DOI: 10.18137/RNU.V9187.25.03.P.97 УДК 004.005.4

**Киреев Андрей Павлович**
старший научный сотрудник, Военно-космическая академия имени А.Ф. Можайского Министерства обороны Российской Федерации, Санкт-Петербург. SPIN-код: 4707-9500, Author ID: 796874
Электронный адрес: vka@mil.ru
**Andrey P. Kireev**
Senior Research Officer, Mozhaisky Military Aerospace Academy of the Ministry of Defense of the Russian Federation, St. Petersburg. SPIN-code: 4707-9500, Author ID: 796874
E-mail address: vka@mil.ru

**Шаров Сергей Алексеевич**
научный сотрудник, Военно-космическая академия имени А.Ф. Можайского Министерства обороны Российской Федерации, Санкт-Петербург. SPIN-код: 9260-4311, Author ID: 1236090
Электронный адрес: vka@mil.ru
**Sergey A. Sharov**
Research Officer, Mozhaisky Military Aerospace Academy of the Ministry of Defense of the Russian Federation, St. Petersburg. SPIN-code: 9260-4311, Author ID: 1236090
E-mail address: vka@mil.ru

# СПОСОБ ВЕРИФИКАЦИИ ИСХОДНОГО КОДА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ БОРТОВОЙ АППАРАТУРЫ КОСМИЧЕСКОГО АППАРАТА С ПРИМЕНЕНИЕМ ТЕМПОРАЛЬНОЙ ЛОГИКИ

**Аннотация.** В статье проведен обзор различных способов верификации программного обеспечения с применением методов дедуктивного доказательства и проверки модели. Приведен анализ применения метода проверки модели для верификации программного обеспечения на ранней стадии процесса разработки программного обеспечения с целью повышения качества. Цель исследования – разработка способа верификации исходного кода программного обеспечения бортовой аппаратуры космического аппарата с применением темпоральной логики. В результате исследований установлено, что существующие способы верификации программного обеспечения бортовой аппаратуры не полностью охватывают вопросы проверки асинхронных или многопоточных версий программного обеспечения. С целью разрешения указанной проблемы авторами предложен способ верификации данного класса программного обеспечения с использованием темпоральной логики, приводятся ограничения для основных функциональных программных блоков с использованием формул линейной темпоральной логики. Большое значение для космических аппаратов имеет надежность и качество программного обеспечения. Дальнейшее эволюционное развитие способов верификации создаваемых систем возможно за счет расширения применения моделей верификации для различных классов программ бортовой аппаратуры, что обеспечит повышение качества создаваемого программного обеспечения.

***Ключевые слова:*** методы формальной верификации, программное обеспечение бортовой аппаратуры, технологии верификации, темпоральная логика, исходный код программного обеспечения.

***Для цитирования:*** *Киреев А.П., Шаров С.А.* Способ верификации исходного кода программного обеспечения бортовой аппаратуры космического аппарата с применением темпоральной логики // Вестник Российского нового университета. Серия: Сложные системы: модели, анализ, управление. 2025. № 3. С. 97 – 105. DOI: 10.18137/RNU.V9187.25.03.P.97

Вестник Российского нового университета
Серия «Сложные системы: модели, анализ и управление». 2025. № 3

# VERIFICATION OF THE SOFTWARE SOURCE CODE OF THE SPACECRAFT ONBOARD EQUIPMENT USING TEMPORAL LOGIC

**Abstract.** The article proposes a method for verifying the source code of the spacecraft onboard equipment software based on the mathematical apparatus of temporal logic. An overview of various software verification methods using deductive proof and model checking is provided. An analysis of the use of the model checking method for software verification at an early stage of the software development process in order to improve quality is given. The aim of the study is to develop a method for verifying the source code of the spacecraft onboard equipment software using temporal logic. Based on the research results, it was found that the existing methods for verifying the onboard equipment software do not fully cover the issues of checking asynchronous or multithreaded versions of software. In order to solve this problem, the authors propose a method for verifying this class of software using temporal logic, and provide restrictions for the main functional software blocks using linear temporal logic formulas. Reliability and quality of software are of great importance for spacecraft. Further evolutionary development of verification methods for the created systems is possible due to the expansion of the use of verification models for various classes of onboard equipment programs, which will improve the quality of the created software.

