Комплекс программных средств разработки и верификации требований и проектных решений АСУ объектами транспортной инфраструктуры

А.В. Баев, канд. техн. наук А.В. Самонов
Военно-космическая академия имени А.Ф.Можайского
Санкт-Петербург, Россия
a.baih@mail.ru, a.samonov@mail.ru

Аннотация. **Представлены состав, структура и способы применения комплекса программных средств, обеспечивающего разработчиков автоматизированных систем управления организационно-техническими комплексами и объектами транспортной инфраструктуры интеллектуальными средствами поддержки при обосновании требований и проектировании систем. Комплекс средств разработан с целью устранения логических и семантических разрывов между представлениями о разрабатываемой системе у участников данного процесса: заказчика и пользователей, системных архитекторов и проектировщиков, инженеров и программистов. Это достигается посредством реализации программно-управляемого процесса разработки и верификации формальных моделей требований и проектных решений в единой для всех его участников модельно-языковой и информационно-программной среде. Построение моделей осуществляется автоматизированным способом на основе шаблонов формального описания требований к функциональным и эксплуатационным характеристикам создаваемых систем, разработанных с использованием сетей Петри, временных автоматов, языков моделирования и проектирования SysML, FUML, OCL. Для валидации и верификации комплекса требований и проектных решений разработаны метамодель и алгоритм построения и анализа трассы выполнения модели в среде виртуальной машины VM FUML. Предложены способы интеграции и использования для автоматизированного тестирования моделей комплекса требований и проектных решений специализированных средств верификации CPN Tools, Rodin, SPIN и Modelica. Данный комплекс обеспечивает более эффективное взаимодействие заказчика и исполнителя как при разработке требований, так и при проектировании системы, раннее обнаружение и устранение дефектов посредством реализации автоматизированных процедур верификации, валидации и коррекции. Это позволит повысить качество требований и проектных решений, а также улучшить экономические показатели в части 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 системы управления, валидация и верификация, временные автоматы, проектирование и моделирование, сети Петри, функциональные и эксплуатационные требования.**

Введение

Разработка автоматизированных систем управления организационно-техническими комплексами и объектами транспортной инфраструктуры (АСУ КОТИ) представляет собой сложную и ресурсоемкую задачу, результат решения которой не всегда удовлетворяет заданным требованиям и укладывается в рамки выделенных временных и финансовых ресурсов. Одной из основных причин этого является наличие логических и семантических разрывов между представлениями о разрабатываемой системе у участников данного процесса: заказчика и пользователей, конструкторов и проектировщиков, инженеров и программистов. Это приводит к тому, что связи между тремя основными артефактами жизненного цикла разработки АСУ КОТИ являются несистемными, фрагментарными, неточными и несогласованными. Данное обстоятельство существенно затрудняет контроль их качества и проверку соответствия друг другу.

Одним из основных способов устранения этих разрывов является создание и применение комплексных программных средств (КПС), построенных и функционирующих на основе принципов и технологий модельно-ориентированной системной инженерии (МОСИ, Model-based systems engineering, MBSE). Примерами таких систем являются: IBM Rational Rhapsody Developer [1], Sparx Enterprise Arhitect [2], MASIW [3]. Данные системы в настоящее время используются в авиационной и космической отрасли, автомобилестроении, энергетике, в военно-промышленном комплексе. Для того чтобы эти средства приносили максимальную пользу, в них должна выдерживаться «золотая середина» между универсальностью и ориентированностью на определенную предметную область. При решении данной задачи необходимо, с одной стороны, использовать и опираться на общепринятые международные и отечественные стандарты, с другой стороны, разработать и применять предметно-ориентированные онтологии, описывающие объекты контроля и способы управления ими.

В данной статье описан комплекс программных средств, обеспечивающий единую модельно-языковую и

информационно-программную среду для разработки и контроля качества основных артефактов жизненного цикла АСУ КОТИ: комплекса требований и проектных решений.

Анализ технологий и средств промышленной разработки автоматизированных систем управления сложными комплексами и объектами

Методологическую и технологическую базу МОСИ образуют методические документы и спецификации, разработанные под эгидой таких организаций, как Международная организация по стандартизации (International Organization for Standardization, ISO), Международная электротехническая комиссия (International Electrotechnical Commission, IEC), Компьютерная ассоциация Института инженеров электротехники и электроники (Institute of Electrical and Electronics Engineers Computer Society, IEEE CS), Международный совет по системной инженерии (International Council on Systems Engineering, INCOSE), Международная ассоциация по управлению проектами (International Project Management Association, IPMA), Международный союз электросвязи (International Telecommunication Union, ITU), некоммерческое объединение OMG (Object Management Group) и многие другие. В разработке этих документов активное участие принимают научно-исследовательские организации (DISA, INCOSE, NIST и др.) и промышленные компании (AT&T, IBM, Oracle, Microsoft, Cisco Systems, NASA и др.). В нашей стране активные исследования в этой области ведутся в таких организациях, как ИСП РАН, ВМК МГУ им. М.В. Ломоносова, СПбГУ, Новосибирском ГТУ, ВКА им. А.Ф. Можайского и др.

МОСИ включает следующие основные технологии: разработка на основе моделей (Model driven development, MDD), архитектура на основе моделей (Model driven architecture, MDA), тестирование на основе моделей (Model based testing, MBT) [4, 5]. Реализация данных технологий предполагает, что создаваемые и используемые на всех этапах жизненного цикла (ЖЦ) системы артефакты (комплекс требований, технический проект, программные и технические реализации) представляются в виде формальных моделей, которые подвергаются процедурам верификации и валидации с помощью специальных автоматизированных методов и средств. По результатам проверки качества этих артефактов принимается решение об их пригодности для использования на следующих этапах ЖЦ системы или направлении на доработку.

В настоящее время на сайте OMG [6] опубликовано более 230 методических документов и спецификаций. С точки зрения рассматриваемых в настоящей статье вопросов наиболее важными из них являются следующие спецификации: 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).

