УДК 004.05
DOI: 10.26102/2310-6018/2021.35.4.019

Методика проектирования автоматизированных систем управления специальными организационно-техническими системами

А.В. Баев✉, А.В. Самонов, В.М. Сафонов

Военно-космическая академия имени А.Ф. Можайского,
Санкт-Петербург, Российская федерация
baih@mail.ru✉

Резюме. Успешная реализация проектов по созданию автоматизированных систем управления специальными организационно-техническими системами в значительной степени зависит от качества комплекса предъявленных к ним требований, а также полноты и корректности их реализации в проектных решениях. Необходимым условием решения этих задач является создание единой для всех участников процесса разработки таких систем модельно-языковой и информационно-программной среды и реализация программно-управляемого процесса обоснования требований, проектирования и реализации проекта. В качестве концептуальной и технологической основы для реализации данного подхода предложено использовать концепции и методы модельно-ориентированной системной и программной инженерии, онтологические модели и языки визуального моделирования. Для реализации программно-управляемого процесса разработки комплекса требований и проектных решений построены и используются паттерны проектирования, созданные на основе онтологии «Модель качества программно-технических комплексов» и UML диаграмм вариантов использования, поведения и классов. Модель качества комплекса требований состоит из характеристик комплекса требований в целом (полнота, непротиворечивость, неизбыточность, системность) и характеристик отдельных требований (внутренняя полнота, корректность, однозначность, прослеживаемость, проверяемость и модифицируемость). Проверка качества формальных моделей комплекса требований и проектных решений осуществляется посредством их валидации и верификации в среде графовой базы данных Neo4j с помощью специальных тестовых запросов на языке Cypher.

Ключевые слова: валидация и верификация, графовые модели, диаграммы поведения и классов, модель качества, онтологии, характеристики качества

Для цитирования: Баев А.В., Самонов А.В., Сафонов В.М. Методика проектирования автоматизированных систем управления специальными организационно-техническими системами. . *Моделирование, оптимизация и информационные технологии.* 2021;9(4). Доступно по: https://moitvivt.ru/ru/journal/pdf?id=1063 DOI: 10.26102/2310-6018/2021.35.4.019

Methodology of designing automated control systems for special organizational and technical systems

A.V. Baev✉, A.V. Samonov, V.M. Safonov

A.F. Mozhaisky Military Space Academy,
Saint Petersburg, Russian Federation
baih@mail.ru✉

Abstract: The successful implementation of projects on the creation of automated control systems for special organizational and technical systems largely depends on the quality of the complex requirements presented to them, as well as the completeness and accuracy of their execution in design solutions. A necessary condition for solving these tasks is the creation of a model-language and information-software environment that is uniform for all participants in the development of such systems and the implementation of a software-controlled process for justifying requirements, designing, and implementing the project. It is proposed to use the concepts and methods of model-oriented system and software engineering, ontological models, and visual modeling languages as a conceptual and technological basis for this approach implementation. To implement the software-driven process of developing a set of requirements and design solutions are built and used design patterns created based on the ontology "Quality Model of software and hardware complexes" and UML diagrams of uses, behaviors, and classes. The quality model requirements set consists of the characteristics of requirements set as a whole (completeness, consistency, non-redundancy, systematicity) and the characteristics of individual requirements (internal completeness, accuracy, unambiguity, traceability, verifiability, and modifiability). The quality of the formal models of the criteria set and the design solutions are checked by validating and verifying them in the Neo4j graph database environment using dedicated test queries in the Cypher language.

***Keywords*:** attribute graph models, model verification, quality characteristics of a set of requirements, UML behavior diagrams

For citation: Baev A.V., Samonov A.V., Safonov V.M. Methodology of designing automated control system for special organization and technical sistems. *Modeling, Optimization and Information Technology*. 2021;9(4). Available from: https://moitvivt.ru/ru/journal/pdf?id=1063 DOI: 10.26102/2310-6018/2021.35.4.019 (In Russ).

Введение

Специальные организационно-технические системы (СОТС) предназначены для обеспечения надежного функционирования энергетических, транспортных, аэрокосмических, производственных, телекоммуникационных, финансовых и других комплексов и систем. Характерными особенностями СОТС являются иерархичность структуры и связей между элементами, многовариантность реализации функций управления, территориальная распределенность, функционирование в режиме реального времени. Для обеспечения их корректного и надежного функционирования создаются и используются автоматизированные системы управления (АСУ). Процесс разработки АСУ СОТС включает три основных этапа: определение требований, проектирование и реализацию. Особую роль и исключительно важное значение для успешного осуществления проекта имеет качество артефактов, создаваемых на первых двух этапах: комплекса требований и проектных решений АСУ СОТС. Из практики создания программно-технических комплексов известно, что удельная стоимость исправления дефектов возрастает по экспоненциальному закону распределения по мере продвижения продукта по этапам разработки. При этом затраты на исправление дефекта, внесенного на этапе обоснования требований, при его обнаружении на этапе проектирования возрастают в 5 раз, на этапе реализации – в 10 раз, на этапе тестирования – в 20 раз, а в ходе эксплуатации – в 100 раз и более [1]. Основными факторами, не позволяющими в настоящее время кардинально улучшить качество разрабатываемых требований, являются:

– наличие терминологических и семантических разрывов в представлениях о создаваемой системе у различных участников данного процесса: пользователей, проектировщиков, разработчиков и тестировщиков;

– объективная сложность задачи формирования адекватного и корректного комплекса требований к создаваемой системе на основе исходной неформальной формы их представления;

– отсутствие средств автоматизированного построения, моделирования и оценивания характеристик качества комплекса требований.

