DOI: 10.15514/ISPRAS-2019-31(5)-13

Методы и средства разработки автоматизированных информационных систем на основе онтологии «Управление качеством программно-технических комплексов»

А.В.Самонов, ORCID: 0000-0002-0390-4481 <a.samonov@mail.ru>

Военно-космическая академия им. А.Ф. Можайского,
197088, Россия, Санкт-Петербург, ул. Ждановская, д.13

Аннотация. Представлены методы и средства реализации программно-управляемого процесса разработки и верификации формальных моделей требований и проектных решений автоматизированных информационных систем критической информационной инфраструктуры. Процессы выполняются в единой для всех его участников модельно-языковой и информационно-программной среде автоматизированным способом на основе предметно-ориентированной онтологий. Онтологии описывают процессы управления качеством программно-технических комплексов на этапах обоснования требований и проектирования систем, разработаны с помощью конструкций и механизмов языков моделирования и проектирования SysML, FUML, OCL, а также математического аппарата сетей Петри, временных автоматов и временных логик. Для валидации и верификации комплекса требований и проектных решений разработаны алгоритмы построения и анализа трассы выполнения модели в среде виртуальной машины VM FUML. Предложены способы интеграции и использования специализированных средств верификации CPN Tools, Rodin, SPIN и Modelica для автоматизированного тестирования моделей комплекса требований и проектных решений. Данный комплекс обеспечивает более эффективное взаимодействие заказчика и исполнителя как при разработке требований, так и при проектировании системы, обнаружение и устранение дефектов посредством реализации автоматизированных процедур верификации, валидации и коррекции. Применение данного подхода позволит повысить качество требований и проектных решений, а также улучшить экономические показатели путем снижения финансовых и временных затрат, связанных с выполнением дополнительных работ как в случае обнаружения дефектов, так и при изменении требований или условий эксплуатации.

Ключевые слова: автоматизированные информационные системы; валидация и верификация; временные автоматы; проектирование и моделирование; сети Петри; функциональные и эксплуатационные требования.

Для цитирования: Самонов А.В. Методы и средства разработки автоматизированных информационных систем на основе онтологии «Управление качеством программно-технических комплексов». Труды ИСП РАН, том 31, вып. 5, 2019 г., стр. 165-182. DOI: 10.15514/ISPRAS-2019-31(5)-13

Methods and Means for Automated Information Systems Development based on Ontology «Software and Hardware Complexes Quality Management»

A.V. Samonov, ORCID: 0000-0002-0390-4481 <a.samonov@mail.ru>

A.F. Mozhaisky Military Space Academy, Saint-Petersburg, 197198, Russia

Abstract. The paper presents development and verification methods and means of requirements and design solutions formal models. They are intended to create complex critical automated information systems in a same model-language and information-software environment for all its participants. The development and verification processes are carried out in an automated way on the basis of subject-oriented ontologies. Ontologies describe the quality management processes of software and hardware complexes at the stages of requirements justification and system design. They are developing by means modeling and design languages SysML, FUML, OCL structures and mechanisms, the Petri nets mathematical apparatus, time automata and time logics. In order to execute of validation and verification for complex of requirements and design solutions, construction and model execution route analysis algorithms in the VM FUML virtual machine environment are developed. Integration and use methods for specialized verification tools CPN Tools, Rodin, SPIN and Modelica as means to automated testing of complex requirements and design solutions models are proposed. This complex provides more effective interaction between the customer and the contractor both in the development of requirements and in the design of the system, along with this, detection and provides limination of defects through the automated verification, validation and correction procedures implementation. This approach application will improve the quality of requirements and design solutions, as well as improve economic performance by reducing the financial and time costs, which associated with the implementation of additional work in the case of defects, and when changing requirements or operating conditions.

Keywords: automated control systems; validation and verification; time machines; design and modeling; Petri nets; functional and operational requirements

For citation: Samonov A.V. Methods and Means for Automated Information Systems Development based on Ontology «Software and Hardware Complexes Quality Management». Trudy ISP RAN/Proc. ISP RAS, vol. 31, issue 5, 2019, pp. 165-182 (in Russian). DOI: 10.15514/ISPRAS-2019-31(5)-13

1. Введение

Основными компонентами критической информационной инфраструктуры (КИИ) государства являются информационно-телекоммуникационные сети и автоматизированные информационные системы (АИС), предназначенные для решения задач государственного управления, обеспечения обороноспособности, безопасности и правопорядка. К системам данного класса, являющихся сложными программно-техническими системами (СПТС), предъявляются повышенные требования к надежности, оперативности и устойчивости функционирования. Разработка АИС КИИ представляет собой сложную и ресурсоемкую задачу, результат решения которой не всегда удовлетворяет заданным требованиям и укладывается в рамки выделенных временных и финансовых ресурсов. Данный факт убедительно подтверждают ежегодно публикуемые отчеты американской исследовательской компании The Standish Group. Так, в отчете за 2018 год утверждается, что доля успешных проектов составляет около 30%, 20% проектов оказались проваленными полностью, остальные 50% проектов столкнулись с проблемами, из-за которых итоговый бюджет превысил первоначальный в среднем в 1,5 раза, сроки выросли почти в 2 раза, около 50% функций не соответствовали заявленным требованиям [1].

Одной из основных причин такого состояния дел является отсутствие у участников процесса создания АИС КИИ (пользователя, проектировщика и программиста-разработчика) единой терминологической и концептуальной базы, адекватного логико-

математического аппарата и эффективных средств поддержки процессов формирования и анализа двух важнейших артефактов жизненного цикла (ЖЦ) АИС КИИ: комплекса требований и проектных решений. В данной статьей предложены и описаны модели, методы и средства разработки единой модельно-языковой и информационно-программной среды, а также методика и алгоритмы реализации в ней программно-управляемого процесса разработки и верификации комплекса требований и проектных решений. Реализация данного подхода предполагает разработку и использование онтологии, описывающей процессы управления качеством программно-технических комплексов на различных этапах ЖЦ систем.

2. Анализ современных технологий, методов и средств промышленной разработки СПТС. Проблемы и пути решения

Исключительная актуальность совершенствования технологий и средств разработки надежного программного обеспечения для критически важных автоматизированных систем обусловила огромное внимание и усилия, предпринимаемые международными и национальными организациями, научным и профессиональным сообществом, коллективами разработчиков и отдельными исследователями для решения имеющихся в данной области проблем. Наиболее системными и практичными являются методические документы и спецификации, разработанные под эгидой организации OMG (Object Management Group), с которой сотрудничают около 800 научно-исследовательских организаций (DISA, INCOSE, NIST и др.) и промышленных компаний (AT&T, IBM, Oracle, Microsoft, Cisco Systems, NASA и др.). В настоящее время на сайте OMG опубликовано более 230 методических документов и спецификаций. С точки зрения рассматриваемых здесь вопросов наиболее важными из них являются спецификации: MOF (Meta Object Facility), UML (Unified Modeling Language), XMI (XML Metadata Interchange), SysML (System Modeling Language), OCL (Object Constraint Language), UTP (UML Testing Profile), ALF (Action Language for Foundational UML), FUML (Semantics of a Foundational Subset for Executable UML Models), ReqIF (Requirements Interchange Format).