В настоящее время наиболее перспективными и активно используемыми языками проектирования сложных автоматизированных информационных систем являются SysML, FUML, OCL. С их помощью можно решать задачи анализа, специфицирования, проектирования и тестирования систем, состоящих из аппаратных средств, программного обеспечения, данных, персонала, процедур, средств и других искусственных и природных систем. Эти языки используются в целом ряде систем автоматизированного проектирования – как промышленных (IBM Rational Rhapsody Developer, Sparx Enterprise Arhitect), так и исследовательских (Modelio [7], Eclipse Papyrus [8], GEMOC Studio [9]). Анализ результатов практического применения этих систем, проблемные вопросы, предложения по их разрешению и совершенствованию методов и средств представлены в целом ряде научно-технических публикаций и отчетов. Ниже представлен обзор наиболее интересных из них.

В первую очередь следует отметить монографии и практические руководства ряда известных специалистов в данной области. В фундаментальном труде известного сербского ученого и эксперта в области MBSE, профессора Белградского университета Д. Милисева (Dragan Milicev) [10] изложены методы, рекомендации и примеры использования для промышленной разработки сложных программно-технических систем (СПТС) исполняемого языка моделирования FUML (Foundational UML). В монографии [11], разработанной коллективом авторов в составе S. Friedenthal, A. Moore, R. Steiner, являющихся активными участниками разработки целого ряда методических документов и спецификаций OMG, а также применяющих свои знания на практике в таких компаниях, как Lockheed Martin и Raytheon Company, в сжатой и хорошо иллюстрированной практическими примерами форме не только изложены приемы и способы применения конструкций и механизмов языка SysML, но и дано объяснение его философии и принципов. Много полезной информации о способах применения для разработки СПТС языка системного моделирования SysML представлено в монографии Lenny Delligatti, имеющего богатый опыт разработки СПТС в компании Lockheed Martin [12].

В статье [13] представлена методика формирования UML-профиля, с помощью которого разрабатывается формальное описание требований к таким характеристикам качества ПО, как конфиденциальность, целостность, доступность и надежность. Профиль построен на основе языка OCL, реализован с помощью средств фреймворка Eclipse EMF. Разработанный с помощью данного профиля комплекс требований может быть верифицирован на предмет его полноты, корректности и согласованности. В случае обнаружения нарушения этих свойств определяется тип дефекта и локализуются содержащие его части модели требований. Прошедший проверку комплекс требований может быть использован для автоматической генерации элементов проекта модели: например, он генерирует наблюдаемые и управляемые интерфейсы из имен ассоциаций.

Как показал проведенный анализ, в настоящее время ведутся активные исследования и работы по созданию предметно-ориентированных языков моделирования (eXecutable Domain-Specific Modeling Languages, xDSML), использующие и развивающие ключевую концепцию МОСИ – метамоделирование. Результатом таких работ является, в частности, разработанный группой исследователей из Франции и Германии программный комплекс

GEMOC Studio [9]. Данный комплекс предназначен для создания xDSML на основе языков моделирования SysML/FUML, разработки исполняемых моделей на этом языке, их валидации и верификации с помощью графического аниматора и интеллектуального механизма трассировки. С помощью комплекса можно осуществлять мониторинг состояния анализируемых моделей (переходов, событий, значений переменных) во время их выполнения в виртуальной машине. В работе [14] дано описание используемой в данном средстве многомерной предметно-ориентированной метамодели трассировки, обеспечивающей более высокую производительность по сравнению со стандартной метамоделью UML за счет исключения из обработки избыточных данных.

При разработке и верификации распределенных динамических дискретных систем, какими являются АСУ КОТИ, полезно использовать методы и средства имитационного моделирования. Одним из наиболее эффективных и адекватных средств формального описания поведения подобных систем, анализа причинно-следственных связей и отслеживания происходящих при этом событий является математический аппарат и язык сетей Петри (СП). Язык СП по своей мощности не уступает контекстно-свободным языкам, класс контекстно-зависимых языков включает класс языков СП, а язык регулярных выражений и автоматных языков – суть подкласс языков СП. Сети Петри являются теоретической основой для формального представления диаграмм активности языка UML 2.5, их автоматической трансляция в сети Петри и обратно. Примеры использования сетей Петри для разработки представлены в целом ряде публикаций [15–18].

В частности, в работе [15] дано описание методики проектирования программного обеспечения с использованием UML-диаграмм последовательности в формате *.xmi* и представлен способ их автоматической трансляции в формат *.cpn*, используемый для описания сетей Петри. Результатом применения данного способа являются иерархические сети Петри, с помощью анализа которых осуществляется верификация проекта программного обеспечения (ПО), представленного в виде UML-диаграмм.

Для описания поведения динамических систем, кроме диаграмм активности, в языках моделирования SysML и FUML также активно используются диаграммы состояний и последовательностей. Математической основой для них являются автоматные модели и темпоральные логики.

В статье [19] дано описание двух методов реализации автоматической проверки нагруженных систем реального времени с использованием сценариев. В первом случае система моделируется как сеть временных автоматов (ВА). Во втором – с помощью набора 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.

Проведенный анализ нормативно-методических документов, научно-технических публикаций и систем позволил сделать вывод о том, что для формального описания комплекса требований и проектных решений при разработке АСУ КОТИ целесообразно использовать следующие средства и возможности языков SysML, FUML и OCL:

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

Для валидации и верификации формальных моделей комплекса требований и проектных решений целесообразно использовать виртуальную машину VM FUML, а также такие инструменты, как CPN Tools [20], Rodin [21], SPIN [22, 23].

Таким образом, для разработки моделей и алгоритмов формального описания основных артефактов процесса создания АСУ КОТИ предлагается использовать следующие математические модели и языки моделирования:

- сети Петри, временные автоматы;
- языки визуального моделирования и проектирования SysML, FUML, OCL, ALF.

