ПРИКЛАДНЫЕ ОНТОЛОГИИ ПРОЕКТИРОВАНИЯ

УДК 004.896 DOI: 10.18287/2223-9537-2021-11-4-450-463

Сквозное проектирование автоматизированных систем на основе онтологий

В.Н. Негода, А.А. Куликова

Ульяновский государственный технический университет, Ульяновск, Россия

Аннотация

В настоящее время проводится большое количество исследований по применению онтологий при разработке автоматизированных систем. Онтологическое моделирование позволяет повысить эффективность процесса разработки программного обеспечения. Разделение труда при разработке автоматизированных систем способствует появлению семантических разрывов между фазами проектного процесса. Практика использования широкого спектра артефактов проектирования, пригодных для повторного применения, является фактором риска нарушения концептуальной целостности, непротиворечивости и полноты разрабатываемых проектных решений. В статье предлагаются: новый подход к онтологическому моделированию автоматизированных систем, который обслуживает проектный процесс их разработки на всех стадиях проектирования вплоть до реализации; новая структура метаданных онтологических спецификаций автоматизированных систем, которая позволяет учитывать семантически важные сущности и особенности разрабатываемой системы. Показано, что применение данного подхода значительно снижает трудозатраты при проектировании сложной автоматизированной системы в условиях изменения требований на различных стадиях создания системы. Применение онтологических моделей в проектном процессе способствует повышению концептуальной целостности, непротиворечивости и полноты разрабатываемых проектных решений.

Ключевые слова: *автоматизация проектирования, автоматизированные системы, сквозное проектирование, вопросно-ответное моделирование, онтология проекта.*

Цитирование: *Негода, В.Н.* Сквозное проектирование автоматизированных систем на основе онтологий / В.Н. Негода, А.А. Куликова // Онтология проектирования. – 2021. – Т.11, №4(42). - С.450-463. – DOI: 10.18287/2223-9537-2021-11-4-450-463.

Введение

Проектирование сложных автоматизированных систем (АС) в соответствии с широко применяемыми стандартами [1-2] представляет собой многоэтапный процесс, в который вовлекаются различные группы специалистов, многообразные технологии проектирования и инструментальные средства поддержки исследования объекта автоматизации. Этот процесс включает: анализ требований, формирование и документирование проектных решений, моделирование и прототипирование, программирование, тестирование и верификацию. Разделение труда и разнообразие применяемых технологий и инструментальных средств способствует возникновению *информационных (семантических) разрывов* в ходе проектирования [3-6], что приводит к нарушениям *непротиворечивости*, *концептуальной целостности* и *полноты* всей совокупности проектных решений.

Подобные нарушения чаще всего возникают в ходе интерпретации проектных спецификаций, передающихся от одних проектировщиков другим. Характерными свойствами спецификаций проектных решений АС являются *многокомпонентность* и *гетерогенность* – коли-

чество сущностей, фигурирующих в различных проектных диаграммах и их описаниях, обычно велико, и эти сущности довольно разнородны. Применение технологий онтологического моделирования в специфицирование проектных решений позволяет автоматизировать интерпретацию разнородных спецификаций и создаёт условия для сквозного проектирования АС с существенно меньшей вероятностью возникновения информационных разрывов [7].

В работе [8] представлена технология онтологического моделирования систем логического управления (СЛУ), которые являются распространённым классом АС управления технологическими процессами [9]. В ходе разработки и исследования этой технологии авторами обнаружены возможности обеспечить сквозное проектирование АС за счёт использования композиции трёх видов онтологических моделей (ОМ): онтологии требований, онтологии процесса проектирования и онтологии программной реализации проектных решений. Такая возможность наглядно проявляется в примере автоматической генерации части программного кода СЛУ на основе совокупности спецификаций этих трёх видов ОМ.

В настоящей работе выполнено теоретическое обобщение результатов экспериментов по применению онтологий в проектировании АС, положенных в основу разработки подхода к сквозному проектированию АС.

1 Основные положения

Проектный процесс строится на основе механизмов *проектного мышления* (*Design Thinking, DT*-подход) [10-12] и *вопросно-ответного моделирования* рассуждений проектировщика с использованием технологий онтологического моделирования [13] и прототипов проектных решений, представляющих собой результаты опыта предыдущих проектов либо текущих проектных экспериментов.

