, riv615@gmail.com

Новиков Федор Александрович — доктор технических наук, старший научный сотрудник, профессор, Санкт-Петербургский политехнический университет Петра Великого, Санкт-Петербург, 195251, Российская Федерация; профессор, Санкт-Петербургский национальный исследовательский Академический университет имени Ж.И. Алфёрова, Российской академии наук, Санкт-Петербург, 194021, Российская Федерация, sc 16441904500, https://orcid.org/0000-0003-4450-0173, fedornovikov51@gmail.com

Федорченко Людмила Николаевна — кандидат технических наук, старший научный сотрудник, Санкт-Петербургский Федеральный исследовательский центр Российской академии наук, Санкт-Петербург, 199178, Российская Федерация, sc 36561350100, https://orcid.org/0000-0002-4008-9316, lnf@iias.spb.su

Статья поступила в редакцию 02.05.2023
Одобрена после рецензирования 16.06.2023
Принята к печати 26.07.2023

Работа доступна по лицензии
Creative Commons
«Attribution-NonCommercial»