В качестве автоматизированных средств верификации формальных моделей комплекса требований и проектных решений были использованы: VM FUML, CPN Tools, SPIN, Rodin. На основе представленных выше моделей, языков и средств создается единая для всех участников разработки АСУ КОТИ модельно-языковая и информационно-программная среда и реализуется последовательно-итерационный, программно-управляемый процесс проектирования и разработки.

Программные средства разработки и верификации формальной модели комплекса требований к АСУ КОТИ

Комплекс требований к АСУ КОТИ включает две основные группы требований: требования к ее функциональным возможностям и требования к качеству и условиям их реализации.

Для обеспечения автоматизированного построения формальных описаний этих требований в среде визуального моделирования были разработаны специальные 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. В частности, для формального описания требований к функциональным возможностям самого верхнего уровня их представления разработан шаблон «function requirement», который является расширением SysML-диаграммы вариантов использования – UseCase diagram (см. таблицу).

Шаблон представления требований
к функциональным возможностям АСУ КОТИ

Поле шаблона	Тип данных	Описание поля шаблона	Обязательность задания
Nameuc	String	Имя (идентификатор требования)	Обязательный
description	String	Краткое описание функции	Обязательный
Actor	actor (actors)	Пользователь функции	Обязательный
Subject	class (block)	Класс (блок, модуль) системы, реализующий функцию	Обязательный
basic_scenarios	scenarios (control &object_flow)	Основной сценарий выполнения функции	Опциональный
alter_scenarios	scenarios (control &object_flow)	Альтернативный сценарий выполнения функции	Опциональный

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 диаграмм пакетов (*Package*), внешнего и внутреннего представления блоков (*Block Definition*, *Internal Block*).

Основной и альтернативные сценарии реализации функций представляются в форме активных и пассивных классов (*Active Class*, *Passive Class*). Для описания методов классов используются диаграммы деятельности (*Activity*), состояний (*State Machine)* и взаимодействия (*Sequence*).

Следующим шагом построения модели комплекса требований является разработка требований к качеству реализации каждой функции, т. е. к эксплуатационным характеристикам АСУ КОТИ:

$$d\_reqs = \left(r_1^{f_i},\ r_2^{f_i}, r_3^{f_i}, r_4^{f_i} \ldots\right),$$

где $r_1^{f_i}$ – требования к оперативности исполнения функции f_i;

$r_2^{f_i}$ – требования к производительности (например, объемы хранимых, обрабатываемых и передаваемых данных, количество пользователей, количество и размеры запросов в единицу времени и др.);

$r_3^{f_i}$ – требования к надежности (коэффициент готовности, время безотказной работы, время восстановления и др.);

$r_4^{f_i}$ – требования к защищенности.

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*), ассоциации (*association*), зависимости (*dependency*) устанавливаются связи между структурными, поведенческими, ограничительными и другими аспектами и характеристиками модели, которые также представляются с помощью средств языка OCL.

Все АСУ КОТИ 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 только вход и выход. Во вторую группу входят реагирующие системы, выполняющие определенные операции и изменяющие свое состояние в зависимости от внешних воздействий. Третью группу 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 представляется четверкой

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

где $P = \{p_1, p_2, \ldots, p_n\}$ – множество мест;
$T = \{t_1, t_2, \ldots, t_n\}$ – 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 сетей Петри, которые следует использовать для верификации формальных моделей требований и архитектуры, являются: живость (отсутствие блокировок и зацикливаний) и ограниченность. СП обладает свойствами 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 будут 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 сетей Петри 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 АСУ КОТИ.

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.

Состав и структура комплекса моделей и программных средств, с помощью которых реализуются процедуры верификации и валидации формальной модели требований к АСУ КОТИ, представлены на рисунке 1.