Оба механизма моделируют рассуждения проектировщика с ориентацией на *мотивационно-целевую составляющую* требований к разрабатываемой системе. Последовательная реализация *DT*-подхода и вопросно-ответный анализ артефактов, генерируемых на каждом из его этапов (текстов требований, формулировок проектных задач, описаний идей их решения и др.) постепенно снижает неопределённость проектной ситуации и приводит к достижению проектировщиком такого понимания проекта АС, которое является необходимым и достаточном для его успешного завершения.

Проектный процесс создания АС включает несколько этапов (см. рисунок 1).

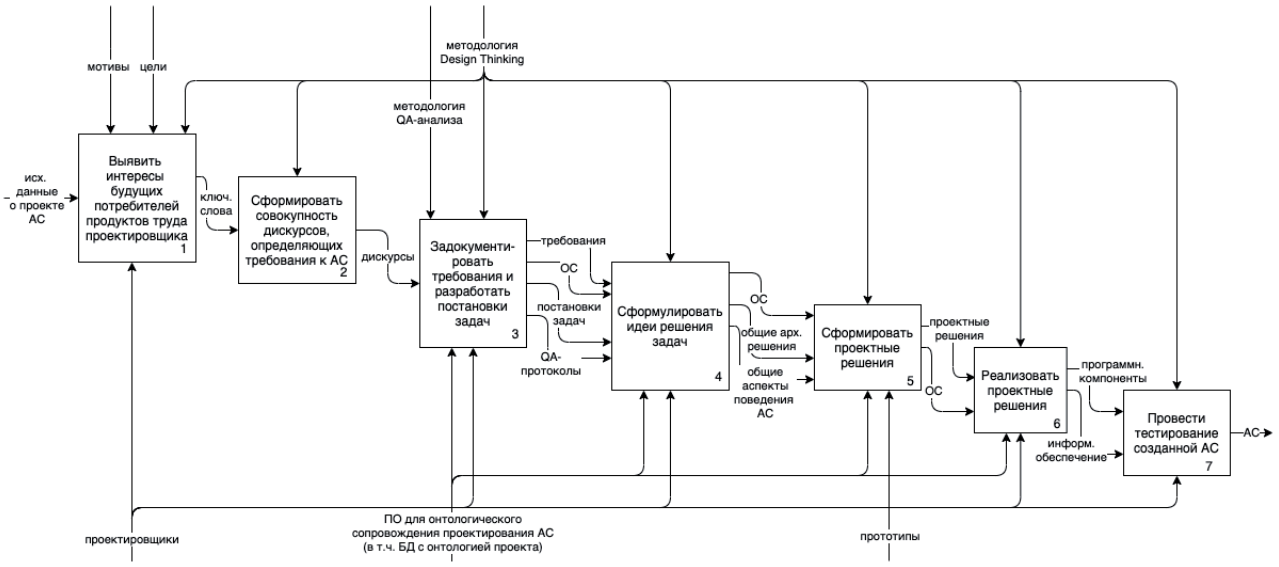


Рисунок 1 – IDEF0-диаграмма проектного процесса разработки АС

1.1 Выявление интересов будущих потребителей продуктов труда проектировщика

На этом этапе системный аналитик формирует исходное представление о проекте в вербальной форме – например, в формате *ключевых слов*, указывающих на обнаруженные данные о проекте АС, которую требуется разработать, – и фокусируется на неопределённости исходных спецификаций задач проекта. Тем самым формируется целеориентирующая основа для вопросно-ответного моделирования в ходе анализа требований. Онтологические спецификации (ОС), возникающие при этом, охватывают *мотивы* и *цели* пользователей создаваемой АС, а также предположения о её свойствах, благодаря которым планируется достичь поставленных целей [14].