***Keywords:*** formal verification methods, onboard equipment software, verification technologies, temporal logic, software source code.

***For citation:*** Kireev A.P., Sharov S.A. (2025) Verification of the software source code of the spacecraft onboard equipment using temporal logic. *Vestnik of Russian New University. Series: Complex Systems: Models, analysis, management*. No. 3. Pp. 97 – 105. DOI: 10.18137/RNU.V9187.25.03.P.97 (In Russian).

## *Введение*

При разработке современного программного обеспечения (далее – ПО) для космических аппаратов (далее – КА) с учетом размеров комплексов бортовой аппаратуры (далее – БА) и количества существенных деталей возможность подключать программные средства автоматизации наталкивается на разнородность и неструктурированность информации. Естественным шагом по преодолению указанной проблемы является формализация информации, переведение её в унифицированный машинный вид, что позволяет автоматизировать её обработку. Помимо традиционных методов отработки комплексов бортового оборудования космических аппаратов на наземных эксплуатационных стендах, для проведения тестирования программного обеспечения широко применяются формальные методы [1, с. 105]. Их целью является спецификация, разработка и верификация различных свойств программного обеспечения КА. При этом применяются математические методы, и поведение системы определяется с помощью формальных спецификаций [2–6]. Дальнейшее расширение проекта поддерживается на основе формальных моделей, обеспечивая тем самым соответствие свойств разрабатываемой системы требованиям, определяя её качество и надежность.

Для ранней диагностики неисправностей бортовой аппаратуры КА, возникающих при функционировании ПО, необходимо применение современных научных методов выявления ошибок. *Цель* статьи – разработка способа верификации исходного кода программного обеспечения бортовой аппаратуры КА с применением научно-методического аппарата линейной темпоральной логики.

### *Особенности методов верификации исходного кода программного обеспечения с применением темпоральной логики*

Одной из задач методов верификации является формальный анализ исходного кода программного обеспечения БА КА. Как видно из Рисунка 1, верификацию исходного кода можно разделить на определение соответствия исходного кода требованиям, отслеживаемость между исходным кодом и моделями поведения и статический анализ исходного кода.

При этом следует, что методы доказательства теорем и проверка модели для своего применения предлагают наличие спецификаций. Для применения метода доказательства теоремы спецификации должны быть извлечены как предшествующие и постусловия с преобразованием в аннотации программы на языке программирования Си. На конечном этапе метода применяются дедуктивные доказательства для проверки используемых аннотаций. Наиболее развитыми инструментами доказательства теорем, которые обычно используются в отрасли, являются VCC [7; 8] и Frama-C[1] [9].

Другой метод верификации основан на проверке модели. Построив формальную модель из исходного кода и используя темпоральную логику (LTL, CTL и др.) для представления спецификаций требований, инструменты проверки моделей, определяют контрпример (путь исполнения системы), когда требования системы не выполняются [10]. Существует множество инструментов проверки модели, таких как SPIN[2], NuSMV[3]. При построении процесса разработки на базе инструмента NuSMV возможно прямое преобразование проверенных формальных моделей в исходный код.

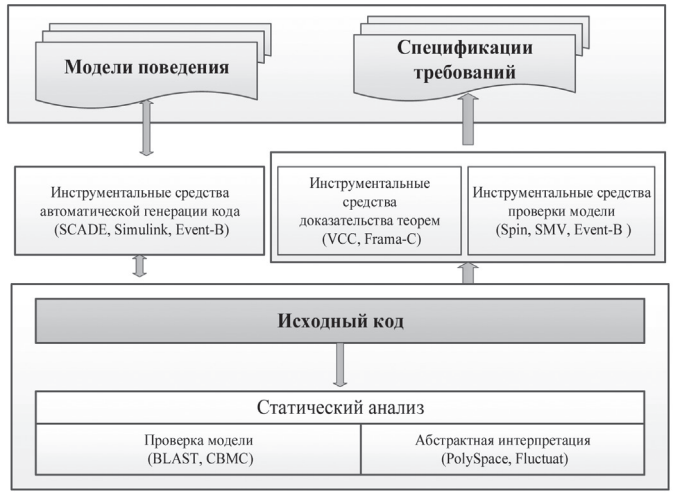


**Рисунок 1.** Методы формального анализа исходного кода программного обеспечения бортовой аппаратуры

*Источник:* здесь и далее рисунки выполнены авторами.