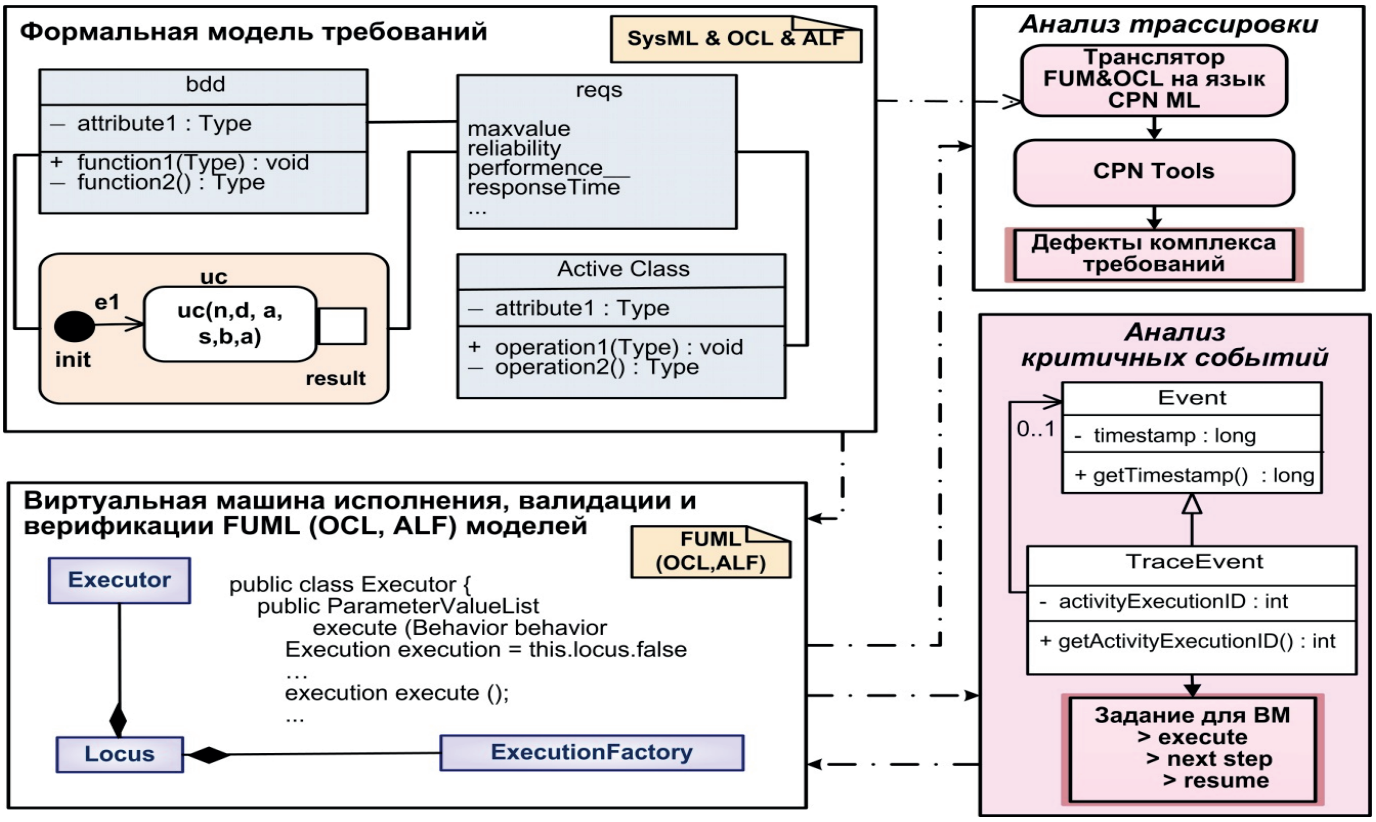

Рис. 1. Состав и структура комплекса моделей и программных средств реализации процедур верификации и валидации формальной модели требований к АСУ КОТИ

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.

Для реализации процедуры валидации модели требований посредством анализа ее состояния в контрольных точках была разработана представленная на рисунке 2 метамодель построения трассы ее исполнения в среде VM FUML. Трасса (класс *Trace*) формируется из конструкций, представленных в трех пакетах: Steps, States и PNnet, описывающих контрольные точки, состояния модели в них и абстрактный синтаксис СП соответственно. На рисунке 3 приведен фрагмент разработанного на основе данной метамодели

алгоритма построения трассы выполнения и анализа модели комплекса требований в среде VM FUML. В отличие от стандартного способа, в данном алгоритме исключены повторные проверки неизменяемых данных и обеспечена возможность его многократного использования.

Для того чтобы требования, представленные в виде диаграмм активностей и состояний, можно было верифицировать средствами CPN Tools и Rodin, они предварительно транслируются на язык CPN ML и EventB соответственно с помощью разработанных для этого модулей Eclipse Papyrus.