Теоретической основой современных технологий разработки СПТС являются концепции и методы модельно-ориентированной системной и программной инженерии (МОСиПИ) (Model-based systems engineering, MBSE), которая включает три составных компонента: разработка на основе моделей (Model driven development, MDD), архитектура на основе моделей (Model driven architecture, MDA), тестирование на основе моделей (Model based testing, MBT) [2, 3]. Технологическую основу составляют такие технологии разработки СПТС как Microsoft Solution Framework (MSF), Oracle Method, Rational Unified Process (RUP), SADT (IDEFx). Наиболее известными средствами реализации этих концепций и технологий являются IBM Rhapsody, Sparx Enterprise Arhitect, Modelio, Eclipse Papyrus и некоторые другие. Данные системы обеспечивают разработчиков СПТС автоматизированными средствами анализа, специфицирования, проектирования и тестирования систем, состоящих из аппаратных средств, программного обеспечения, данных, персонала, процедур, средств и других искусственных и природных систем. Для разработки таких средств поддержки используются языки визуального моделирования и проектирования SysML, FUML, OCL, ALF, ArhiMate, AADL. Для контроля качества артефактов ЖЦ систем, получаемых в процессе проектирования и разработки СПТС, применяются различные средства верификации и валидации, например, CPN Tools, Rodin, SPIN, Modelica.

В нашей стране активные исследования в этой области ведутся в таких организациях как ИСП РАН, ВМК МГУ имени М. В. Ломоносова, Санкт-Петербургском ГУ, Новосибирском ГТУ, ВКА имени А.Ф. Можайского и др. В результате этих исследований были созданы: система поддержки проектирования и верификации комплексов бортового

авиационного оборудования MASIW, технология тестирования программных интерфейсов – UniTESK, статический анализатор Svace и др. Ниже представлен краткий обзор основных направлений исследований и работ по развитию и совершенствованию технологий создания СПТС.

Теоретические основы проектирования и верификации СПТС на основе теоретико-категорного похода к метапрограммированию изложены в публикациях ведущего научного сотрудника ИПУ РАН Ковалева С.П. [4, 5]. В них представлены способы применения теории категорий для решения проблемы представления разнородных технологий программной инженерии в единой форме, удобной для их интеграции и координации в рамках общего цикла проектирования программных систем. Особое внимание уделяется современным технологиям, таким, как разработка, управляемая моделями (model checking), и аспектно-ориентированное программирование (aspect-oriented programming), для которых построены универсальные теоретико-категорные семантические модели.

Одним из современных средств описания архитектуры программно-аппаратных систем является Architecture Analysis & Design Language (AADL) [6]. На его основе была разработана и в настоящее время активно используется система поддержки проектирования и верификации комплексов бортового авиационного оборудования MASIW, разработанная ИСП РАН совместно с ГосНИИАС в рамках государственной программы по развитию Интегрированной Модульной Авионики (ИМА). При разработке MASIW были использованы библиотеки и средства Eclipse Modeling Framework, Graphical Editing Framework, Eclipse Team Providing, SVN Team Provider, GIT Team Provider. Как отмечено в статье [7], инструментальные средства MASIW позволяют решать следующие задачи:

- создание, редактирование и управление моделями программно-аппаратных комплексов (ПАК) на языке AADL;
- анализ моделей на предмет достаточности аппаратных ресурсов и согласованности интерфейсов, оценки характеристик проектируемых сетей передачи данных, построенных в соответствии со стандартом AFDX (Avionics Full-Duplex Switched Ethernet);
- распределение функциональных приложений по вычислительным модулям с учетом ограничений ресурсов аппаратной платформы и требований к надежности и безопасности ПАК;
- генерация программного кода и конфигурационных данных для ОС РВ VxWorks653 и оконечных устройств сети AFDX.

Пример использования специального расширения языка AADL – Error Model Annex (EMA) и инструмента MASIW для моделирования и анализа безопасности проектируемых ПАК представлен в статье [8]. С помощью EMA создается модель, в которой для каждого компонента ПАК разрабатывается конечный автомат, состояниями которого являются штатные и нештатные, в том числе опасные и отказные состояния данного компонента. Влияние сбоев компонентов системы на другие компоненты описывается посредством указания логических условий распространения ошибок между различными типами компонентов в различных состояниях с учетом вероятностей их возникновения.

Для анализа рисков используются следующие алгоритмы: анализ дерева неисправностей, анализ видов и последствий отказов, марковский анализ. Реализация описанного в данной статье подхода позволяет выявлять и устранять критичные для безопасности создаваемой системы дефекты проектных решений уже на этапе ее проектирования.

В статье [9] представлен обзор следующих методов автоматической генерации тестов для верификации программного обеспечения:

- структурного тестирования с помощью символического выполнения (structural testing using symbolic execution);
- тестирования на основе моделей (model-based testing);
- комбинаторного тестирования (combinatorial testing);
- выборочного тестирования (random testing);
- поиска на основе тестирования (search-based testing).

В статье [10] представлен автоматизированный способ построения UML диаграмм последовательностей из описания UML диаграмм вариантов использования и диаграмм классов. Для его реализации используется язык ATL и разработанные авторами статьи метамодели диаграмм вариантов использования, классов и последовательностей, а также правила получения из первых двух диаграмм третьей. Результатом преобразования является диаграмма последовательностей в формате XMI, которая затем преобразуется в формат XSLT для отображения диаграммы последовательностей в среде графического редактора для просмотра, анализа и доработки. Недостатком предложенного алгоритма является отсутствие возможности автоматической коррекции исходных моделей в случае внесения изменений в диаграмму последовательностей. Это обусловлено тем, что преобразования с помощью языка ATL являются однонаправленными, т.е. работают с моделями источника только для чтения и создают целевые модели только для записи.

В статье [11] описан метод автоматической генерации программного кода на основе проекта программы, представленного на языке ALF. Особого внимания заслуживает описание концептуальной схемы механизма генерации программного кода с помощью средств редактора ALF и используемых для этого правил в нотации расширенной формы Бэкуса-Наура (Enhanced-BackusNaur-Form (EBNF) – язык ATL). Авторы отмечают следующие достоинства инструмента трансформации модели архитектуры языка ATL: возможность описания как декларативных, так и императивных языковых конструкций, наличие средств объединения модулей, позволяющих создавать и повторно использовать наборы правил преобразования. Результатом является программный код на языке Java, соответствующий метамодели Modisco Java.