[1] A platform to make your C code safer and more secure // Frama-C. URL: http://frama-c.com (дата обращения: 15.04.2021).
[2] Verifying Multi-threaded Software with Spin. URL: http://spinroot.com. (дата обращения: 24.06.2025).
[3] NuSMV: A new symbolic model checker. URL: http://nusmv.fbk.eu (дата обращения: 15.04.2021).

Вестник Российского нового университета
Серия «Сложные системы: модели, анализ и управление». 2025. № 3

Темпоральная логика фокусируется на соблюдении последовательности и времени событий и может описывать широкий спектр свойств, таких как безопасность, достижимость, справедливость и свойства реального времени. Для формальных спецификаций, написанных с использованием темпоральной логики, модель ПО должна быть представлена соответствующим формальным языком, таким как Promela, BDD[4][11] и др.

Кроме того, цикл разработки программного обеспечения может быть значительно сокращен с помощью средств автоматической генерации кода. Например, инструменты SCADE от Esterel Technologies и Simulink от Mathworks имеют встроенные генераторы кода, которые могут преобразовывать модели поведения в эквивалентный код языка Си. Код, сгенерированный из формальной модели, обеспечивает отслеживаемость от исходного кода к модели поведения.

С этой целью анализируется граф управления и поток данных в сочетании с абстрактной интерпретацией и методами символического выполнения, чтобы обеспечить проверку переполнения данных, ошибки деления на ноль, доступ к массиву за пределами границ, ошибки времени выполнения в исходном коде, а также для реализации точного определения расположения ошибок.

Наиболее часто используемые средства статического анализа исходного кода включают PolySpace[5], Fluctuat (основаны на абстрактной интерпретации), SLAM, BLAST [12] и CBMC (основаны на проверке модели) и др.

Рассмотрим применение программных средств верификации исходного кода программного обеспечения, основанных на методе проверки модели.

### *Математическая постановка задачи верификации исходного кода программного обеспечения бортовой аппаратуры*

В качестве исходных данных в статье используется исходный код программного обеспечения, разработанный на языке программирования Си.

Для проверки исполнения принятых свойств ПО бортовой аппаратуры задаются формулы линейной темпоральной логики LTL. В случае проверки допустимых границ массива из адресной области процесса необходимо убедиться, что при любом его функционировании значения границ массива не будут превышены.

Формула темпоральной логики LTL, выражающая данный инвариант, представлена выражением

$$G\left(\neg array\_round\_1 \vee \neg array\_round\_2\right), \quad (1)$$

где $G$ – модальный оператор ‘глобально, все время’; *array_round_1* – первое условие для границы многомерного массива; *array_round_2* – второе условие для границы многомерного массива.

Логический инвариант проверки выполнения ограничения инициализации указателя на данные перед обращением функции обработки представлен выражением

$$\left(\neg function\_load\right) \cup \left(pointer\_data\_init \wedge \neg function\_load\right), \quad (2)$$

[4] Seshia S.A. EECS 219C: Formal Methods Binary Decision Diagrams (BDDs). EECS, UC Berkeley. URL: https://people.eecs.berkeley.edu/~sseshia/219c/lectures/BinaryDecisionDiagrams.pdf (дата обращения: 24.06.2025).

[5] Polyspace. Making Critical Code Safe and Secure. URL: https://www.mathworks.com/products/polyspace.html (дата обращения: 15.04.2021).

где $U$ – модальный оператор ‘с тех пор, как’; *function_load* – начало выполнения процедуры загрузки данных; *pointer_data_init* – условие инициализации указателя на данные.

Логический инвариант проверки освобождения мьютекса при функционировании конкурирующих процессов представлен выражением

$$G\left(mutex\_free \rightarrow G\, process\_1\_run\right) \vee G\left(mutex\_free \rightarrow G\, process\_2\_run\right), \quad (3)$$

где *mutex_free* – условие освобождения мьютекса; *process_1_run* – начало выполнения первого процесса; *process_2_run* – начало выполнения второго процесса.