Для преодоления негативного влияния указанных выше факторов и повышения качества и результативности процессов обоснования требований и проектирования СОТС предлагается разработать единую модельно-языковую и информационно-программную среду (ЕМЯИПС). В статьей представлено описание моделей, методов и средств разработки ЕМЯИПС, методика и алгоритмы реализации в ней программно-управляемого процесса разработки и валидации комплекса требований и проектных решений.

Анализ современных технологий, методов и средств разработки АСУ СОТС

Исключительная актуальность совершенствования технологий и средств разработки АСУ СОТС 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 (https://www.omg.org/). 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), FUML (Semantics of a Foundational Subset for Executable UML Models), ReqIF (Requirements Interchange Format).

Теоретической основой технологий разработки АСУ СОТС 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]. Наиболее известными технологиями промышленной разработки сложных программно-технических комплексов являются Rational Unified Process (RUP), Microsoft Solution Framework (MSF), Oracle Method, SADT (IDEFx). Практическими реализациями этих технологий являются линейки продуктов IBM Rhapsody, Sparx Enterprise Architect, Modelio, MASIW. Данные системы обеспечивают разработчиков 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, ArhiMate, AADL. Для контроля качества артефактов жизненного цикла (ЖЦ) систем, получаемых в процессе проектирования и разработки систем, применяются различные средства верификации и валидации, например, CPN Tools, Rodin, SPIN, Modelica.

В нашей стране активные исследования в этой области ведутся в 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. [3, 4]. Примеры использования онтологического подхода при разработке сложных программно-технических комплексов представлены в [5-7].

Методика разработки комплекса требований и проектных решений АСУ СОТС

Основными этапами методики разработки комплекса требований и проектных решений АСУ СОТС являются (Рисунок 1):

– представление комплекса требований к АСУ СОТС на языке XML посредством заполнения специальных паттернов, соответствующих структуре технического задания на разработку АСУ;

– построение модели комплекса требований в виде совокупности диаграмм вариантов использования (d_uc) и диаграмм классов (d_class);

– описание алгоритмов реализации функциональных возможностей АСУ СОТС с помощью специальных шаблонов, построенных на основе диаграмм поведения (d_act, d_seq, d_sm);

– разработка архитектуры и проектных решений АСУ СОТС в виде совокупности диаграмм блоков (d_block) и диаграмм классов (d_class);

– описание требований к эксплуатационно-техническим характеристикам АСУ СОТС на языке объектных ограничений OCL (Object Constraint Language).

Построение формальной модели комплекса требований к АСУ ОТС осуществляется в среде визуального моделирования с помощью специальных шаблонов – паттернов проектирования. Для описания алгоритмов реализации функциональных возможностей АСУ СОТС используются паттерны проектирования, разработанные на основе диаграмм поведения: деятельности (d_act), последовательности (d_seq) и состояний (d_sm). Выбор этих типов диаграмм для построения паттернов проектирования обусловлен следующими соображениями. АСУ ОТС, с точки зрения особенностей алгоритма их функционирования, можно разделить на три большие группы. Первую группу образуют системы, выполняющие свои функции в соответствии с определенным алгоритмом, порядок выполнения которого зависит только от входных данных. Для описания алгоритмов данного типа целесообразно использовать диаграммы деятельности (*d_act*). Во вторую группу входят реагирующие системы, поведение которых определяется внешними воздействиями и текущим состоянием. Для описания алгоритмов данного типа целесообразно использовать диаграммы состояний (*d_sm*). Третью группы образуют системы, взаимодействующие друг с другом посредством обмена сигналами и сообщениями. Для описания алгоритмов данного типа целесообразно использовать диаграммы последовательности (*d_seq*).

При разработке архитектуры и проектных решений АСУ СОТС используются паттерны проектирования, построенные на основе диаграмм блоков и диаграмм классов. Функциональные блоки системы, реализующие функциональные возможности, описываются посредством соответствующих расширений диаграмм пакетов (Package), внешнего и внутреннего представления блоков (Block Definition, Internal Block). Для описания пользователей и систем внешнего окружения используются паттерны на основе активных классов (Active Class). Пассивные классы (Passive Class) описывают артефакты, создаваемые и используемые АСУ во время функционирования

(информационные, материальные, вычислительные, сетевые ресурсы, а также команды, сигналы, события).