В статье [12] дано описание двух методов реализации автоматической проверки нагруженных систем реального времени с использованием сценариев. В первом система моделируется как сеть временных автоматов (ВА). Во втором – как набор диаграмм динамических последовательностей (LSCs) и требований в виде отдельной анализируемой диаграммы LSC. Авторы статьи разработали временные (темпоральные) расширения для подмножества ядра языка LSC и определили его семантику на основе трассировки. Анализируемая диаграмма LSC транслируется в ее поведенческий эквивалент в нотации диаграммы ВА. Верификация корректности модели осуществляется посредством моделирования диаграммы ВА в режиме реального времени с использованием аппарата темпоральной логики CTL (Computational Tree Logic) с последующим сравнением полученного результата с эталоном. Оба метода реализованы с помощью средств инструмента UPPAAL.

В работе [13] предлагается метод построения разверток для безопасных консервативных вложенных сетей Петри, основанный на трансляции таких сетей в классические сети Петри. Для классических сетей Петри затем применяются стандартные методы построения разверток. Также в работе обсуждаются сравнительные достоинства двух подходов.

Одним из первых этапов разработки СПТС является их декомпозиция на функциональные компоненты. Исследованию и совершенствованию методов декомпозиции программных систем посвящено множество работ.

В статье [14] представлены два подхода решения данной задачи, которые были использованы при дедуктивной верификации формальной спецификации мандатной

сущностно-ролевой модели управления доступом и информационными потоками в ОС семейства Linux (МРОСЛ ДП-модель) - с использованием формального метода Event-B и техники пошагового уточнения.

В статье [15] предложены методические рекомендации и комплекс средств реализации процесса сквозного контроля качества СПТС на всех этапах их жизненного цикла: обоснования и формализации требований, проектирования, реализации, испытаний. Для реализации данного процесса разработана единая модельно-языковая и информационно-программная среда, основанная на онтологии автоматизируемой предметной области, использующая языки визуального проектирования и моделирования fUML, OCL, ALF, а также автоматизированные средства валидации и верификации CPN Tools, Rodin, SPIN.

В настоящее время ведутся активные исследования и работы по созданию предметно-ориентированных языков моделирования (xDSML, eXecutable Domain-Specific Modeling Languages), использующие и развивающие ключевую концепцию МОСиПИ – метамоделирование. Результатом таких работ, в частности, является разработанный группой исследователей из Франции и Германии программный комплекс GEMOC Studio [16]. Данный комплекс предназначен для создания xDSML на основе языков моделирования SysML/FUML, разработки исполняемых моделей на этом языке (xDSML), их валидации и верификации с помощью графического аниматора и интеллектуального механизма трассировки. С помощью данного комплекса можно осуществлять мониторинг состояния анализируемых моделей (переходов, событий, значений переменных) во время их выполнения в виртуальной машине.

В работе [17] дано описание используемой в данном средстве многомерной предметно-ориентированной метамодели трассировки, обеспечивающей более высокую производительность по сравнению со стандартной UML метамоделью за счет исключения из обработки избыточных данных.

В работе [18] для создания средств динамической верификации и валидации проектных поведенческих моделей предлагается использовать исполняемые предметно-ориентированные языки моделирования xDSMLs (Executable Domain-Specific Modeling Languages). Средства на их основе позволяют осуществлять мониторинг состояния анализируемых моделей (переходов, событий, значений переменных) во время их выполнения. Предложен новый генеративный подход, основанный на многомерной предметно-ориентированной метамодели трассировки, с помощью которого реализуется построение и управление трассами исполнения для моделей, соответствующих заданному xDSML. Как утверждают авторы работы, данный метод имеет более высокую производительность по сравнению со стандартной UML метамоделью, благодаря возможности исключать из обработки избыточные данные (например, анализируемые трассы) с помощью механизмов соответствующего xDSML.

Завершая анализ публикаций и представленных в них решений, можно сделать следующие выводы:

- основные усилия исследователей направлены на разработку методов и средств автоматизированной генерации и верификации программных реализаций СПТС, в меньшей степени – на автоматизацию разработки и верификации проектных решений и практически отсутствуют решения для автоматизированного формирования и верификации комплекса требований;
- в качестве основных математических моделей и построенных на их основе средств для автоматической верификации используются математический аппарат и алгоритмы анализа сетей Петри, темпоральные логики, исполняемые FUML и xUML модели, SMT/SAT решатели задач, а также языки моделирования AADL, UML, FUML, SysML, OCL и разработанные на их основе предметно-ориентированные языки xDSMLs.

Таким образом, следуя принципам и методологии модельно-ориентированного подхода, можно утверждать, что первоочередными задачами, которые необходимо решить для реализации программно-управляемого процесса разработки АИС КИИ, являются:

- построение единой модельно-языковой и информационно-программной среды разработки и верификации АИС КИИ;
- разработка алгоритмического, инструментального и методического обеспечения реализации программно-управляемого процесса разработки и верификации формальных моделей требований, архитектуры и реализации АИС КИИ.

Для построения единой модельно-языковой и информационно-программной среды разработки и верификации АИС КИИ предлагается использовать:

- языки моделирования UML, FUML, SysML, OCL и ALF;
- методы и средства разработки предметно-ориентированной онтологии АИС КИИ (ODM – Ontology Definition Metamodel, RDF – Resource Description Framework);
- библиотеки и программные продукты, реализованные в рамках проекта Eclipse: Eclipse Modeling Framework, Graphical Editing Framework, Papyrus, Modelio.

Выбор этих моделей, языков и средств обусловлен тем, что, во-первых, их развитие активно поддерживается со стороны ведущих предприятий разработчиков и организаций потребителей СПТС, во-вторых, как сами технологии, так и основанные на них средства являются открытыми и доступными для изучения, применения и совершенствования.

3. Модели, методы и средства построения единой модельно-языковой и информационно-программной среды для разработки комплекса требований и проектных решений

На основе проведенного анализа нормативно-методических документов, научно-технических публикаций и программных средств проектирования и разработки АИС КИИ для построения единой модельно-языковой и информационно-программной среды были выбраны следующие конструкции и механизмы языков визуального проектирования SysML, FUML и OCL:

- диаграммы пакетов (Package), внешнего и внутреннего представления блоков (Block Definition, Internal Block), активных и пассивных классов (Active Class, Passive Class) - для описания состава и структуры системы;
- диаграммы вариантов использования (UseCase), деятельности (Activity), состояний (State Machine) и взаимодействия (Sequience) - для описания поведения и способов взаимодействия ее компонентов;
- средства языка OCL – для описания требований к эксплуатационным характеристикам (производительности, надежности, защищенности и др.);
- отношения обобщения (generalization), агрегации (aggregation), композиции (composition), ассоциации (association), зависимости (dependency), представленные с помощью средств языка OCL - для описания связей между структурными, поведенческими, ограничительными и другими аспектами, элементами и свойствами системы.