1.2 Формирование совокупности дискурсов, определяющих требования к АС

Далее формируется совокупность дискурсов D, представляющих собой короткие тексты на естественном языке, определяющие требования к АС в контексте мотивов и целей её заказчика. Каждый дискурс содержит исходную неопределённость, которая может быть снижена за счёт генерирования вопросов Q и формулирования ответов на них A. При этом каждый ответ A_n будет также содержать неопределённость, которая будет меньше исходной и также может быть снижена за счёт генерирования новых вопросов и формулирования новых ответов. В результате такого вопросно-ответного анализа формируется QA-протокол в виде дерева вопросов и ответов, который используется в качестве основы для онтологического специфицирования АС и формирования онтологии проекта O^P. Особенности методологии QA-моделирования, а также примеры задач по разработке АС, в ходе которых успешно применяется вопросно-ответный подход, представлены в [15].

QA-протокол позволяет отследить логику совместных рассуждений представителей заказчика и проектировщиков, выявить требования к АС, ассоциировать вытекающие из требований проектные задачи с единицами опыта (существующими проектными решениями, пригодными для повторного использования), а также снабдить ОС проекта мотивами конечных потребителей установлением отношений сущностей мотивов и целей с сущностями требований. ОС требований образуют основу для трёх видов проектных процедур последующих этапов: документирования требований, разработки постановок задач и верификации проектных решений.

1.3 Документирование требований и разработка постановок задач

На данном этапе происходит осмысление требований, представленных в форме совокупности дискурсов D, на основе вопросно-ответного моделирования. При этом строятся такие формулировки требований, которые устраивают заказчика и исполнителя проекта АС по степени достижения целей и уровням рисков осуществимости в рамках согласованных ограничений бюджета и времени разработки. Результаты QA-анализа служат основой для документирования требований согласно общепринятым стандартам, а также разработки совокупности постановок задач, реализация которых обеспечивает создание АС.

ОС содержат отношения между требованиями и постановками задач, которые позволяют контролировать полноту покрытия всей совокупности требований сущностями постановок задач. Высокий уровень автоматизации функции такого контроля обеспечивается за счёт интерпретируемости ОС, последовательно наращиваемых на каждом этапе.

Совокупность постановок задач является основой для проектного менеджмента – формирования команд исполнителей, распределения задач, контроля исполнения. Интерпретируемость ОС обеспечивает контролируемость покрытия требований содержанием документов, планирующих проектный процесс, и спецификацию проектных решений.

1.4 Формулировка идей решения задач

Идеи решения задач образуют концептуальную основу проекта АС в виде совокупности архитектурных решений, общей структурно-функциональной организации и общих аспектов поведения, которые фиксируются в спецификациях онтологии проекта, связанных отношениями вида «быть основанным», «быть обусловленным», «являться детализацией» со спецификациями предыдущих этапов проектирования.

1.5 Формирование проектных решений

Проектные решения формируются с использованием прототипов, в качестве которых могут представляться спецификации алгоритмов, агрегатов данных, процедур обработки из предыдущих проектов команды разработчиков и доступных библиотек шаблонов проектирования, классов и других программно-информационных ресурсов, рассматриваемых в рамках *DT*-подхода в качестве базы опыта [16, 17]. В ОС на этом этапе важную роль играет отношение «быть реализованным». Это отношение фиксирует факты реализации требований, идей, алгоритмов и других проектных решений через программно-информационные прототипы.

1.6 Реализация проектных решений

Реализация всей совокупности проектных решений производится через связанную совокупность программных компонентов и исходное информационное обеспечение, достаточное для организации тестирования АС.

1.7 Тестирование созданной АС

ОС проекта организуются таким образом, что на их основе может быть сгенерирован тестируемый прототип АС, созданный в рамках *DT*-подхода. Связывание сущностей такого прототипа с ОС порождает свойство концептов «быть материализованным» на мета-уровне.

2 Организация сквозного проектирования АС

Предлагается модель для специфицирования проектных решений в форме онтологии:

$$O^P = (C, R, F, Pr, Agg, A),$$

где C – множество концептов, R – множество отношений семантических типов, F – множество n функций интерпретации, определяемых как множество отображений:

$$F = \{f_n: C^* \times R^* \to C^{**} \times R^{**} | C^*, C^{**} \subseteq C; R^*, R^{**} \subseteq R\}$$

Pr – множество свойств понятий и отношений: $Pr = \{(pr_i, e_j) | pr \in Pr, e \in C \cup R\}$

Agg – множество агрегатов, A – множество аксиом.