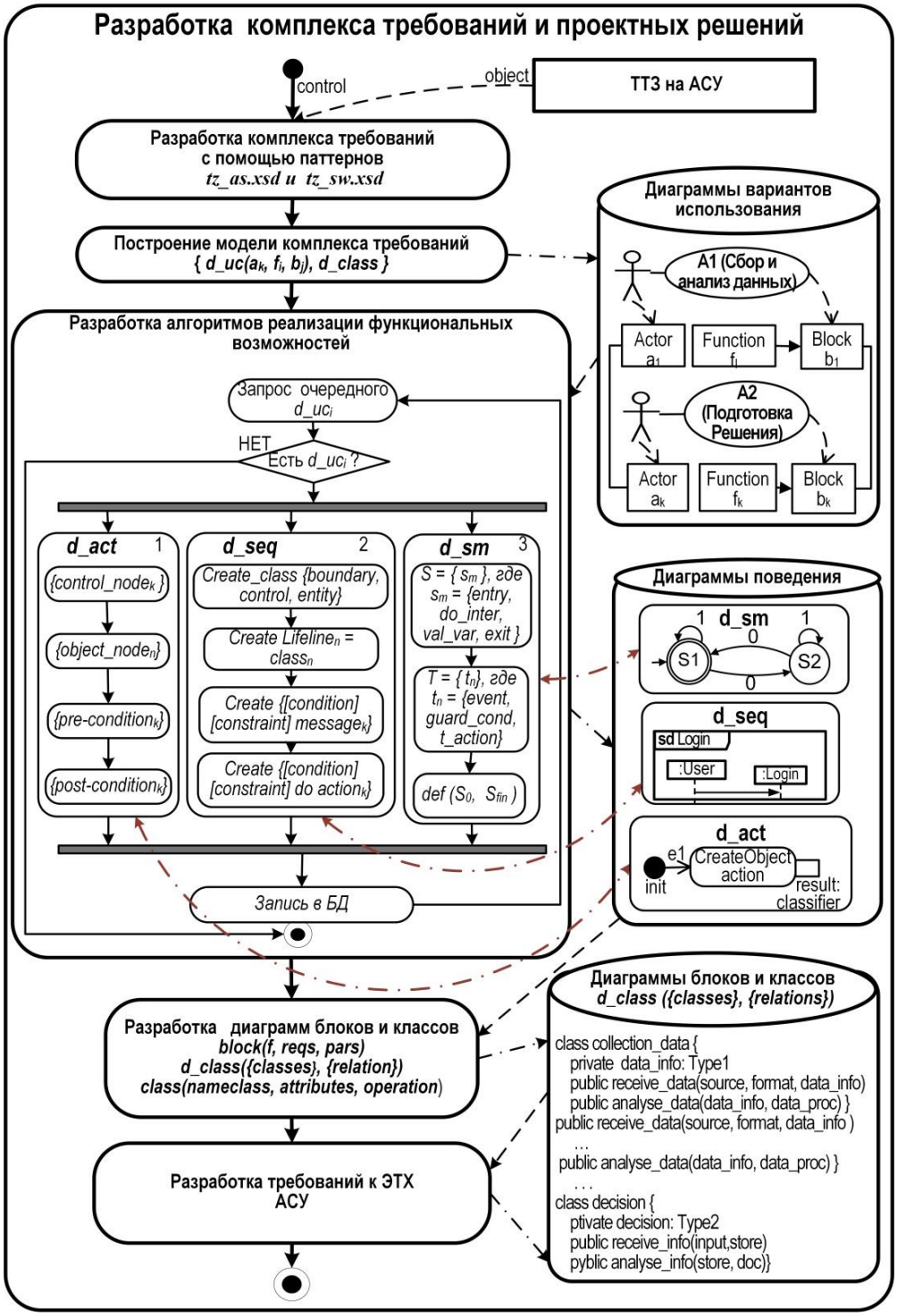

Рисунок 1 – Этапы методики разработки комплекса требований и проектных решений АСУ СОТС

Figure 1 – Stages of the methodology for developing a set of requirements and design solutions for automated control systems

Следующим шагом разработки архитектуры и проектных решений является разработка требований к качеству реализации каждой функции на основе онтологии «Модель качества программно-технических систем» [7]. В соответствии с данной

онтологией характеристики качества АСУ СОТС сведены в восемь групп: функциональная пригодность, производительность, надежность, совместимость, удобство использования, удобство сопровождения, защищенность, переносимость. Наиболее важными эксплуатационными характеристиками АСУ СОТС являются: *надежность*, *защищенность* и *производительность* (Рисунок 2).