Для валидации и верификации формальных моделей комплекса требований и проектных решений целесообразно использовать виртуальную машину *VM FUML*, а также такие инструменты как *CPN Tools* [19], *Rodin* [20], *SPIN* [21, 22]. На рис. 1 отражена предлагаемая схема (алгоритм) построения формальных моделей комплекса требований и проектных решений АИС КИИ.

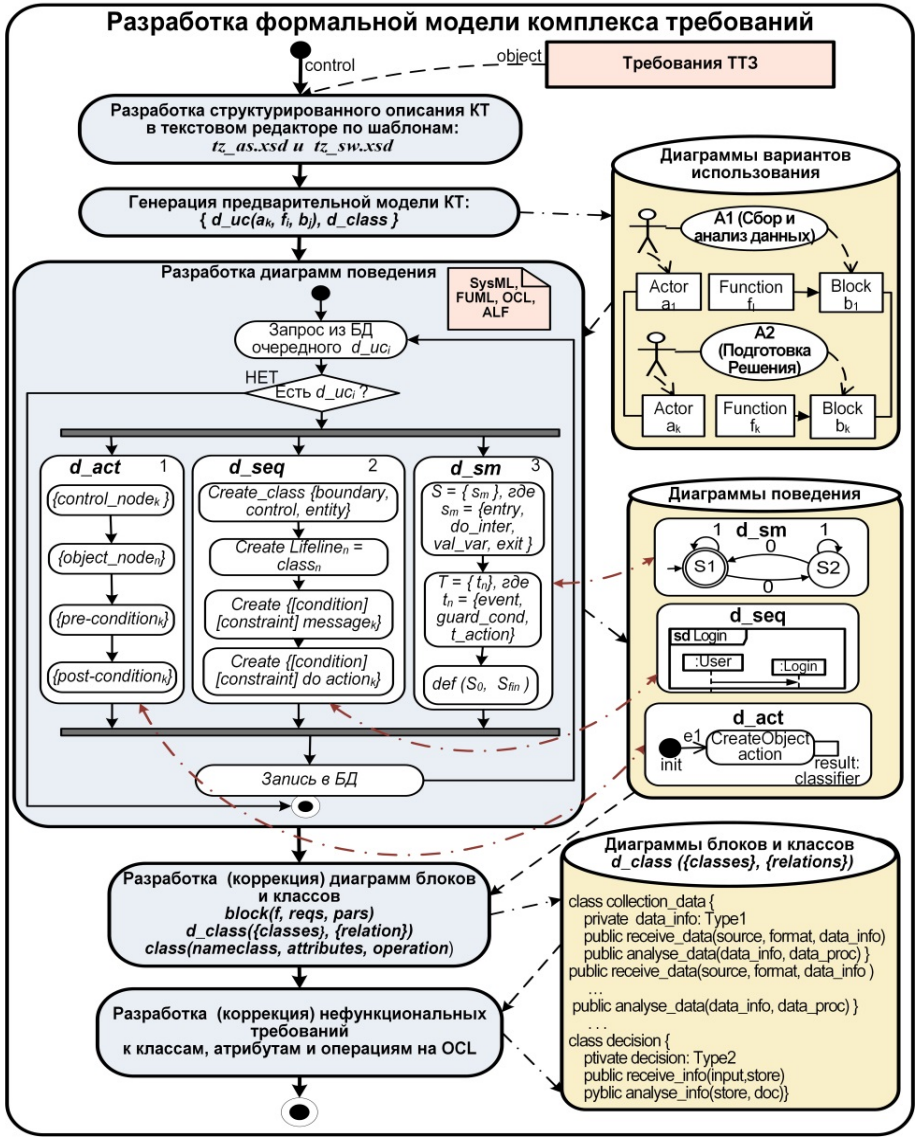

Рис. 1. Схема построения формальных моделей комплекса требований и проектных решений АИС КИИ

Fig. 1. Scheme for formal model construction of requirements and design complex for automated information systems

Построение формальных моделей требований и проектных решений в соответствии с выше представленной схемой заключается в выполнении следующих процедур:

- разработки структурированного описания комплекса требований (КТ) в текстовом редакторе на основе специальных шаблонов: tz_as.xsd и tz_sw.xsd;

- генерации предварительной модели КТ в виде совокупности предметно-ориентированных диаграмм вариантов использования;
- построения для каждой такой диаграммы диаграмм поведения (d_act, d_seq, d_sm), обеспечивающих более подробное и точное описание требований к порядку, способам и условиям реализации функционала системы;
- описания состава и структуры системы в виде совокупности диаграмм специализированных блоков и классов (d_blocks, d_class);
- определения нефункциональных (эксплуатационно-технических) требований к качеству реализации функционала, среде и условиям эксплуатации АИС.

В следующих разделах статьи описаны модели и средства реализации процедур автоматизированной разработки и верификации формальных моделей требований и проектных решений.

4. Методы и средства разработки и верификации формальной модели комплекса требований к АИС КИИ

Управление качеством АИС КИИ должно осуществляться на всех этапах их ЖЦ и представлять собой тесно связанную совокупность следующих процессов:

1) выбор и обоснование характеристик и показателей характеристик качества (ПХК), к которым следует предъявлять требования, и определение критериев, которым значения ПХК должны удовлетворять;
2) обеспечение требований к качеству АИС КИИ на этапе проектирования;
3) обеспечение требований к качеству АИС КИИ на этапе реализации;
4) измерение и оценивание ПХК на этапе приемо-сдаточных испытаний;
5) обеспечение требований к качеству АИС КИИ на этапе эксплуатации и сопровождения.

Для каждого из этих процессов необходимо разработать соответствующую онтологию. В данной статье основное внимание уделяется первым двум процессам и используемым для их реализации онтологиям.

Комплекс требований к АИС КИИ включает две основные группы требований: требования к ее функциональным возможностям и требования к качеству их реализации.

Для обеспечения автоматизированного построения формальных описаний требований в среде визуального моделирования были разработаны предметно-ориентированные шаблоны (стереотипы), представляющие собой расширения диаграмм вариантов использования (*UseCase*), деятельности (*Activity*), состояний (*State Machine*), взаимодействия (*Sequence*), внешнего (*block definition diagram*) и внутреннего (*internal block diagram*) представления компонентов системы, активных (*Active Class*) и пассивных (*Passive Class*) классов. Каждый *i*-й вариант использования представляет собой функциональное требование и описывается следующим образом:

$$uc_i = (nameuc_i,\ description_i,\ actor_k, subject_j,\ basic\_scenario_i,\ alter\_scenario_i).$$

Функциональные блоки системы (*subject*) описываются посредством соответствующих расширений диаграмм пакетов (*Package*), внешнего и внутреннего представления блоков (*Block Definition, Internal Block*). Активные классы (*Active Class*) описывают пользователей, системы и элементы внешнего окружения, которые могут инициировать какую-либо деятельность АИС.