Под концептами понимаются классы понятий CC, характерные для АС, и их экземпляры CI, специфицирующие конкретную систему. Т.е. отношение «быть экземпляром класса общего понятия в конкретной АС» является структурообразующей основой, декомпозирующей множество концептов и отношений:

$$C = CC \cup CI,\ R = R^{CC} \cup R^{CI} \cup R^{CCI},$$

где $R^{CC} \subseteq CC \times CC$, $R^{CI} \subseteq CI \times CI$, $R^{CCI} \subseteq CC \times CI$.

Дополнительно определены структурообразующие типы концептов и отношений:

$$C = C^I \cup C^J \cup C^D \cup C^S \cup C^T,\ R = R^I \cup R^J \cup R^D \cup R^S \cup R^T,$$

где C^I – концепты, представляющие этапы проектирования, а R^I – отношения непосредственного следования между концептами, добавляемыми в онтологию проекта на каждом этапе проектирования; C^J – концепты, представляющие множество проектных задач, характерных

для процессов проектирования АС, а R^J – отношения обусловленности между концептами, связанными с конкретными проектными задачами; C^D и R^D – концепты и отношения, обслуживающие документирование разрабатываемой АС (например, названия структурных элементов *UML*-диаграмм и типовые связи между такими элементами или названия типовых полей проектной документации, определяемых в соответствии со стандартами); C^S и R^S – концепты и отношения, представляющие спецификации конкретной АС; C^T и R^T – концепты и отношения, определяющие правила верификации проектных решений и тестирования программ, а также поддерживающие итоговую аттестацию АС через комплексное тестирование и выполнение программы испытаний.

Процедуру формирования и модификации ОМ проекта можно разделить на работу с метаданными ОС и работу со спецификациями конкретной АС. Данная процедура выполняется *итеративно*, обеспечивая наращивание ОС на каждом этапе проектного процесса.

2.1 Формирование метаданных онтологии проекта

Формирование метаданных онтологии проекта осуществляется следующей последовательностью шагов.

1) формирование или модификация пространства агрегатов *Agg* – путём выделения следующих групп:
   - этапы проектного процесса *I*, а также производимые на каждом из них артефакты проектирования (рассуждения проектировщика, проектная документация, *UML*-диаграммы, программа автоматизации), подвергающиеся онтологическому специфицированию: $a^I: I \to Agg^*$;
   - проектные задачи, проектные решения и проектные операции;
   - прототипы, представляющие прецеденты *P* в накопленном опыте автоматизации – готовые и подтвердившие свою состоятельность проектные решения, пригодные для повторного использования, которые подвергаются онтологическому специфицированию для ассоциирования с другими сущностями проекта: $a^P: P \to Agg^*$.
2) формирование или модификация множества классов понятий CC^i, характеризующих объект проектирования на этапе проектного процесса *i*. Классы таких понятий соответствуют типичным сущностям АС, например: входы и выходы АС; операции и действия АС; события, возникающие в АС; сообщения о событиях, действиях, значениях входов и выходов; процессы, осуществляемые в ходе функционирования АС и объекта автоматизации; функциональные, параметрические и иные требования; протоколы о ходе функционирования системы и др. Экземпляры CI^i созданных классов связываются с сущностями артефактов проектирования текущего этапа Atf^i (спецификациями требований, UML-диаграммами, исходным кодом программы автоматизации и др.) отношением материализации, которое задаётся функцией: $m^i: CI^i \to Atf^i$.
3) формирование или модификация аксиом A^i, накладывающих ограничения на структуру метаданных ОС: аксиомы описывают возможные типы семантических отношений RT^i между понятиями CI^i различных классов CC^i (при этом определяются конкретные классы, которые могут быть связаны определёнными типами отношений), а также возможность связывания этих классов с определёнными агрегатами *Agg*:

$$ax^R: \{(cc_n, cc_m) | cc \in CC; n, m \in \mathbb{N}; n, m \in [1, |Cl|]\} \to$$
$$\{rt_p | rt \in RT, p \in \mathbb{N}, p \in [1, |RT|]\}$$
$$ax^{Agg}: CC \to Agg.$$

2.2 Формирование ОС АС

Формирование ОС разрабатываемой АС осуществляется следующей последовательностью шагов.