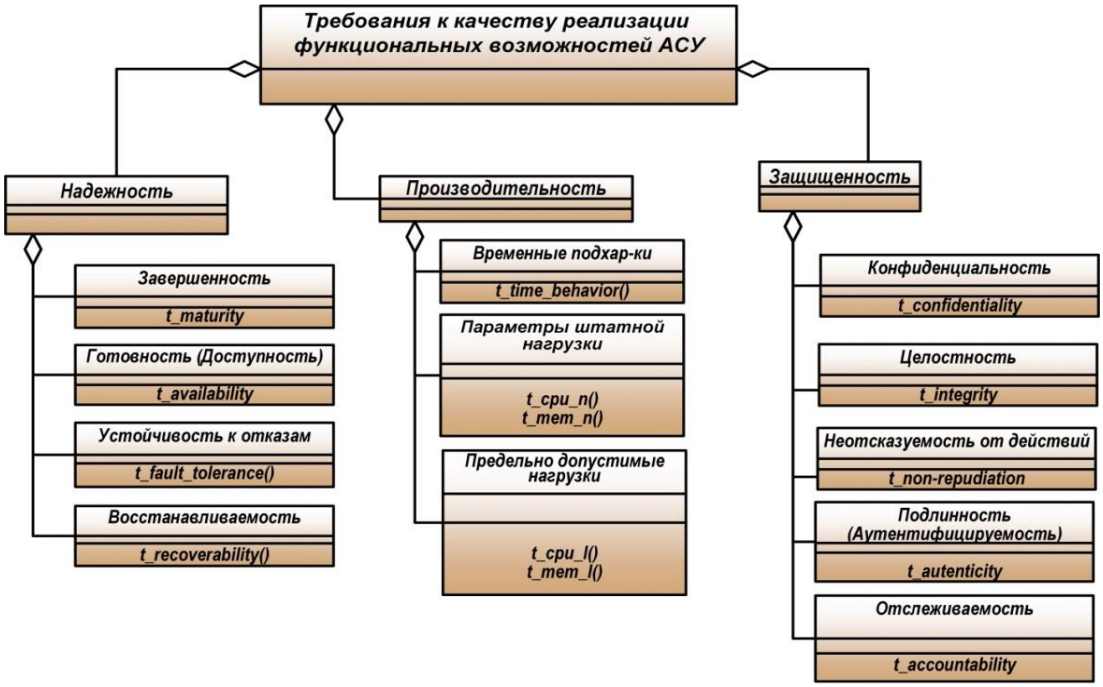

Рисунок 2 – Онтология «Требования к качеству реализации функциональных возможностей АСУ СОТС»

Figure 2 – «Requirements to Realization Quality of Functional Abilities for automated systems» Ontology

Для задания требований к характеристике качества «надежность» используются следующие показатели характеристик качества (ПХК) и соответствующие им функции: t_maturity – для ПХК «завершенность», v_fault tolerance – для ПХК «отказоустойчивость», v_recoverability – для ПХК «восстанавливаемость», t_availability – для ПХК «готовность».

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. Для задания и последующей оценки характеристики качества АСУ СОТС «защищенность» используются следующие ПХК: конфиденциальность, целостность, неподдельность, отслеживаемость, подлинность.

Полученные в результате комплекс требований и проектные решения АСУ СОТС представляют собой иерархически организованную совокупность UML диаграмм вариантов использования, поведения и классов:

$M = \{d\_uc[i, j]\}, i=1,...,I, j=1,..., J,$

где $d\_uc[i,j] = <actor[n,m], func[k,l], module[p,q]>$ – диаграммы вариантов использования,

actor[n,m] – субъекты, инициирующие выполнение функций управления,

func[k,l] – функции АСУ, реализующие варианты использования,

module[p,q] – модули (компоненты) АСУ, выполняющие определенные функции управления.

Функции АСУ ОТС описываются следующим образом:

func[k,l] (Subj[n,m]({p_s}), Obj_j[f,g]({p_o}), Method[r,q](p_m)),

где *Subj[n,m]({p_s})* – должностные лица органов управления, имеющие атрибуты *{p_s},*

Obj_j[f,g]({p_o}) – объекты управления, имеющие атрибуты {p_o},

Method [r,q](p_m)) – методы выполнения функции, обладающие атрибутами (p_m).

Для хранения, валидации, верификации и применения таких метаграфовых моделей, целесообразно использовать средства графовых баз данных. Примеры реализации такого подхода представлены в [8, 9]. В [8] описан метод валидации математических моделей, представленных ориентированными взвешенными знаковыми графами, с использованием алгоритма эффективных управлений. Модель валидируется посредством анализа спектральных свойств матрицы смежности графа, представленной нечеткой когнитивной картой. В [9] для верификации графовой модели, описывающей функциональные требования к проектируемой системе, предложен алгоритм, реализованный на языке логического программирования Пролог. В следующем разделе статьи представлены методы и средства валидации и верификации моделей требований и проектных решений, основанные на формальной спецификации атрибутивной графовой модели с использованием языка Cypher в базе данных Neo4j.

Методы и средства верификации формальной модели комплекса требований к АСУ ОТС в среде СУБД Neo4j

Созданная в среде графического редактора UML модель с помощью модуля преобразования и загрузки переводится в формат графовой модели данных и загружается в базу данных, функционирующую под управлением СУБД Neo4j [10]. Проверка полноты и корректности графовой модели осуществляется с помощью программного компонента верификации, включающего три модуля: формирования тестовых запросов, их выполнения и анализа полученных результатов (Рисунок 3).

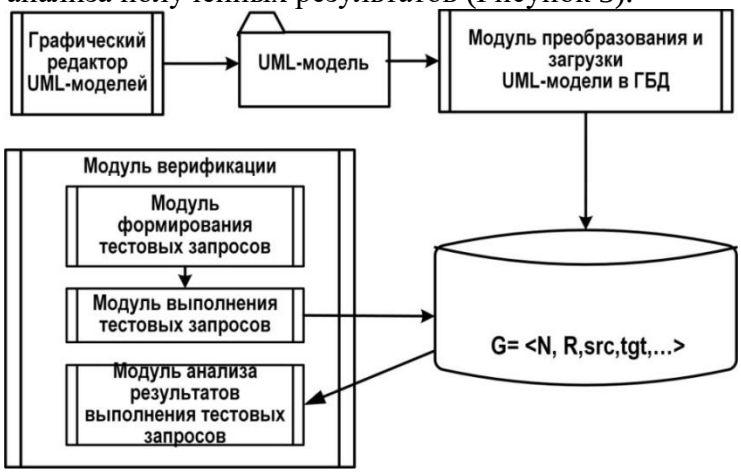

Рисунок 3 – Процесс загрузки и верификации UML модели КТ в среде графовой базы данных Neo4j

Figure 3 – The process of loading and verifying the UML requirements model in the Neo4j graph database environment