Пассивные классы (*Passive Class*) описывают артефакты, создаваемые и используемые АИС во время функционирования (информационные, материальные, вычислительные, сетевые, людские ресурсы; решения, команды, сигналы, события, …).

$Active\ Class = < Actors, Systems, Envs >$.

Основной и альтернативные сценарии реализации функций представляются посредством специализированных активных и пассивных классов (*Active Class, Passive Class*). Для описания методов этих классов используются предметно-ориентированные диаграммы поведения, построенные с помощью механизма расширения из диаграмм деятельности (*Activity*), состояний (*State Machine*) и взаимодействия (*Sequence*).

Все АИС КИИ, с точки зрения особенностей алгоритма их функционирования, можно разделить на три группы. Первую группу образуют системы, выполняющие свои функции в соответствии с определенным алгоритмом, имеющим вход и выход. Во вторую группу входят реагирующие системы, выполняющие определенные операции и изменяющие свое состояние в зависимости от внешних воздействий. Третью группы образуют системы, взаимодействующие посредством обмена сообщениями.

Для формального описания функционирования систем первой группы наиболее адекватным средством являются диаграммы деятельности (*activity*), основанные на математическом аппарате сетей Петри (СП).

Классическая СП описывается следующей формулой:

$$CPN = (P, T, F, m_I),$$

где $P = \{p_1, p_2, \dots, p_n\}$ – множество мест;

$T = \{t_1, t_2, \dots, t_n\}$ – множество переходов, таких что $P \cap T = \emptyset$;

$F \subseteq P \times T \cup T \times P$ – матрица инцидентности;

$m_I : P \to N$ – начальная маркировка.

Основными свойствами СП, которые следует использовать для верификации формальных моделей требований и архитектуры, являются: живость и ограниченность. СП обладает свойством живости, если она обладает свойствами достижимости и покрываемости. Достижимость – это наличие путей перехода в заданные состояния из начального состояния $m \in [m_I\rangle$. Покрываемость – возможность перехода в заданные состояния из других состояний $m'' \in [m'\rangle$. Сеть Петри является k −ограниченной, если и только если все маркировки $m \in [m_I\rangle$, и для всех $p \in P$: $m(p) \leq k$. Обладающая данным свойством СП и, как следствие, описываемая с ее помощью система будет функционировать в определенных границах. Частным случаем свойства ограниченности является безопасность. Сеть Петри (P, T, F, m_I) является безопасной, если все маркировки $m \in [m_I\rangle$, и для всех $p \in P$: $m(p) \leq 1$.

Важным достоинством использования СП является наличие целого ряда автоматизированных средств верификации построенных с их помощью моделей. Одним из наиболее развитых и эффективных является *CPN Tools*, который и предлагается использовать в нашем случае для анализа формальных моделей требований и архитектуры АИС КИИ.

Формальная модель комплекса требований подвергается процедурам валидации и верификации. Процедура валидации заключается в оценивании комплекса требований на предмет его полноты и корректности и выполняется как программными средствами, так и неформальной экспертизой специалистов в данной предметной области.

При верификации проверяются такие свойства комплекса требований, как непротиворечивость, системность, неизбыточность, безопасность, живость, отсутствие взаимоблокировок, невыполнимых операций, зацикливаний.

Следующим шагом построения модели комплекса требований является разработка требований к качеству реализации каждой функции. Наиболее важными эксплуатационным характеристикам АИС КИИ являются: надежность, защищенность и производительность. Для реализации программно-управляемого процесса их описания разработана онтология «Требования к качеству реализации функциональных возможностей АИС КИИ» (рис.2).

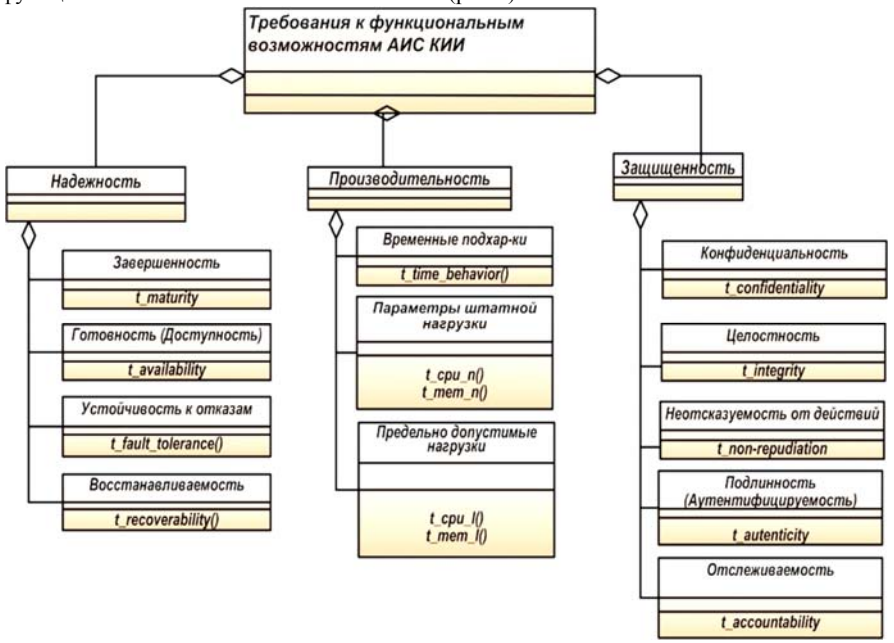

Рис. 2. Онтология «Требования к качеству реализации функциональных возможностей АИС КИИ»
Fig. 2. «Requirements to Realization Quality of Functional Abilities for automated information systems» Ontology

Для задания требований к характеристике качества «надежность» используются следующие ПХК и шаблоны (функции): *t_maturity* – для ПХК «завершенность», *v_fault tolerance* – для ПХК «отказоустойчивость», *v_recoverability* – для ПХК «восстанавливаемость», *t_availability* – для ПХК «готовность». Данные ПХК определяют требования к качеству реализации соответствующих функциональных возможностей АИС КИИ.

Для задания требований к характеристике качества «производительность» предлагается использовать характеристики времени выполнения функций в различных режимах работы, которые определяются такими параметрами как количество пользователей, объемы и темп запросов, сложность и ресурсоемкость алгоритмов решения задач и др. При этом необходимо проанализировать как минимум два режима: штатной и максимальной нагрузки.

Требования к эксплуатационным характеристикам (производительности, надежности, защищенности, совместимости и др.) описываются с помощью средств языка *OCL*. Для каждой из этих характеристик разработаны соответствующие шаблоны, которые используются в подключаемых к инструменту *Eclipse Papyrus* модулях (*plugins*) для реализации программно-управляемого процесса их формального описания и

верификации. Затем с помощью отношений обобщения (*generalization*), агрегации (*aggregation*), композиции (*composition*), зависимости (*dependency*), подчиненности (*subordinations*), дислокации (*dislocations*) и др. устанавливаются связи между структурными, поведенческими, ограничительными и другими аспектами и характеристиками модели, которые также представляются с помощью средств языка OCL.