1) формирование экземпляров классов понятий CI^i_j, соответствующих конкретным сущностям АС, включающее два процесса:
   - связывание дискурсов из множества D и результатов их QA-анализа с понятиями C^i_j, онтологии проекта:
   $$l^i: D \times QA^i \to \{(cc^i_j, ci^i_j) | cc \in CC, ci \in CI\} \times \{(ci^i_j, ci^i_d, rt) | d = f(D), rt \in RT\};$$
   - трансформация продуктов онтологического специфицирования, выполненного на предыдущих этапах проектного процесса, O^{i-1} и связывание их с концептами C^i_j в соответствии с результатами вопросно-ответного моделирования QA^i:
   $$t^i: O^{i-1} \times QA^i \to \{(cc^i_j, ci^i_j) | cc \in CC, ci \in CI\} \times \{(ci^{i-1}_j, ci^i_j, r) | ci \in CI\}.$$
2) описание множества свойств понятий Pr и их значений V: в качестве свойств могут быть заданы, например, имена Nm сущностей E АС в конкретном артефакте:
   - термины, используемые в спецификациях требований;
   - имена объектов, например, состояний АС, используемые для создания UML-диаграмм;
   - имена классов, функций, параметров, используемых в исходном коде программы автоматизации:
   $$p: \{(e_n, nm_n) | e \in E, nm \in Nm\} \to \{(pr_m, v_m) | pr \in Pr, v \in V\}$$

   Проектный процесс характерен включением множеств *пространства имён*, что является одной из причин информационных разрывов. Включение пространств имён в ОМ АС способствует снижению когнитивных усилий проектировщиков на осознание того, что разными именами названа одна и та же сущность.
3) описание отношений между понятиями R, которые формируются на основе содержания дискурсов D и связанных с ними QA-протоколов в соответствии с аксиомами A:
   $$r^i: D \times QA \times A \to \{(c_n, c_m, rt)\}.$$
4) агрегация концептов и отношений в соответствии с аксиомами A^i – важное отличие агрегатов от классов заключается в том, что один и тот же элемент ОМ может относиться сразу к нескольким агрегатам:
   $$agg^i: \{e^i | e \in C \cup R\} \times A^i \to \{(e^i, agg) | agg \in Agg\}.$$
5) формирование выборок ОС АС, пригодных для генерирования артефактов проектирования (документации, *UML*, исходного кода) на основе соответствующих агрегатов:
   $$g^i: C^D_i \times R^D_i \times Agg^i \to C^* \times R^*$$

3 Оценка эффектов создания ОМ проекта для СЛУ

Для оценки эффектов описываемого подхода разработана ОМ *СЛУ роботами-плиткоукладчиками*, осуществляющими укладывание плитки на негоризонтальной поверхности. С правой стороны робота-плиткоукладчика находятся манипуляторы укладки плитки на бетонное основание. Чтобы не двигаться по уже уложенным плиткам, робот движется по квадратной спирали по часовой стрелке. На рисунке 2 показана схема возможной траектории движения такого робота, записи в клетках которой имеют следующий формат:

НомерПлитки_НомерВитка: x, y, НаправлениеДвиженияПослеУкладкиПлитки,

где *x, y* – координаты робота в такой системе координат, где единица соответствует одному плиткоместу. Плитки нумеруются с 0.

25_3: -2, 3 →	26_3: -1, 3 →	27_3: 0, 3 →	28_3: 1, 3 →	29_3: 2, 3 →	30_3: 3, 3 ↓
24_3: -2, 2 ↑	09_2: -1, 2 →	10_2: 0, 2 →	11_2: 1, 2 →	12_2: 2, 2 ↓	31_3: 3, 2 ↓
23_3: -2, 1 ↑	08_2: -1, 1 ↑	01_1: 0, 1 →	02_1: 1, 1 ↓	13_2: 2, 1 ↓	32_3: 3, 1 ↓
22_3: -2, -0 ↑	07_2: -1, 0 ↑	00_1: 0, 0 ↑	03_1: 1, 0 ↓	14_2: 2, 0 ↓	33_3: 3, 0 ↓
21_3: -2, -1 ↑	06_2: -1, -1 ↑	05_2: 0, -1 ←	04_2: 1, -1 ←	15_2: 2, -1 ↓	34_3: 3, -1 ↓
20_3: -2, -2 ↑	19_3: -1, -2 ←	18_3: 0, -2 ←	17_3: 1, -2 ←	16_3: 2, -2 ←	35_3: 3, -2 •

Рисунок 2 – Схема траектории движения робота-плиткоукладчика

В качестве инструментальной основы для реализации ОМ выбрана СУБД *Neo4j*, которая базируется на структуре *LPG* (*Labeled Property Graph*) и активно применяется на практике, в том числе для решения задач, связанных с онтологическим моделированием [18].