Графовая модель данных, в формат которой преобразуются UML модели комплекса требований и проектных решений, представляет собой атрибутивный многореляционный граф следующего вида [11, 12]:

$G = <N, R, src, tgt, l, \lambda, \tau >$, где:

N – узлы графа G,

R – ребра графа G,

$src: R \rightarrow N$ – функция, сопоставляющая каждому ребру R узел, из которого оно выходит,

$tgt: R \rightarrow N$ – функция, сопоставляющая каждому ребру R узел, в который оно входит,

$\lambda: N \rightarrow 2^L$ – функция, которая сопоставляет каждому идентификатору узла набор ролей L, которые может выполнять узел,

$\tau: R \rightarrow T$ – функция, которая сопоставляет каждому идентификатору ребра его тип T,

$l: N \times K \rightarrow V_N$ – функция, определяющая значения атрибутов узлов графа, K – имена атрибутов, V_N – значения атрибутов,

$p: R \times K \rightarrow V_R$ – функция, определяющая значения атрибутов ребер графа, K – имена атрибутов, V_R – значения атрибутов.

При загрузке UML модели в графовую базу данных элементы диаграмм преобразуются в узлы графа, а связи между ними – в ребра соответствующего графа. Ребра и узлы наследуют типы и атрибуты элементов UML модели, из которых они сформированы. Пример представления диаграммы варианта использования, описывающей порядок решения расчетной задачи «Задача_1» в виде атрибутивного графа G, приведен на Рисунке 4.

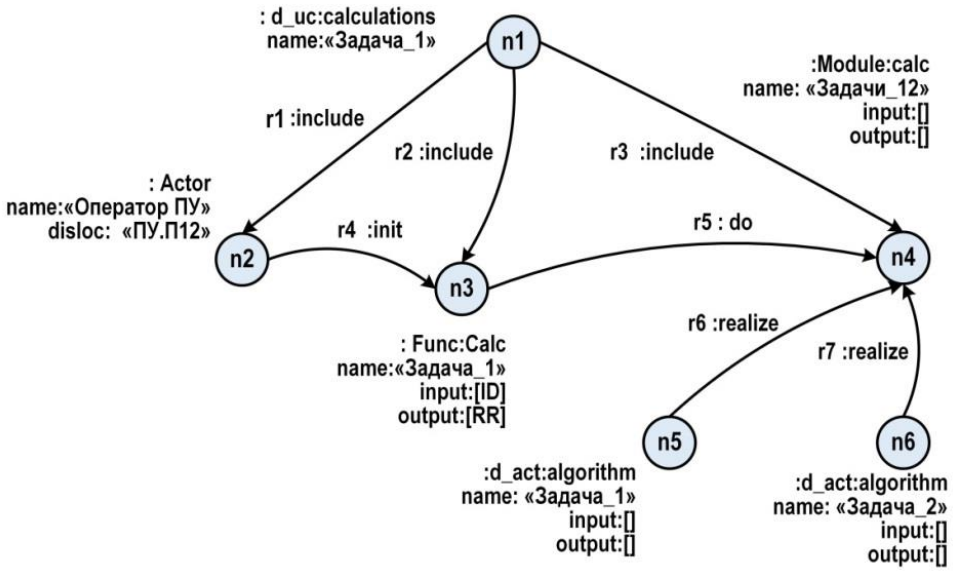

Рисунок 4 – Представление диаграммы варианта использования в виде атрибутивного графа

Figure 4 – Representation of the use case diagram in the form of an attribute graph

Для создания, хранения и обработки данных в среде СУБД Neo4j используется язык Cypher, формальная семантика которого включает отношение соответствия $(|=)$ и две функции $p()$ и $\pi()$ и выражается в виде формулы: $(p(u^*) \subset G) |= \pi(u)$. Функция или

шаблон π(u) задает параметры, определяющие выбор пути (путей) $p(u^*)$ и критерии для принятия решения. Отношение соответствия $(|=)$ проверяет, удовлетворяет ли путь $p(u^*)$ в графе G шаблону, соответствующему функции $\pi(u)$, при присвоении свободным переменным значений u.

Процесс верификации и валидации модели КТ заключается в решении следующих задач:

1) представление требований к характеристикам качества КТ в виде типовых тестовых запросов к базе данных;

2) выполнение тестовых запросов к базе данных;

3) оценивание характеристик качества комплекса требований на основе анализа результатов выполнения тестовых запросов.

Шаблон для формирования запросов к узлам представляет собой кортеж:

$\chi = <a, L^*, l^* >$, где:

$a \in N \cup \{nil\}$ – идентификатор узла,

$L^* \subset L$ – набор типов узлов,

l^* – значения атрибутов узлов.

Шаблон (pattern) для формирования запросов к ребрам представляет собой кортеж следующего вида:

$\rho = <d, a, T, P, I >$ где:

$d \in \{\rightarrow, \leftarrow, \leftrightarrow\}$ спецификация направления связи;

$a \in N \cup \{nil\}$ – имя ребра,

$T^* \subset T$ – типы ребер;

p^* – значения атрибутов ребер;

I – либо nil, либо (m, n) $m, n \in N \cup \{nil\}$.