Состав и структура комплекса моделей и программных средств, с помощью которых реализуются процедуры верификации и валидации формальной модели требований к АИС КИИ, представлены на рис. 3.

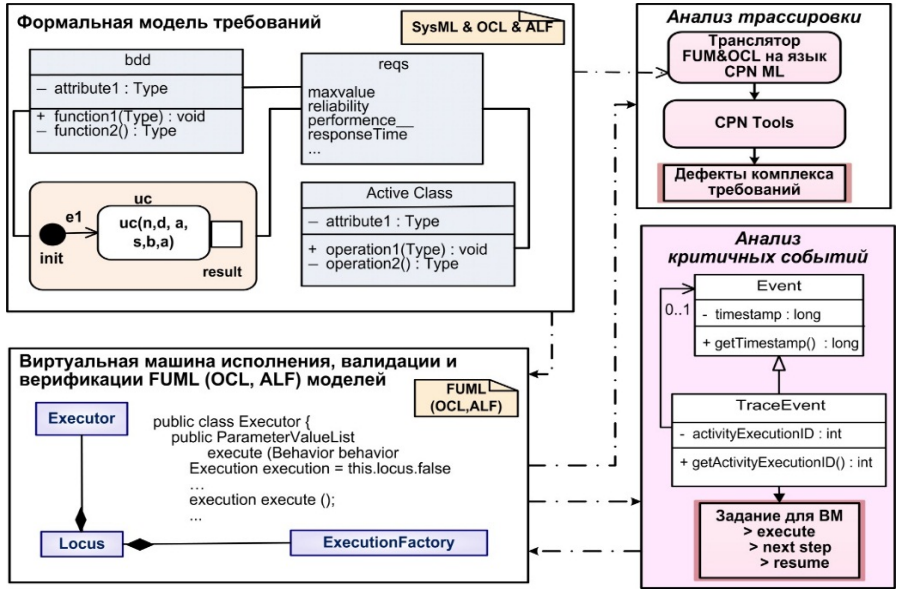

Рис. 3. Состав и структура комплекса моделей и программных средств реализации процедур верификации и валидации формальной модели требований к АИС КИИ»
Fig. 3. Set and Structure of Complex Models and Software for Formal Requirements Model Validation and Verification

Процедура тестирования комплекса требований включает:

- валидацию комплекса требований посредством исполнения его формальной модели в среде виртуальной машины VM FUML и анализа ее состояния в контрольных точках и при обработке критичных событий;
- анализ поведенческой части модели, представленной диаграммами активности, с помощью *CPN Tools* на предмет выявления тупиковых и мертвых состояний, стабильности, ограниченности и безопасности;
- анализ поведенческой части модели, представленной диаграммами состояний, с помощью инструмента *Rodin* на предмет выявления тупиковых и мертвых состояний, необрабатываемых стимулов и др.;
- верификацию качества реализации в диаграмме классов базовых механизмов объектно-ориентированной парадигмы (инкапсуляции, наследования, полиморфизма, абстракции, обмена сообщениями) посредством расчета и оценивания метрик, характеризующих эти механизмы.

5. Методы и средства разработки и верификации архитектуры и проектных решений

Проектирование архитектуры и проектных решений АИС КИИ осуществляется системным архитектором на основе разработанного и верифицированного на предыдущем этапе комплекса требований в той же модельно-языковой и информационно-программной среде.

Для этого ему предоставляются графические и табличные шаблоны проектирования структурных и поведенческих аспектов системы.

При разработке шаблонов для проектирования архитектуры и проектных решений АИС КИИ с помощью диаграмм активности были использованы раскрашенные временные сети Петри (РВСП), формальное описание которых представляется следующим выражением:

$$TPN \equiv < P, T, C, F, m_1, S, Z >,$$

где P, T, F и m_1– имеют тоже значение, что и для простых сетей Петри;

$C=\{c_1, c_2, \ldots, c_d\}$ – цвета или типы маркеров;

$S=(s_1, s_2, \ldots, s_n)$ – вектор временных задержек маркеров в позициях;

$Z = \{z_1, z_2, \ldots, z_r\}$ – вектор времени срабатывания разрешенных переходов.

При разработке шаблонов для проектирования архитектуры и проектных решений АИС КИИ с помощью диаграмм состояний были использованы математические модели временных автоматов (ВА) [23, 24], описываемые следующим выражением:

$$TA \equiv < S, C, D, R, f, s_0 >,$$ где:

S – конечное множество позиций автомата;

C – конечное множество локальных часов, значения которых возрастают синхронно с реальным временем и могут принимать значения из множества действительных чисел $\Re$;

D – конечное множество действий;

$R \subseteq S \times P \times D \times C \times S$ - множество переходов автомата;

$f: S \to P$ – функция, которая ставит в соответствие каждой позиции $s \in S$ некоторый предикат $p \in P$;

$s_0 \in S$ – начальная позиция автомата.

Формальная модель архитектуры и проектных решений должна удовлетворять следующим характеристикам качества:

- полнотой и корректностью реализации функциональных требований;
- полнотой и корректностью реализации эксплуатационных требований;
- согласованностью и непротиворечивостью всех диаграмм модели;
- отсутствием избыточности диаграмм и их атрибутов;
- отсутствием взаимоблокировок, невыполнимых операций и зацикливаний, которые могут привести к нарушению безопасности и живости системы.

Для разработки программных средств верификации проектных решений была разработана и использована онтология «Оценивание качества реализации АИС КИИ» (рис. 4).

Все требования к качеству АИС в данной модели сведены в 8 групп: функциональная пригодность, производительность, надежность, совместимость, удобство использования, удобство сопровождения, защищенность, переносимость. Функциональная пригодность включает три показателя качества характеристики (ПХК): корректность, полноту и целесообразность. Для расчета и оценивания

соответствия этих ПХК заданным требованиям используются функции: *v_functional_completeness, v_functional_correctness* и *v_functional_appropriateness*.

Для расчета и оценивания характеристики качества «надежность» используются следующие функции: *v_maturity* – для ПХК «завершенность», *v_availability* – для ПХК «готовность», *v_fault tolerance* – для ПХК «отказоустойчивость», *v_recoverability* – для ПХК «восстанавливаемость».