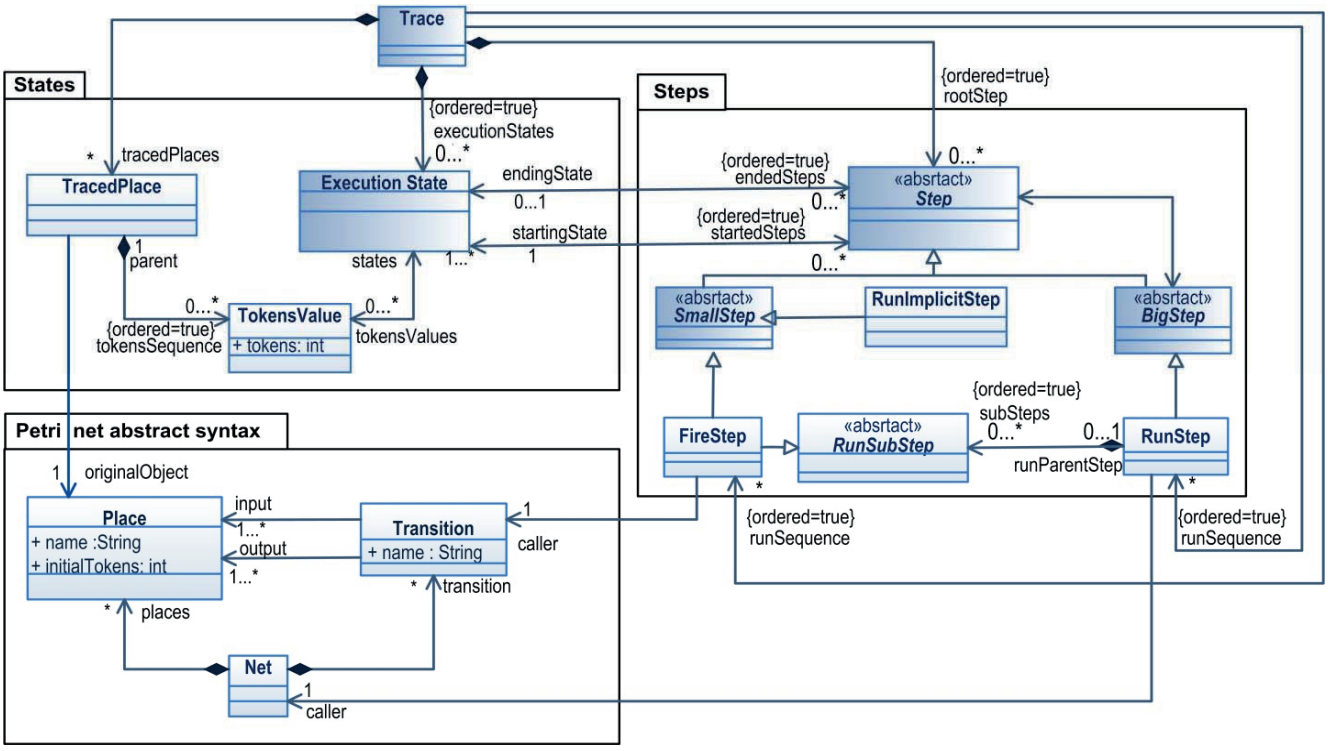

Рис. 2. Метамодель построения трассы исполнения формальной модели комплекса требований в среде VM FUML

```
begin
   c_trace, c_exeState, c_step, c_smallStep, c_bigStep ← createBaseGeneric-
Classes()
  mm_trace ← {c_trace, c_exeState, c_step, c_smallStep, c_bigStep}
  foreach c_exe ∈ {c ∈ mm_exe | containsDynamicProperties(c)}
    do
     c_traced ← createClass()
     mm_trace ← mm_trace ∪ {c_traced}
     c_trace.createReferenceTo(c_traced, [0..*], unordered)
     if containsStaticProperties(c_exe) then
        c_orig ← getClassFromAbstractSyntax(c_exe)
      c_traced.createReferenceTo(c_orig, [1..1])
   foreach p ∈ getDynamicPropertiesOf(c_exe) do
      c_value ← createClass()
      mm_trace ← mmtrace ∪ {c_value}
     c_value.properties ← {copyProperty(p)}
     c_traced.createReferenceTo(c_value, [0..*], ordered)
     c_value.createReferenceTo(c_traced, [1..1])
     c_exeState.createReferenceTo(c_value, [0..*], unordered)
     c_value.createReferenceTo(c_exeState, [1..1])
  foreach r ∈ exeTransf do
    map_steps ← createMap()
    createStepClass(r, map_steps,mm_trace, c_smallStep, c_bigStep)
    replaceReferencesToExecutionMM(mm_trace,mm_as,mm_exe)
end
```

Рис. 3. Фрагмент алгоритма построения трассы выполнения модели комплекса требований в среде VM FUML

В случае обнаружения каких-либо дефектов в комплекс требований вносятся необходимые исправления, после чего он подвергается повторной процедуре валидации и верификации. После прохождения описанной в данном разделе процедуры тестирования комплекс функциональных и эксплуатационных требований будет обладать следующими необходимыми для его использования на этапе проектирования АСУ КОТИ свойствами: полнотой, корректностью, непротиворечивостью, системностью, однозначностью, согласованностью. Такие свойства комплекса требований, как прослеживаемость, верифицируемость и модифицируемость обеспечиваются средствами единой модельно-языковой и информационно-программной среды, в рамках которой осуществляется разработка всех артефактов жизненного цикла АСУ КОТИ.

Алгоритмы и программные средства разработки и верификации архитектуры и проектных решений

Проектирование архитектуры и проектных решений АСУ КОТИ 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:

- для описания состава и структуры аппаратных и программных компонентов системы – диаграммы внешнего (bdd, block definition diagram) и внутреннего (ibd, internal block diagram) блоков;
- для представления реализующих определенные функции компонентов системы – диаграммы активных классов (active_class);
- для описания используемых или создаваемых в процессе функционирования системы объектов и артефактов – диаграммы пассивных классов (passive_class);
- для представления алгоритмов реализации методов классов (функций) – диаграммы деятельности (*Activity*), состояний (*State Machine)* и взаимодействия (*Sequience*).

Для представления отношений между ними и описания различного рода ограничений и условий используются средства языка OCL и библиотека Eclipse OCL Standard Library [16].

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

$$N = (P, T, C, F, m_1, S, Z),$$

где P, T, F и m_1 имеют то же 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 шаблона для проектирования архитектуры и проектных решений АСУ КОТИ с помощью диаграмм состояний были использованы математические модели временных автоматов (ВА) [24, 25], 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 $\Re$;

$f: S \to P$ – функция, которая ставит в соответствие каждой позиции $s \in S$ некоторый предикат $p \in P$; в общем случае p есть логическое выражение из ограничений вида $c \le k$, $c < k$, $c_i - c_j \le k$, $c_i - c_j < k$, где $c_i, c_j \in C$, $k \in \Re$. Переход из позиции $s \in S$ должен быть выполнен до того, как предикат $p = f(s)$ станет ложным, в противном случае автомат прекращает работу;

D – конечное множество действий;

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

$s_0 \in S$ – начальная позиция автомата.

Для моделирования АСУ КОТИ, которые состоят из нескольких взаимодействующих компонент, целесообразно использовать сети временных автоматов (СВА).

Формальная модель архитектуры и проектных решений должна обладать следующими характеристиками 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.

Состав и структура комплекса программных средств, обеспечивающих проверку этих характеристик, представлены на рисунке 4. Основными из них являются: виртуальная машина VM FUML, CPN Tools, SPIN, Rodin, Modelica. Проверка полноты и корректности реализации функциональных требований осуществляется посредством анализа трасс выполнения функций в среде VM FUML, включая полноту учета и правильность обработки критичных событий. Данный вид проверки выполняется при непосредственном участии экспертов и системных проектировщиков. Кроме того, модели, представленные в виде диаграмм активностей (*d_act*), транслируются на язык CPN ML и затем верифицируются средствами CPN Tools. Диаграммы последовательностей (*d_seq*) транслируются на язык Promela для последующей верификации в среде SPIN. Диаграммы состояний (*d_sm*) транслируются на язык EventB и проверяются с помощью средств Rodin.