Модель качества комплекса требований состоит из характеристик комплекса требований в целом и характеристик отдельных требований. Комплекс требований к АСУ СОТС в целом должен обладать следующими характеристиками качества: полнотой, непротиворечивостью, неизбыточностью, системностью [7].Свойство *полноты* КТ характеризует степень учета в нем требований к функциональным возможностям и эксплуатационно-техническим характеристикам, необходимым для ее применения по назначению, удовлетворяющим требованиям нормативно-технических и регламентирующих документов и соответствующим ожиданиям конечных пользователей.

Непротиворечивость КТ характеризует согласованность формулировок всех требований между собой, а также отсутствие противоречий с требованиями нормативно-технических и других регламентирующих документов. *Неизбыточность* КТ означает однократность формулировки каждого требования и ограничения, отсутствие семантических пересечений между разными требованиями, невыводимость одних требований из других. Системность характеризует качество КТ с точки зрения целостности представления, оптимальности детализации, глубины и точности описания взаимосвязанных требований.

Характеристики качества отдельных требований разбиты на две группы: внутренние и внешние. К внутренним характеристикам качества отдельного требования относятся: внутренняя полнота, корректность, однозначность. Внешними характеристиками качества являются: прослеживаемость, проверяемость и модифицируемость.

Оценка корректности алгоритмов реализации функциональных возможностей АСУ осуществляется посредством выполнения запросов к графовой базе данных. Тестовые запросы на языке Cypher генерируются на основе соответствующих правил корректности диаграмм поведения, с помощью которых эти алгоритмы описаны. В Таблице 1 представлены примеры тестовых запросов на языке Cypher и спецификаций правил корректности на языке OCL, на основе которых они сгенерированы.

Таблица 1 – Правила корректности описания моделей требований
Table 1 – Rules for the correctness of the description of requirements models

№	Описание	Спецификация на OCL	Запрос на Cypher
1.	Модель имеет один начальный узел	*self.uslnode->selectByType(InitialNode) ->size()=1*	*MATCH (a:uml_InitialNode) RETURN count(a)=1*
2.	Модель имеет, по крайней мере, один заключительный узел	*self.uslnode->selectByType(FinalNode) ->size() >= 1*	*MATCH (a:uml_InitialNode) WITH collect(a) AS a, count(a) AS n WHERE not n=1 RETURN a*
3.	Модель имеет не менее одного шаг действия	*self.uslnode->selectByKind(FlowStep) ->size()>=1*	*MATCH (a:uml_OpaqueAction) RETURN count(a)>=1*
4.	Начальный узел имеет одно исходящее ребро потока и не имеет никаких входящих ребер потока	*(self.flowedge->select(t:FlowEdge|t.source.oclIsTypeOf (InitialNode))->size()=1)and (self.flowedge ->select (b:FlowEdge|(b.source.oclIsTypeOf(InitialNode)) and (b.oclIsTypeOf(BasicFlowEdge)))->size()=1) and (self.flowedge->select(t:FlowEdge| t.target.oclIsTypeOf(InitialNode))->size()=0)*	*MATCH p=(a:uml_InitialNode)-[d:uml_ControlFlow]->(b) WITH collect(p) AS p, count(d) AS d WHERE not d=1 RETURN p MATCH p=(a:uml_InitialNode)<-[d:uml_ControlFlow]-(b) RETURN p*
5.	Конечный узел имеет одно входящее ребро потока и не имеет исходящих ребер потока	*self.uslnode->selectByType(FinalNode)-> forAll (f:FinalNode|(self.flowedge-> select(e:FlowEdge|e.target=f) ->size() =1) and (self.flowedge->select (e: FlowEdge|e.source=f)->size()=0))*	*MATCH (a:uml_ActivityFinalNode) WHERE not ()-[:uml_ControlFlow]->(a) RETURN a MATCH p=(a:uml_ActivityFinalNode)-[:uml_ControlFlow]->(b) RETURN p*

№	Описание	Спецификация на OCL	Запрос на Cypher
6.	Узел принятия решений имеет одно входящее ребро потока и не менее двух исходящих ребер потока	*d:DecisionNode|(self.flowedge->select(e: FlowEdge|e.target=d)->size()=1) and (self.flowedge->select(e:FlowEdge|e. source=d)->size()>=2)*	*MATCH p=(b)-[d:uml_ControlFlow]->(a:uml_DecisionNode) WITH a, collect(p) AS p, count(d) AS d WHERE not d=1 RETURN p*

Оценивание характеристики «*внутренняя полнота*» требования к функциональной возможности осуществляется посредством проверки наличия в описывающей его диаграмме поведения всех обязательных элементов. Характеристика «*корректность*» оценивается на предмет правильности описания алгоритма решения задачи, с точки зрения пользователя и структурных ограничений диаграммы.

Проверка характеристики КТ *полнота* осуществляется посредством его сравнения с эталонным профилем ТЗ на разработку АСУ СОТС, построенным на основе требований ГОСТ РВ 15.201-2003, ГОСТ 34.602-89, ГОСТ ИСО/МЭК 25010-2015. В случае отсутствия в комплексе требований к АСУ СОТС какого-либо требования, разработчику технического задания для устранения обнаруженного дефекта предлагаются три способа: определить требование, указать явно его отсутствие, определить этап и процесс ЖЦ, на котором оно будет задано.