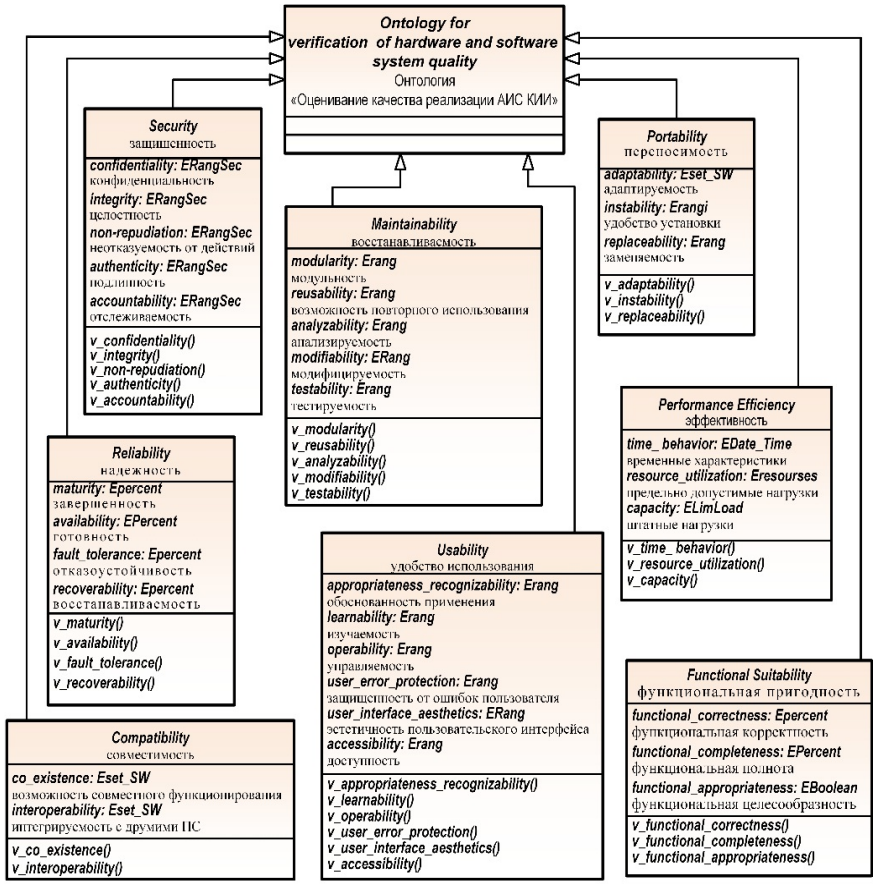

Рис. 4. Онтология «Оценивание качества реализации АИС КИИ»
Fig. 4. «Verification of hardware and software System Realization Quality» Ontology

Для оценивания характеристики «производительность» используются характеристики времени выполнения функций в двух основных режимах работы: штатной и максимальной нагрузки. При расчете должны учитываться доступные вычислительные, программные, сетевые и другие ресурсы.

Состав и структура комплекса программных средств, обеспечивающих проверку соответствия характеристик качества проекта АИС КИИ заданным требованиям, представлены на рис. 5.

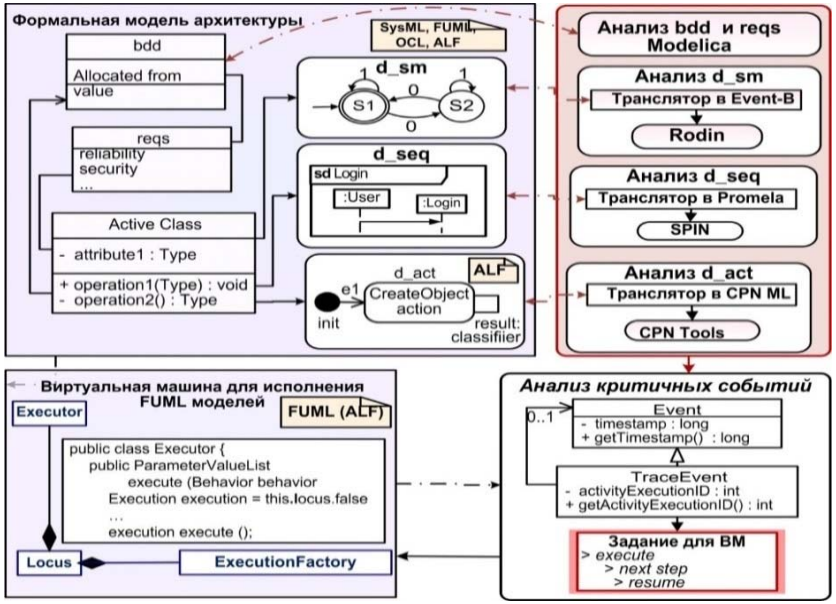

Рис. 5. Состав и структура комплекса программных средств, используемых для валидации и верификации проектных решений
Fig. 5. Set and Structure of Complex Software for Design Decisions, Validation and Verification

Для верификации технической части проектных решений предлагается использовать средство моделирования кибер-физических систем *OpenModelica* [25]. В случае обнаружения каких-либо дефектов проектные решения возвращаются архитекторам системы на доработку. Исправленный проект подвергается повторной процедуре валидации и верификации. В результате реализации данного последовательно-итерационного и программно-управляемого процесса будут разработаны проектные решения, в полной мере соответствующие предъявленным к АИС КИИ требованиям пользователей, условиям применения и нормативным документам.

6. Заключение

Представленные в статье методы и комплексы программных средств предназначены для реализации программно-управляемого процесса разработки, верификации и валидации АИС КИИ. Работа на всех этапах ЖЦ осуществляется в единой модельно-языковой и информационно-программной среде, построенной на основе технологий объектно-ориентированного анализа и проектирования, языков и средств визуального моделирования SysML, FUML, OCL. Их применение сокращает логические и семантические разрывы между представлениями о разрабатываемой системе у участников процесса создания АИС КИИ, существенно повышая эффективность их взаимодействия. Это приведет к повышению качества всех артефактов жизненного цикла разработки систем, и в первую очередь комплекса требований и проектных решений.

Реализация автоматизированных процедур верификации, валидации и коррекции комплекса требований и проектных решений с помощью таких средств как VM FUML, CPN Tools, SPIN и Rodin, позволит улучшить экономические показатели в части снижения

финансовых и временных затрат, связанных с выполнением дополнительных работ по внесению изменений, как в случае обнаружения каких-либо дефектов, так и при изменении самих требований или условий эксплуатации АИС КИИ.

Список литературы / References

[1]. The Standish Group report. URL: https://www.standishgroup.com/store/services/10-chaos-report-decision-latency-theory-2018-package.html, accessed 21.05.2019.

[2]. Systems Engineering and Software Engineering [online]. URL: https://www.sebokwiki.org/wiki/Systems_Engineering_and_Software_Engineering, accessed 25.05.2019.

[3]. Laura E. Hart. Introduction To Model-Based System Engineering (MBSE) and SysML [online]. URL: https://www.incose.org/docs/default-source/delaware-valley/mbse-overview-incose-30-july-2015.pdf, accessed 21.05.2019.

[4]. Ковалёв С.П. Теоретико-категорный подход к метапрограммированию. М.: ИПУ РАН, 2014, 112 стр./ S.P. Kovalev. Category-theoretic approach to metaprogramming. M.: ICS RAS, 2014, 112 p. (in Russian).