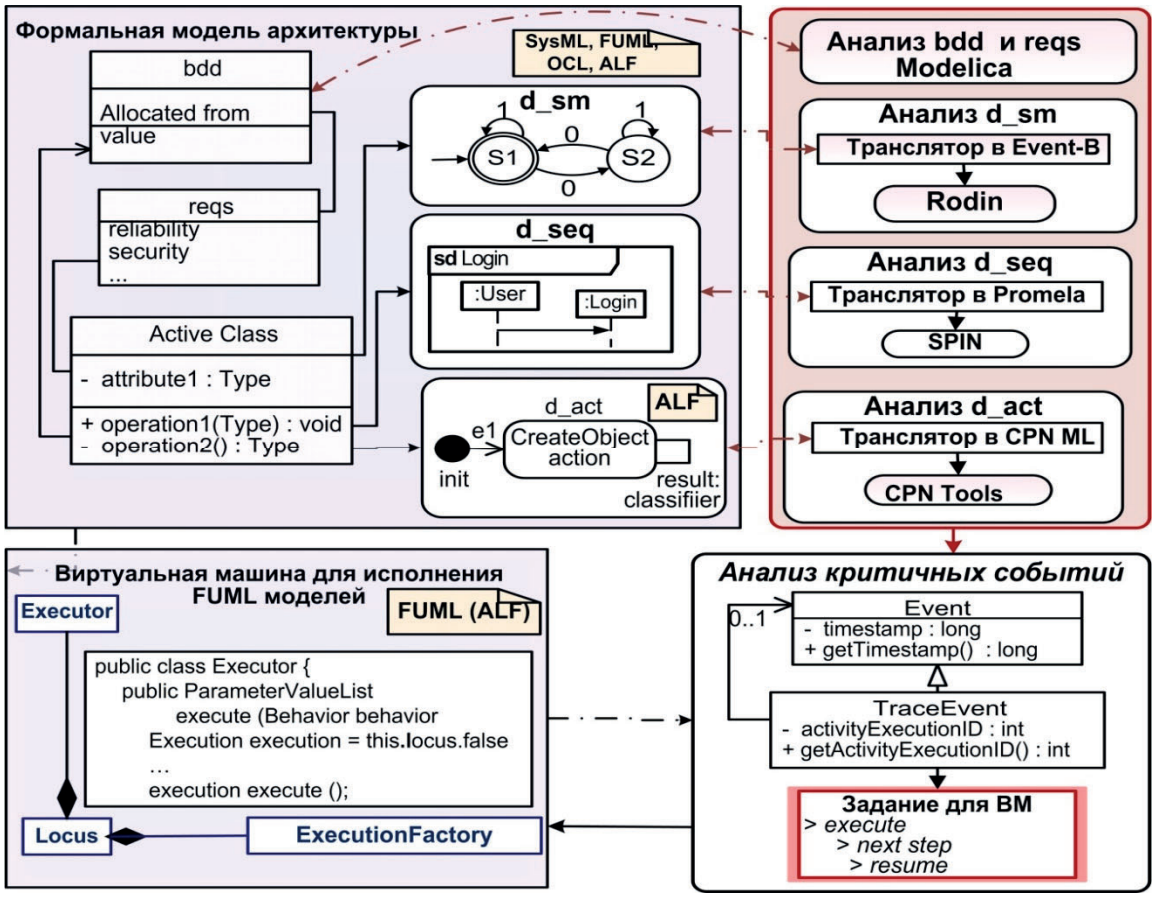

Рис. 4. Состав и структура комплекса программных средств, обеспечивающих валидацию и верификацию проектных решений

Процедура проверки непротиворечивости, согласованности и неизбыточности входящих в проект АСУ КОТИ диаграмм заключается в проверке следующих условий:

- каждый класс представлен не менее чем в одной поведенческой диаграмме;
- для каждого метода класса в проекте имеется единственная диаграмма поведения (активности, состояний, последовательности);
- сигнатуры методов класса и описывающих их реализацию диаграмм поведения совпадают;
- каждая поведенческая диаграмма имеет связь (отношение *realization*) с методом класса, который она реализует;
- все действующие лица (*actor*) и информационные объекты из поведенческих диаграмм должны быть представлены в виде классов (*class*);
- типы данных элементов диаграмм, связанных отношениями агрегации и композиции, должны совпадать.

Для описания и верификации приведенных выше условий используется язык OCL и библиотека Eclipse OCL Standard Library [26].

Для верификации технической части проектных решений предлагается использовать средство моделирования киберфизических систем OpenModelica [27]. Проектные решения, представленные в виде SysML-диаграмм блоков и классов, с помощью специальных программ транслируются в соответствующие конструкции объектно-ориентированного языка Modelica.

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 АСУ КОТИ требованиям пользователей, условиям применения и нормативным документам.

Заключение

Представленный в статье комплекс программных средств предназначен для реализации программно-управляемого процесса разработки АСУ КОТИ в единой для всех его участников модельно-языковой и информационно-программной среде, построенной на основе технологий объектно-ориентированного анализа и проектирования, языков и средств визуального моделирования SysML, FUML, OCL. Его применение позволит существенно сократить логические и семантические разрывы между представлениями о разрабатываемой системе у участников процесса создания АСУ КОТИ, повысит эффективность их

взаимодействия и приведет к повышению качества всех артефактов жизненного цикла разработки систем.

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 как в случае обнаружения каких-либо дефектов, так и при изменении самих требований или условий эксплуатации АСУ КОТИ.

Литература

1. IBM Engineering Systems Design Rhapsody – Developer. – URL: http://www.ibm.com/ru-ru/marketplace/uml-tools (дата обращения 20.06.2019).

2. Sparx Systems Enterprise Architect. – URL: http://sparxsystems.com/products/ea (дата обращения 21.05.2019).

3. Буздалов Д.В. Инструментальные средства проектирования систем интегрированной модульной авионики / Д.В. Буздалов, С.В. Зеленов, Е.В. Корныхин, А.К. Петренко, А.В. Страх, А.А. Угненко, А.В. Хорошилов // Труды ИСП РАН. – 2014. Т. 26. – Вып. 1. – С. 201–230.

4. Systems Engineering and Software Engineering // SEBoK. – URL: http://www.sebokwiki.org/wiki/Systems_Engineering_and_Software_Engineering (дата обращения 25.05.2019).

5. Hart L.E. Introduction to Model-Based System Engineering (MBSE) and SysML // International Council on Systems Engineering Website. – URL: http://www.incose.org/docs/default-source/delaware-valley/mbse-overview-incose-30-july-2015.pdf (дата обращения 21.05.2019).

6. The OMG Specifications Catalog // Object Management Group. – URL: http://www.omg.org/spec (дата обращения 20.06.2019).