Проверка характеристики комплекса требований *непротиворечивость* осуществляется на предмет наличия несогласованности и противоречий, примерами которых являются:

– использование терминов, характеристик, единиц измерения, не соответствующих принятой терминологии, тезаурусу и нормативно-техническим документам;

– нарушение представления родовидовых, пространственно-временных и причинно-следственных отношений;

– не соответствие требований к эксплуатационно-техническим характеристикам системы (надежности, производительности, защищенности, совместимости и др.) требованиям нормативно-технических и регламентирующих документов.

Для проверки характеристики комплекса требований «*неизбыточность*» выполняются следующие виды тестирования:

– поиск двух и более одинаковых требований;

– обнаружение смысловых пересечений между разными требованиями посредством поиска шаблонов (подграфов), которые входят в другие шаблоны (графы и подграфы);

– формально-логический анализ атрибутивных графовых моделей с целью обнаружения возможности получения одних требований из других.

Систематичность формальной модели комплекса требований контролируется и обеспечивается посредством представления требований в виде иерархической структуры с четко выделенными атрибутами и точным описанием взаимосвязей и зависимостей между ними. Для оценивания соответствия модели комплекса требований представленным выше показателям качества разработаны тестовые запросы. При их разработке использованы библиотеки APOC и GDS (Graph Data Science) [10, 13, 14], содержащие большое количество процедур и функций создания, преобразования, фильтрации, интеграции и анализа графовых данных. Результатом выполнения

описанных выше процедур является корректная формальная модель комплекса требований к АСУ СОТС.

Заключение

Представленные в статье методы и средства обладают следующими отличительными особенностями. Разработка комплекса требований и проектных решений АСУ СОТС выполняется с помощью специальных шаблонов, учитывающих специфику алгоритмов их реализации и построенных на основе онтологии «Качество программно-технических систем». Высокое качество формальной модели комплекса требований и проектных решений обеспечивается посредством их валидации и верификации в среде графовой базы данных Neo4j с помощью разработанных тестовых запросов. Методы и средства валидации и верификации комплекса требований и проектных решений АСУ СОТС используют такие преимущества графовых моделей и баз данных, как проверенный математический и алгоритмический аппарат, высокую производительность и масштабируемость, адекватность и наглядность визуального представления.

СПИСОК ИСТОЧНИКОВ

1. Selby R.W. Software Engineering: Barry W. Boehm's Lifetime Contributions to Software Development, Management, and Research. *Wiley-IEEE Computer Society Press*. June 2007.
2. Shevchenko N. An Introduction to Model-Based Systems Engineering (MBSE). *Carnegie Mellon University's Software Engineering Institute*. Доступно по: http://insights.sei.cmu.edu/blog/introduction-model-based-systems-engineering-mbse/ (дата обращения: 24.09.2021).
3. Буздалов Д.В., Зеленов С.В., Корныхин Е.В., Петренко А.К., Страх А.В., Угненко А.А., Хорошилов А.В. Инструментальные средства проектирования систем интегрированной модульной авионики. *Труды ИСП РАН*. 2014;26(1):201–230.
4. Kildishev D.S., Khoroshilov A.V. Formalizing metamodel of Requirements Management System. *Trudy ISP RAN/Proc. ISP RAS*. 2018;30(5):163–176. DOI: 10.15514/ISPRAS-2018-30(5)-10.
5. Самохвалов Э.Н., Ревунков Г.И., Гапанюк Ю.Е. Использование метаграфов для описания семантики и прагматики информационных систем. *Вестник МГТУ им. Н.Э. Баумана. Сер. «Приборостроение»*. 2015;1(100):83–99.
6. Наместников А.М., Гуськов Г.Ю., Филиппов А.А.. Интеллектуальный анализ проектов программных систем на основе онтологического подхода. *Автоматизация процессов управления*. 2020;1(59):75–85.
7. Самонов А.В. Методы и средства разработки автоматизированных информационных систем на основе онтологии «Управление качеством программно-технических комплексов». *Труды ИСП РАН*. 2019;31(5):165–182.
8. Васильев В.С., Целых А.Н., Целых Л.А. Метод валидации графовых моделей на основе алгоритма эффективных управлений. *Труды учебных заведений связи*. 2020; 6(3):58–65. DOI:10.31854/1813-324X-2020-6-3-58-65.
9. Бурляева Е.В., Кононенко В.В., Корнюшко В.Ф., Разливинская С.В. Алгоритмы и программа верификации функциональных моделей. *Программные продукты и системы*. 2021;34(2):221–229. DOI: 10.15827/0236-235X.134.221-229.
10. Neo4j Graph Platform. Доступно по: https://neo4j.com/developer/graph-platform (дата обращения: 10.09.2021).

11. Francis N, Green A., Guagliardo P. Formal Semantics of the Language Cypher Version 1.1: core read-only fragment. Доступно по: https://arxiv.org/pdf/1802.09984.pdf (дата обращения: 14.09.2021).
12. Marton J., Szárnyas G., Varró D. Formalising open Cypher Graph Queries in Relational Algebra. Доступно по: https://arxiv.org/pdf/1705.02844.pdf (дата обращения: 9.09.2021).
13. Apache TinkerPop™ is a graph computing framework for both graph databases (OLTP) and graph analytic systems (OLAP). Доступно по: http://tinkerpop.apache.org (дата обращения: 14.09.2021).
14. Нидхем М., Ходлер А. *Графовые алгоритмы. Практическая реализация на платформах Apache Spark и Neo4j*.; пер. с англ. В.С. Яценкова. М.: ДМК Пресс; 2020.