Проверка получения сообщения путем обработки при функционировании двух асинхронных процессов информационного протокола представлена следующим выражением:

$$F\, process\_rec\_message \rightarrow \left(\neg process\_send\_message\right) U process\_rec\_message, \quad (4)$$

где $F$ – модальный оператор ‘когда-нибудь в будущем’; *process_rec_message* – начало процесса обработки сообщений; *process_send_message* – начало процесса отправки сообщений.

Для доказательства выполнения логических инвариантов при различных сценариях функционирования ПО используется метод верификации. Он позволяет определить ошибки функционирования программного обеспечения и путь исполнения, на котором не выполняются заданные пользователем свойства. При этом в процессе верификации анализируются все пути функционирования программной многопоточной системы.

На основе математической постановки задачи разработан способ верификации исходного кода программного обеспечения бортовой аппаратуры с использованием линейной темпоральной логики.

### *Способ верификации исходного кода программного обеспечения бортовой аппаратуры*

Существует множество методов оценки качества кода и уменьшения количества остаточных ошибок ПО: ручное пошаговое выполнение кода, экспертная оценка, статический анализ исходного кода, модульное и интеграционное тестирование. Данные методы справляются с задачами при тестировании последовательного кода – определяют ошибки, но они не совершенны для задач тестирования многопоточного кода. Иногда применяют термин «граница качества», который накладывается стандартными методами тестирования [2].

Существующие способы верификации программного обеспечения бортовой аппаратуры не полностью охватывают вопросы проверки асинхронных или многопоточных версий программного обеспечения. С данной целью необходимо применять специальные инструменты, предназначенные для проведения формальной проверки асинхронного или многопоточного программного обеспечения. В качестве примера такого инструмента для программного обеспечения, написанного на языке программирования Си, рассмотрим применение Modex[6].

Инструмент поддерживает три типа утверждений, которые пользователь может добавить в исходный код: базовые утверждения, утверждения ответа и утверждения приоритета.

*Базовое утверждение* имеет форму *assert (ВЫРАЖЕНИЕ)*. Проверяется, что выражение принимает значение *ИСТИНА* всякий раз, когда выполняется оператор утверждения во всех возможных системных исполнениях.

---

[6] Modex – a model extractor, to automatically extract Spin verification models from multi-threaded C code // GitHub. URL: https://github.com/nimble-code/Modex (дата обращения: 24.06.2025).

Вестник Российского нового университета
Серия «Сложные системы: модели, анализ и управление». 2025. № 3

*Утверждение ответа* имеет форму *assert_r (ВЫРАЖЕНИЕ)*. Проверяется, что выражение принимает значение *ИСТИНА* в течение конечного числа шагов после достижения утверждения во всех возможных исполнениях системы. Выражение должно стать истинным во всех системных исполнениях.

*Утверждение приоритета* имеет вид *assert_p (ВЫРАЖЕНИЕ 1, ВЫРАЖЕНИЕ 2)*. Проверяется, что выражение 1 истинно каждый раз, когда это утверждение выполняется, и остается истинным по крайней мере до тех пор, пока выражение 2 не станет истинным.

Тестовая программа, созданная для описанного выше инструмента, обычно определяет следующие сущности: процессы, взаимодействие процессов и объекты данных.

На Рисунке 2 представлен алгоритм верификации исходного кода программного обеспечения, поэтапную работу которого рассмотрим.

***На первом этапе*** работы алгоритма тестовой программы для исходного кода определяются исходные файлы на языке Си с помощью одной или нескольких команд *"%F"*. Далее определяются целевые процедуры, которые будут включены в качестве потоков асинхронного процесса, при помощи команды *"%X"*. Кроме того, требуются некоторые тестовые драйверы, а также минимальная инфраструктура, необходимая для обеспечения связи между процессами в тестовой программе. Реализация их производится либо в исходном коде на языке Си как функции Си, которые затем извлекаются как часть окончательной модели, или как процессы на языке Promela.