7. Modelio Open Source – UML and BPMN free modeling tool. – URL: http://www.modelio.org (дата обращения 20.06.2019).

8. Eclipse Papyrus. Modeling environment. – URL: http://www.eclipse.org/papyrus (дата обращения 20.06.2019).

9. Eclipse GEMOC Studio. – URL: http://projects.eclipse.org/projects/modeling.gemoc (дата обращения: 20.06.2019).

10. Milicev D. Model-Driven Development with Executable UML, Indianapolis, Wiley Publishing, Inc., Wrox, 2009. – 816 p.

11. Friedenthal S., Moore A., Steiner R. A Practical Guide to SysML: The Systems Modeling Language. 2nd edition, Morgan Kaufmann, 2011. – 640 p.

12. Delligatti L. SysML Distilled: A Brief Guide to the Systems Modeling Language, Addison-Wesley Professional, 2013. – 304 p.

13. Hatebur D., Heisel M. (2010) A UML Profile for Requirements Analysis of Dependable Software. *In: Schoitsch E. (eds) Proceedings of the 29th International Conference on Computer Safety, Reliability, and Security (SAFECOMP 2010), Vienna, Austria, 14–17 September 2010. Lecture Notes in Computer Science, Vol. 6351*, Berlin, Springer-Verlag, 2010. – Pp. 317–331. – URL: http://www.researchgate.net/publication/221147725_A_UML_Profile_for_Requirements_Analysis_of_Dependable_Software (дата обращения 20.06.2019).

14. Bousse E., Mayerhofer T., Combemale B., Baudry B. Advanced and Efficient Execution Trace Management for Executable Domain-Specific Modeling Languages, *Software and Systems Modeling*, 2019, Vol. 18, No. 1. – Pp. 385–421. – URL:http://link.springer.com/article/10.1007/s10270-017-0598-5 (дата обращения 27.04.2019).

15. Марков А.В. Автоматизация проектирования и анализа программного обеспечения с использованием языка UML и сетей Петри : дисс. … канд. техн. наук: 05.13.11. – Новосибирск, 2015. – 176 с.

16. Anand S. et al. An Orchestrated Survey on Automated Software Test Case Generation, *Journal of Systems and Software,* 2013, Vol. 86, Is. 8. – Pp. 1978–2001.

17. Andre E., Choppy C., and Reggio G. Activity Diagrams Patterns for Modeling Business Processes. – URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.650.6505&rank=1 (дата обращения 19.06.2019).

18. Rahim M., Boukala-ioualalen M., Hammad A. Petri Nets Based Approach for Modular Verification of SysML Requirements on Activity Diagrams. – URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.664.7517&rank=1 (дата обращения 27.05.2019).

19. Li S., Balaguer S., David A. et al. Scenario-Based Verification of Real-Time Systems Using Uppaal, Formal Methods in System Design, 2010, Vol. 37, Is. 2–3. – Pp. 200–264.

20. CPN Tools – A tool for editing, simulating, and analyzing Colored Petri nets. – URL: http://cpntools.org (дата обращения 27.05.2019).

21. Home of Event-B and the Rodin Platform. – URL: http://www.event-b.org (дата обращения 11.05.2019).

22. SPIN – Formal Verification. – URL: http://spinroot.com/spin/whatispin.html (дата обращения 11.06.2019)

23. Введение в язык PROMELA и систему комплексной верификации SPIN : учебное пособие / И.В. Шошмина, Ю.Г. Карпов. – СПб. : Изд-во Политехнического ун-та, 2010. – 110 с.

24. Bengtsson J., Yi W. Timed Automata: Semantics, Algorithms and Tools. Uppsala University. *In: Desel J., Reisig W., Rozenberg G. (eds) Lectures on Concurrency and Petri Nets. ACPN 2003. Lecture Notes in Computer Science, Vol. 3098*, Berlin, Springer-Verlag, 2004. – pp. 87–124. – URL: http://www.fi.muni.cz/~xpelanek/IA158/TA-intro.pdf. (дата обращения 27.05.2019).

25. Твардовский А.С., Лапутенко А.В. О возможностях автоматного описания параллельной композиции временных автоматов // Труды ИСП РАН. – 2018. – Том 30, Вып. 1. – С. 25–40.

26. Eclipse OCL (Object Constraint Language). – URL: http://projects.eclipse.org/projects/modeling.mdt.ocl (дата обращения 27.05.2019).

27. OpenModelica. – URL: http://www.openmodelica.org (дата обращения 27.05.2019).

Complex of Software for Development and Verification of Requirements and Design Solutions Automated Control System by Transport Infrastructure Objects

A.V. Baev, PhD A.V. Samonov
Military Space Academy named after A.F. Mozhaisky
Saint Petersburg, Russia
a.baih@mail.ru, a.samonov@mail.ru

Abstract. **The composition, structure and methods of application of a complex of software that provides developers of automated control systems of organizational and technical complexes and objects of transport infrastructure with intelligent means of support in justifying the requirements and designing systems are presented. The complex of means is developed for the purpose of elimination of logical and semantic gaps between representations about the developed system at participants of this process: the customer and users, system architects and designers, engineers and programmers. This is achieved through the implementation of a program-driven process of development and verification of formal models of requirements and design solutions in a common model-language and information-software environment for all its participants. Construction of models is carried out by the automated method on the basis of templates of the formal description of requirements to functional and operational characteristics of the created systems developed with use of Petri nets, time automata, languages of modeling and design SysML, FUML, OCL.**

For validation and verification of complex requirements and design solutions developed meta-model and algorithm for design and analysis of execution routes of model that run in a virtual machine environment FUML VM. The methods of integration and use for automated testing of models of requirements and design solutions of specialized verification tools CPN Tools, Rodin, SPIN and Modelica presents. 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, early detection and elimination of defects through the implementation of automated verification, validation and correction procedures. This will improve the quality of requirements and design solutions, as well as improve economic performance in terms of reducing the financial and time costs 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.**

References

1. IBM Engineering Systems Design Rhapsody – Developer. Available at: http://www.ibm.com/ru-ru/marketplace/uml-tools (accessed 20.06.2019).