[5]. Ковалев С.П. Теоретико-категорный подход к проектированию программных систем. Фундаментальная и прикладная математика, том 19, вып. 3, 2014 г., стр. 111–170 / Kovalev S.P. Category-theoretic approach to the design of software systems. Fundamental and Applied Mathematics, vol. 19, issue 3, 2014, pp. 111-170 (in Russian).

[6]. Peter H. Feiler, David P. Gluch. Model-Based Engineering with AADL: An Introduction to the SAE Architecture Analysis & Design Language. Addison-Wesley Professional, 2012, 479 p.

[7]. Д.В. Буздалов, С.В. Зеленов, Е.В. Корныхин, А.К. Петренко, А.В. Страх, А.А. Угненко, А.В. Хорошилов. Инструментальные средства проектирования систем интегрированной модульной авионики. Труды ИСП РАН, том 26, вып. 1, 2014 г., стр. 201-230 / D.V. Buzdalov, S.V. Zelenov, E.V. Kornykhin, A.K. Petrenko, A.V. Strakh, A.A. Ugnenko, A.V. Khoroshilov. Design tools for integrated modular avionics systems. Trudy ISP RAN/Proc. ISP RAS, vol. 26, issue 1, 2014, pp. 201-230 (in Russian). DOI: 10.15514/ISPRAS-2014-26(1)-6.

[8]. Зеленов С.В., Зеленова С.А. Моделирование программно-аппаратных систем и анализ их безопасности. Труды ИСП РАН, том 29, вып. 5, 2017 г., стр. 257-282 / S.V Zelenov, S.A. Zelenova Modeling of software and hardware systems and analysis of their safety. Trudy ISP RAN/Proc. ISP RAS, vol. 29, vol. 5, 2017, pp. 257-282 (in Russian). DOI: 10.15514/ISPRAS-2017-29(5)-13

[9]. An Orchestrated Survey on Automated Software Test Case Generation. Journal of Systems and Software, vol. 86, issue 8, 2013, pp. 1978-2001.

[10]. Photchana Sawprakhon, Yachai Limpiyakorn. Sequence Diagram Generation with Model Transformation Technology. In Proc. of the International MultiConference of Engineers and Computer Scientists, 2014, pp. 584-589.

[11]. Thomas Buchmann and Alexander Rimer. Unifying Modeling and Programming with ALF. In Proc. of the Second International Conference on Advances and Trends in Software Engineering, 2016, pp. 10-15.

[12]. Li S., Balaguer S., David A. et al. Scenario-based verification of real-time systems using Uppaal. Formal Methods in System Design, vol. 37, Issue 2–3, 2010, pp 200–264.

[13]. Ермакова В.О., Ломазова И.А. Трансляция вложенных сетей Петри в классические сети Петри для верификации разверток. Труды ИСП РАН, том 28, вып, 4, 2016, стр. 115-136 / Ermakova V.O., Lomazova I.A. Translation of Nested Petri Nets into Petri Nets for Unfoldings Verification. Trudy ISP RAN/Proc. ISP RAS, vol. 28, issue 4, 2016, pp. 115-136 (in Russian). DOI: 10.15514/ISPRAS-2016-28(4)-7.

[14]. Девянин П.Н., Кулямин В.В., Петренко А.К., Хорошилов А.В., Щепетков И.В. Сравнение способов декомпозиции спецификаций на Event-B. Программирование, том 42, № 4, 2016 г., стр. 17-26 / Comparison of specification decomposition methods in Event-B Devyanin P.N., Kulyamin V.V., Petrenko A.K., Khoroshilov A.V., Shchepetkov I.V. Programming and Computer Software, vol. 42, № 4, 2016, pp. 198-205.

[15]. Самонов А.В., Самонова Г.Н. Методика и средства разработки и верификации формальных FUML моделей требований и архитектуры сложных программно-технических систем. Труды ИСП РАН, том 30, вып. 5, 2018, стр. 125-148 / Samonov A.V., Samonova G.N. Methodology and Tools for Development and Verification of formal fUML models of Requirements and Architecture

for complex software and hardware systems. Trudy ISP RAN/Proc. ISP RAS, 2018, vol. 30, issue 5, pp.125-148. DOI: 10.15514/ISPRAS-2018-30(5)-8.

[16]. Eclipse GEMOC Studio [online]. URL: https://projects.eclipse.org/projects/modeling.gemoc, accessed 20.06 2019.

[17]. Meyers B., Deshayes R., Lucio L., Syriani E., Vangheluwe H., Wimmer M. ProMoBox: A Framework for Generating Domain-Specific Property Languages. Lecture Notes in Computer Science, vol. 8706, 2014, pp. 1-20.

[18]. Bousse E., Mayerhofer T., Combemale B., Baudry B. Advanced and efficient execution trace management for executable domain-specific modeling languages. Software & Systems Modeling, vol. 18, issue 1, 2019, pp 385–421.

[19]. CPN Tools Homepage. CPN Tools is a tool for editing, simulating, and analyzing Colored Petri nets URL: http://cpntools.org/, accessed 27.05.2019.

[20]. Rodin. URL: http://www.event-b.org/, accessed 11.05.2019.

[21]. SPIN. URL: http://spinroot.com/spin/whatispin.html, accessed 11.06.2019.

[22]. Ю.Г. Карпов, И.В. Шомшина. Введение в язык Promela и систему комплексной верификации Spin: учебное пособие. Изд-во Политехнического ун-та, 2010, 110 стр. / Y.G. Karpov, I.V. Shamshina. Introduction to the language Promela and the verification system of the comprehensive Spin: a training manual. Publishing house of Polytechnic University, 2010, 110 p. (in Russian).

[23]. Timed Automata: Semantics, Algorithms and Tools. Johan Bengtsson and Wang Yi. Lecture Notes in Computer Science, vol. 3098, 2003, pp. 87-124.

[24]. Твардовский А.С., Лапутенко А.В. О возможностях автоматного описания параллельной композиции временных автоматов. Труды ИСП РАН, том 30, вып. 1, 2018 г., стр. 25-40. / A.S Twardowski., AV. Lopotenco. On the possibilities of automaton description of parallel composition of time automata. Trudy ISP RAN/Proc. ISP RAS, volume 30, vol. 1, 2018, pp. 25-40 (in Russian). DOI: 10.15514/ISPRAS-2018-30(1)-2.

[25]. Openmodelica. URL: https://www.openmodelica. org/, accessed 27.05.2019.

Информация об авторе / Information about author

Александр Валерьянович САМОНОВ – кандидат технических наук, доцент, старший научный сотрудник. Сфера научных интересов: системный анализ, информатика, математическое и имитационное моделирование и разработка сложных программно-технических систем, методы и средства обеспечения информационной безопасности.

Alexander Valerianovitch SAMONOV – PhD in Technical Sciences, Associated Professor, Senior Researcher. Research interests: system analysis, computer science, system and software engineering, methods and means of information security.