Далее рассматриваются определения стандартных версий таблиц сопоставления для сегментов *"%L"* в тестовой программе. В результате работы инструмента Modex создаются файлы с расширением *lut* или *nlut.* Данные файлы содержат полностью заполненные таблицы сопоставления для каждого целевого объекта (функции в исходном коде), на который была сделана ссылка в команде *"%X"* в тестовой программе с заполненными правилами преобразования по умолчанию.

***На следующем этапе*** определяется необходимое покрытие сегментов *"%L"* в тестовой программе. Переопределяются операции потока, экземпляры процессов (используются операции языка Promela) и все операции отправки и получения данных.

Затем создается полная модель проверки исходного кода программного обеспечения. Для этого применяется вызов команды *modex* без параметров:

*$ modex thread_sample.prx*

Далее из модели создается код проверки. В случае необходимости определяется необходимый перечень переменных с помощью командной опции сегментов *"%D"*.

***На заключительном этапе*** производится компиляция кода, сгенерированного верификатором Spin.

В результате функционирования модели верификатор будет создавать реальные отчеты об ошибках (нарушение различных видов утверждений, заданных для многопоточного приложения, а также дополнительных условий проверки: неинициализированные указатели, индексы массивов, пустое выполнение вычислительных циклов, наличие недостижимых состояний). По умолчанию поиск верификатора ограничен глубиной в десять тысяч шагов исполнения. При необходимости производится настройка количества шагов тестирования кода многопоточного приложения.

Таким образом, для проведения качественной диагностики неисправностей ПО, возникающих при функционировании бортовой аппаратуры КА, необходимо применение

современных методов выявления ошибок. Помимо традиционных методов тестирования исходного кода программного обеспечения требуется верификация исходного кода ПО БА с использованием моделей проверки (Model Checking), основанных на спецификациях темпоральных логик [2; 4; 5].