2. Sparx Systems Enterprise Architect. Available at: http://sparxsystems.com/products/ea (accessed 21.05.2019).

3. Buzdalov D.V. et al. Tools for System Design of Integrated Modular Avionics [Instrumental'nye sredstva proektirovaniya sistem integrirovannoy modul'noy avioniki], *Proc. of the Institute for System Programming of the RAS [Trudy ISP RAN]*, 2014, Vol. 26, Is. 1. – Pp. 201–230.

4. SEBoK: Systems Engineering and Software Engineering. Available at: http://www.sebokwiki.org/wiki/Systems_Engineering_and_Software_Engineering (accessed 25.05.2019).

5. Laura E. Hart. Introduction To Model-Based System Engineering (MBSE) and SysML. Available at: http://www.incose.org/docs/default-source/delaware-valley/mbse-overview-incose-30-july-2015.pdf (accessed 21.05.2019).

6. Object Management Group: The OMG Specifications Catalog. Available at: http://www.omg.org/spec (accessed 20.06 2019).

7. Modelio Open Source – UML and BPMN free modeling tool. Available at: http://www.modelio.org (accessed 20.06 2019).

8. Eclipse Papyrus. Modeling environment. Available at: http://www.eclipse.org/papyrus (accessed 20.06 2019).

9. Eclipse GEMOC Studio. Available at: http://projects.eclipse.org/projects/modeling.gemoc (accessed 20.06 2019).

10. Milicev D. Model-Driven Development with Executable UML, Indianapolis, Wiley Publishing, Inc., Wrox, 2009. – 816 p.

11. Friedenthal S., Moore A., Steiner R. A Practical Guide to SysML: The Systems Modeling Language. 2nd edition, Morgan Kaufmann, 2011. – 640 p.

12. Delligatti L. SysML Distilled: A Brief Guide to the Systems Modeling Language, Addison-Wesley Professional, 2013. – 304 p.

13. Hatebur D., Heisel M. (2010) A UML Profile for Requirements Analysis of Dependable Software. *In: Schoitsch E. (eds) Proceedings of the 29th International Conference on Computer Safety, Reliability, and Security (SAFECOMP 2010), Vienna, Austria, 14–17 September 2010. Lecture Notes in Computer Science, Vol. 6351*, Berlin, Springer-Verlag, 2010. – Pp. 317–331. Available at: http://www.researchgate.net/publication/221147725_A_UML_Profile_for_Requirements_Analysis_of_Dependable_Software (accessed 20.06 2019).

14. Bousse E., Mayerhofer T., Combemale B., Baudry B. Advanced and Efficient Execution Trace Management for Executable Domain-Specific Modeling Languages, *Software and Systems Modeling*, 2019, Vol. 18, No. 1. – Pp. 385–421. Available at: http://link.springer.com/article/10.1007/s10270-017-0598-5 (accessed 27.04.2019).

15. Markov A.V. Automation of Software Design and Analysis Using UML and Petri Nets [Avtomatizatsiya proyektirovaniya i analiza programmnogo obespecheniya s ispol'zovaniyem yazyka UML i setey Petri]: diss. on competition of a scientific degree Ph.D. (Engin.), Novosibirsk, 2015. – 176 p.

16. Anand S. et al. An Orchestrated Survey on Automated Software Test Case Generation, *Journal of Systems and Software*, 2013, Vol. 86, Is. 8. – Pp. 1978–2001.

17. Andre E., Choppy C., and Reggio G. Activity Diagrams Patterns for Modeling Business Processes. Available at:http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.650.6505&rank=1 (accessed 19.06.2019).

18. Rahim M., Boukala-ioualalen M., Hammad A. Petri Nets Based Approach for Modular Verification of SysML Requirements on Activity Diagrams. Available at: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.664.7517&rank=1 (accessed 27.05.2019).

19. Li S., Balaguer S., David A. et al. Scenario-Based Verification of Real-Time Systems Using Uppaal, Formal Methods in System Design, 2010, Vol. 37, Is. 2–3. – Pp. 200–264.

20. CPN Tools – A tool for editing, simulating, and analyzing Colored Petri nets. Available at: http://cpntools.org (accessed 27.05.2019).

21. Home of Event-B and the Rodin Platform. Available at: http://www.event-b.org (accessed 11.05.2019).

22. SPIN – Formal Verification. Available at: http://spinroot.com/spin/whatispin.html (accessed 11.06.2019)

23. Shoshmina I.V., Karpov Yu.G. Introduction to PROMELA language and SPIN Integrated Verification System: Study guide [Vvedeniye v yazyk PROMELA i sistemu kompleksnoy verifikatsii SPIN: Uchebnoe posobie], Saint Petesburg, Peter the Great St. Petersburg Polytechnic University, 2010. – 110 p.

24. Bengtsson J., Yi W. Timed Automata: Semantics, Algorithms and Tools. Uppsala University. *In: Desel J., Reisig W., Rozenberg G. (eds) Lectures on Concurrency and Petri Nets. ACPN 2003. Lecture Notes in Computer Science, Vol. 3098*, Berlin, Springer-Verlag, 2004, pp. 87–124. Available at: http://www.fi.muni.cz/~xpelanek/IA158/TA-intro.pdf. (accessed 27.05.2019).

25. Tvardovskii A.S., Laputenko A.V. On the possibilities of FSM description of parallel composition of timed Finite State Machines [O vozmozhnostyakh avtomatnogo opisaniya parallel'noy kompozitsii vremennykh avtomatov], *Proc. of the Institute for System Programming of the RAS [Trudy ISP RAN]*, 2018, Vol. 30, Is. 1. – Pp. 25–40.

26. Eclipse OCL (Object Constraint Language). Available at: http://projects.eclipse.org/projects/modeling.mdt.ocl. (accessed: 27.05.2019).

27. OpenModelica. Available at: http://www.openmodelica.org (accessed: 27.05.2019).