REFERENCES

1. Selby R.W. Software Engineering: Barry W. Boehm\'s Lifetime Contributions to Software Development, Management, and Research. *Wiley-IEEE Computer Society Press.* June 2007.
2. Shevchenko N. An Introduction to Model-Based Systems Engineering (MBSE). *Carnegie Mellon University\'s Software Engineering Institute Blog*. Available at: http://insights.sei.cmu.edu/blog/introduction-model-based-systems-engineering-mbse/ (accessed: 24.09.2021).
3. Buzdalov D.V., Zelenov S.V., Kornyhin E.V., Petrenko A.K., Strah A.V., Ugnenko A.A., Horoshilov A.V. Instrumental'nye sredstva proektirovaniya sistem integrirovannoj modul'noj avioniki. *Trudy ISP RAN = Proceedings of ISP RAS*. 2014;26(1):201–230.
4. Kildishev D.S., Khoroshilov A.V. Formalizing metamodel of Requirements Management System. *Trudy ISP RAN/Proc. ISP RAS*. 2018;30(5):163–176. DOI: 10.15514/ISPRAS-2018-30(5)-10.
5. Samohvalov E.N., Revunkov G.I., Gapanyuk Yu.E. Ispol'zovanie metagrafov dlya opisaniya semantiki i pragmatiki informacionnyh sistem. ISSN 0236-3933. *Vestnik MGTU im. N.E. Baumana. Ser. «Priborostroenie» = Herald of the Bauman Moscow State Technical University. Series Instrument Engineering*. 2015;1(100):83–99.
6. Namestnikov A.M., Gus'kov G.Yu., Filippov A.A. Intellektual'nyj analiz proektov programmnyh sistem na osnove ontologicheskogo podhoda. *Avtomatizaciya processov upravleniya = Automation of Control Processes*. 2020;1(59):75–85.
7. Samonov A.V. Metody i sredstva razrabotki avtomatizirovannyh informacionnyh sistem na osnove ontologii «Upravlenie kachestvom programmno-tekhnicheskih kompleksov». *Trudy ISP RAN = Proceedings of ISP RAS*. 2019;31(5):165–182.
8. Vasil'ev V.S., Celyh A.N., Celyh L.A. Metod validacii grafovyh modelej na osnove algoritma effektivnyh upravlenij. *Trudy uchebnyh zavedenij svyazi = Proceedings of Telecommunication Universities* 2020; 6(3):58–65. DOI:10.31854/1813-324X-2020-6-3-58-65.
9. Burlyaeva E.V., Kononenko V.V., Kornyushko V.F., Razlivinskaya S.V. Algoritmy i programma verifikacii funkcional'nyh modelej. *Programmnye produkty i sistemy = SOFTWARE & SYSTEMS*. 2021;34(2):221–229. DOI: 10.15827/0236-235X.134.221-229.
10. Neo4j Graph Platform. Available at: https://neo4j.com/developer/graph-platform (accessed: 10.09.2021).
11. Francis N, Green A., Guagliardo P. Formal Semantics of the Language Cypher Version 1.1: core read-only fragment. Available at: https://arxiv.org/pdf/1802.09984.pdf (accessed: 14.09.2021).

12. Marton J., Szárnyas G., Varró D. Formalising open Cypher Graph Queries in Relational Algebra. Available at: https://arxiv.org/pdf/1705.02844.pdf (accessed: 9.09.2021).
13. Apache TinkerPop™ is a graph computing framework for both graph databases (OLTP) and graph analytic systems (OLAP). Available at: http://tinkerpop.apache.org (accessed:14.09.2021).
14. Needham M., Hodler A. *Grafovye algoritmy. Prakticheskaya realizaciya na platformah Apache Spark i Neo4j*.; per. s angl. V.S. Yacenkova. M.: DMK Press; 2020.

ИНФОРМАЦИЯ ОБ АВТОРАХ / INFORMATION ABOUT THE AUTHORS

Баев Алексей Владимирович, научный сотрудник, Военно-космическая академия имени А.Ф. Можайского, Санкт-Петербург, Российская федерация
e-mail: *baih@mail.ru*

Alexey V. Baev, Research Scientist A.F.Mozhaiskiy Military Space Academy, Saint Petersburg, Russian Federation

Самонов Александр Валерьянович, кандидат технических наук, доцент. Старший научный сотрудник Военно-космическая академия имени А.Ф. Можайского, Санкт-Петербург, Российская федерация
e-mail: *a.samonov@mail.ru*
ORCID: 0000-0002-0390-4481

Alexander V. Samonov - Phd In Technical Sciences, Associate Professor, Senior Research Scientist A.F.Mozhaiskiy Military Space Academy, Saint Petersburg, Russian Federation

Сафонов Вадим Максимович, кандидат технических наук, начальник научно-исследовательской лаборатории. Военно-космическая академия имени А.Ф. Можайского, Санкт-Петербург, Российская федерация
e-mail: *safonov-vm@mail.ru*

Vadim M. Safonov, Phd In Technical Sciences, Head Of The Laboratory A.F.Mozhaiskiy Military Space Academy, Saint Petersburg, Russian Federation

Статья поступила в редакцию 12.10.2021; одобрена после рецензирования 15.12.2021; принята к публикации 21.12.2021

The article was submitted 12.10.2021; approved after reviewing 15.12.2021; accepted for publication 21.12.2021