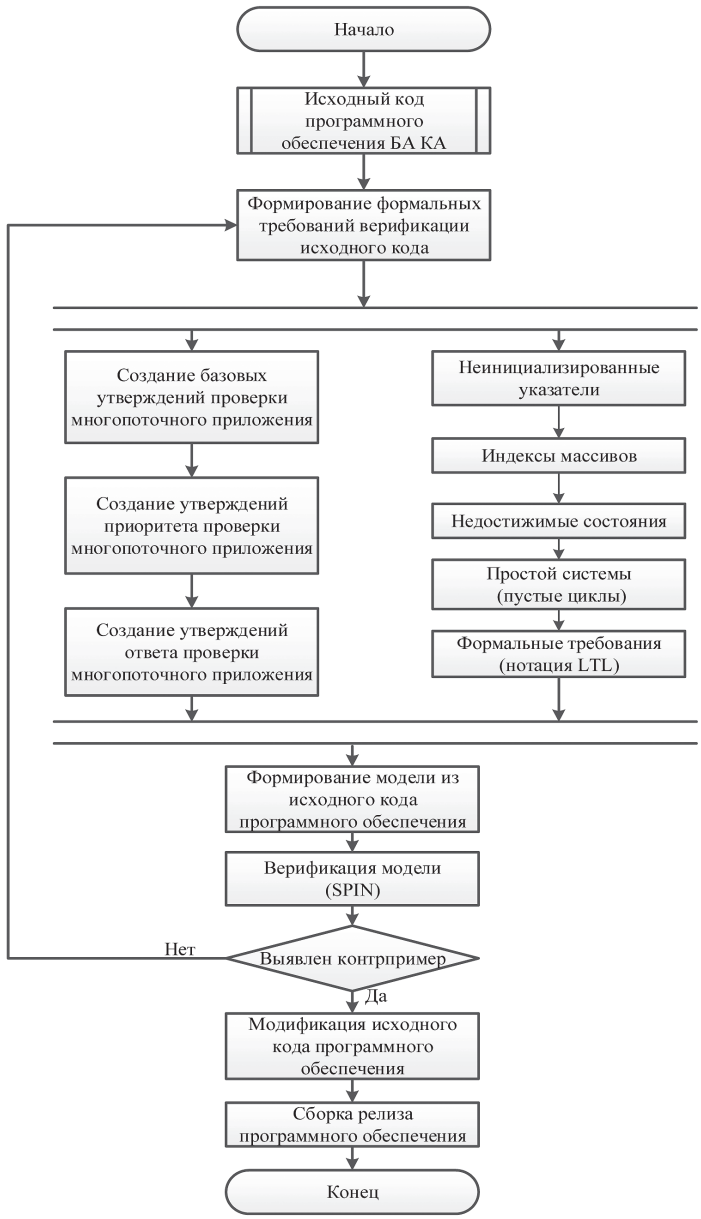


**Рисунок 2.** Алгоритм верификации исходного кода программного обеспечения бортовой аппаратуры

Вестник Российского нового университета
Серия «Сложные системы: модели, анализ и управление». 2025. № 3

### *Заключение*

Для реализации современных методов верификации исходного кода ПО бортовой аппаратуры КА необходимо использовать математический аппарат темпоральных логик LTL и инструментальные средства верификации [2; 6; 8].

Представлен разработанный способ верификации исходного кода ПО бортовой аппаратуры с использованием линейной темпоральной логики LTL.

Применение средств верификации на основе темпоральной логики позволит получить высокий уровень надежности и совместимости аппаратного и ПО бортовой аппаратуры КА, снизить временные и финансовые издержки на различных этапах ее жизненного цикла за счет раннего обнаружения ошибок ПО.

### Литература

1. *Киреев А.П., Шаров С.А.*, Средства верификации протокола информационного взаимодействия специального программного обеспечения бортовой аппаратуры космических аппаратов // Вестник Российского нового университета. Серия: Сложные системы: модели, анализ, управление. 2022. № 1. С. 104–111. DOI: 10.18137/RNU.V9187.22.01. EDN HQTZEG.

2. *Lee E.A., Seshia S.A.* Introduction to Embedded Systems – A Cyber-Physical Systems Approach. 2nd edition. MIT Press, 2017. 564 p. ISBN 978-0-262-53381-2.

3. *Cao Z., Lv W., Huan, Y., Shi J., Li Q.* Formal Analysis and Verification of Airborne Software Based on DO-333 // Electronics. 2020. Vol. 9. No. 2. Article no. 327. DOI: 10.3390/electronics9020327

4. *Кларк Э., Грамберг О., Пелед Д.* Верификация моделей программ / Пер. с англ. В. Захарова и др. М. : Изд-во Московского центра непрерывного математического образования, 2002. 416 с. ISBN 5-94057-054-2.

5. *Карпов Ю.Г.* Model Checking. Верификация параллельных и распределенных программных систем. СПб. : БХВ-Петербург, 2010. 551 с. ISBN 978-5-9775-0404-1.

6. *Spitzer C.R., Ferrell U., Ferrell T.* Digital avionics handbook. 3rd edition. CRC Press, 2014. 848 p. ISBN 1439868611.

7. *Dahlweid M., Moskal M., Santen T., Tobies S., Schulte W.* VCC: Contract-based modular verification of concurrent C // 2009 31st International Conference on Software Engineering – Companion Volume. Vancouver, BC, Canada, 2009. Pp. 429–430. DOI: 10.1109/ICSE-COMPANION.2009.5071046

8. *Moskal M., Schulte W., Cohen E., Hillebrand M.A., Tobies S.* Verifying C Programs: A VCC Tutorial. Working draft, version 0.2, May 8, 2012. URL: https://www.researchgate.net/publication/265810350_Verifying_C_Programs_A_VCC_Tutorial (дата обращения: 24.06.2025).

9. *Шапкин П.А.* Система верифицируемых спецификаций программных компонентов с поддержкой встраивания и извлечения // Программные продукты и системы. 2025. Т. 38. № 1. С. 65–76. DOI: 10.15827/0236-235X.149.065-076. EDN NVABDI.

10. *Гурин Р.Е., Рудаков И.В., Ребриков А.В.* Методы верификации программного обеспечения // Наука и Образование. МГТУ им. Н.Э. Баумана. 2015. № 10. С. 235–251. DOI: 10.7463/1015.0823129. EDN VDRGOX.

11. *Akers S.B.* Binary decision diagrams // IEEE Transactions on Computers. 1978. Vol. C-27. No. 6. Pp. 509–516. DOI: 10.1109/TC.1978.1675141

12. *Henzinger T.A., Jhala R., Majumdar R., Sutre G.* Software Verification with BLAST // Ball T., Rajamani S.K. (Eds) Model Checking Software. SPIN 2003. Lecture Notes in Computer Science. 2003. Vol. 2648. Springer, Berlin, Heidelberg. DOI: https://doi.org/10.1007/3-540-44829-2_17

**References**

1. Kireev A.P., Sharov S.A. (2022) Verification tools for the information interaction protocol of special software for onboard equipment of spacecraft. Vestnik of Russian New University. Series: Complex systems: models, analysis and control. No. 1. Pp. 104–111. DOI: 10.18137/RNU.V9187.22.01 (In Russian).
2. Lee E.A., Seshia S.A. (2017) Introduction to Embedded Systems – A Cyber-Physical Systems Approach. 2nd edition. MIT Press. 564 p. ISBN 978-0-262-53381-2.
3. Cao Z., Lv W., Huan, Y., Shi J., Li Q. (2020). Formal Analysis and Verification of Airborne Software Based on DO-333. *Electronics*. Vol. 9. No. 2. Article no. 327. DOI: 10.3390/electronics9020327
4. Clarke E., Grumberg O., Peled D. (1999) *Model checking*. MIT Press. 313 p. ISBN 0262032708. (Russian edition: transl. by V. Zakharov. Moscow : Moscow Center for Continuous Mathematical Education Publ., 2002).
5. Karpov Yu.G. (2010) Model Shecking. Verifikatsiya parallel'nykh i raspredelennykh programmnykh sistem [Model Checking. Verification of parallel and distributed software systems]. St. Petersburg : BHV-Petersburg. 551 p. ISBN 978-5-9775-0404-1. (In Russian).
6. Spitzer C.R., Ferrell U., Ferrell T. (2014) *Digital Avionics Handbook*. 3rd edition. CRC Press. 848 p. ISBN 1439868611.
7. Dahlweid M., Moskal M., Santen T., Tobies S., Schulte W. VCC: Contract-based modular verification of concurrent C. In: *2009 31st International Conference on Software Engineering – Companion Volume*. Vancouver, BC, Canada, 2009. Pp. 429–430. DOI: 10.1109/ICSE-COMPANION.2009.5071046
8. Moskal M., Schulte W., Cohen E., Hillebrand M.A., Tobies S. (2012) *Verifying C Programs: A VCC Tutorial. Working draft, version 0.2*. May 8, 2012. URL: https://www.researchgate.net/publication/265810350_Verifying_C_Programs_A_VCC_Tutorial (accessed 24.06.2025).
9. Shapkin P.A. (2025) A System of Verifiable Specifications of Software Components with Embedding and Extraction Support. *Software & Systems*. Vol. 38. No. 1. Pp. 65–76. DOI: 10.15827/0236-235X.149.065-076 (In Russian).
10. Gurin R.E., Rudakov I.V., Rebrikov A.V. (2015) Methods of Software Verification. *Science and Education of the Bauman MSTU*. No. 10. Pp. 235–251. DOI: 10.7463/1015.0823129 (In Russian).
11. Akers S.B. (1978) Binary decision diagrams. In: *IEEE Transactions on Computers*. Vol. C-27. No. 6. Pp. 509–516. DOI: 10.1109/TC.1978.1675141
12. Henzinger T.A., Jhala R., Majumdar R., Sutre G. (2003). Software Verification with BLAST. In: Ball T., Rajamani S.K. (Eds) *Model Checking Software. SPIN 2003. Lecture Notes in Computer Science.* Vol. 2648. Springer, Berlin, Heidelberg. DOI: https://doi.org/10.1007/3-540-44829-2_17

| | |
|---|---|
| Поступила в редакцию: 01.08.2025 | Received: 01.08.2025 |
| Поступила после рецензирования: 02.09.2025 | Revised: 02.09.2025 |
| Принята к публикации: 15.09.2025 | Accepted: 15.09.2025 |