3.1 Описание ОС СЛУ на мета-уровне

ОМ рассматривается как совокупность ОС, охватывающих этапы проектного процесса: анализ требований, формирование проектных решений и реализация. Множество пространство *агрегатов Agg = {Requirements, Solutions, Development}* представлено с помощью тегов узлов и связей графа в базе данных, где тег *Requirements* имеют спецификации, относящиеся к этапу анализа требований, *Solutions* – к этапу формирования проектных решений и *Development* – к этапу реализации программы автоматизации.

Классы понятий представлены следующими множествами:

- на уровне требований: CC^R = *{Source, Inputs, Outputs, Events, Actions, Conditions}*, где *Source* – описания требований к системе на естественном языке, в том числе дискурсы; *Inputs* и *Outputs* – входы и выходы СЛУ; *Events* – события, возникающие в СЛУ; *Actions* – операции СЛУ; *Conditions* – спецификации элементов технических условий;
- на уровне проектных решений: CC^S = *{States, Transitions, Predicates, Actions}*, где *States* – состояния СЛУ; *Transitions* – переходы между состояниями; *Predicates* – условия перехода между состояниями, *Actions* – операции и действия, которые необходимо выполнить до либо после переходов в соответствии с требованиями;
- на уровне реализации: CC^D = *{Operations, Functions, Parameters}*, где *Operations* – операции, *Functions* – функции, *Parameters* – параметры, использованные в исходном коде программы автоматизации.

В качестве *аксиом* заданы: принадлежность классов понятий определённым агрегатам и обязательное наличие определённых типов семантических связей между классами. Примеры аксиом второго типа на языке логики предикатов имеют вид: *вызывать (Transition, Action)*; *предшествовать (Transition, Action)*; *переходить в (State, State)*; *быть переходом из (Transition, State)*; *быть переходом в (Transition, State)* и др.

ОС уровня проектных решений отчасти наследуют спецификации уровня требования, а спецификации уровня реализации - спецификации уровня проектных решений.

3.2 Использование ОС СЛУ для генерирования артефактов проектирования

ОС проекта на всех уровнях организованы таким образом, что являются пригодными для генерирования таких артефактов проектирования, как фрагменты технического задания, *UML*-диаграммы (в частности, диаграмма состояний), исходный код программы автоматизации СЛУ на языке *C*. Программы автоматизации обработки ОС для генерирования перечисленных артефактов реализованы с помощью *Cypher*-запросов к базе данных *Neo4j* и скриптов на языке *Python*.

Фрагменты технического задания на разработку СЛУ роботами-плиткоукладчиками, сгенерированные на основе ОС уровней требований и проектных решений:

- реализация СЛУ должна предусматривать следующие состояния: завершённость процесса; инициализация; движение вперёд; аварийное состояние; движение вправо; движение назад; движение влево;
- переход из состояния «инициализация» в состояние «аварийное состояние» должен осуществляться при условии превышении времени пребывания в состоянии инициализации;
- переход из состояния «инициализация» в состояние «движение вперёд» должен осуществляться при условии завершения инициализации; после перехода из состояния «инициализация» в состояние «движение вперёд» должно осуществляться движение к начальной точке.

На рисунке 3 показаны ОС уровня проектных решений, визуализированные в виде графа в среде *Neo4j Browser*, на рисунке 4 – сгенерированный на их основе фрагмент диаграммы состояний СЛУ. Для визуализации диаграммы состояний использован инструмент *PlantUML*, позволяющий создавать диаграммы из описаний на формальном языке (формальные спецификации оказалось возможным сгенерировать на основе данных из ОМ проекта).

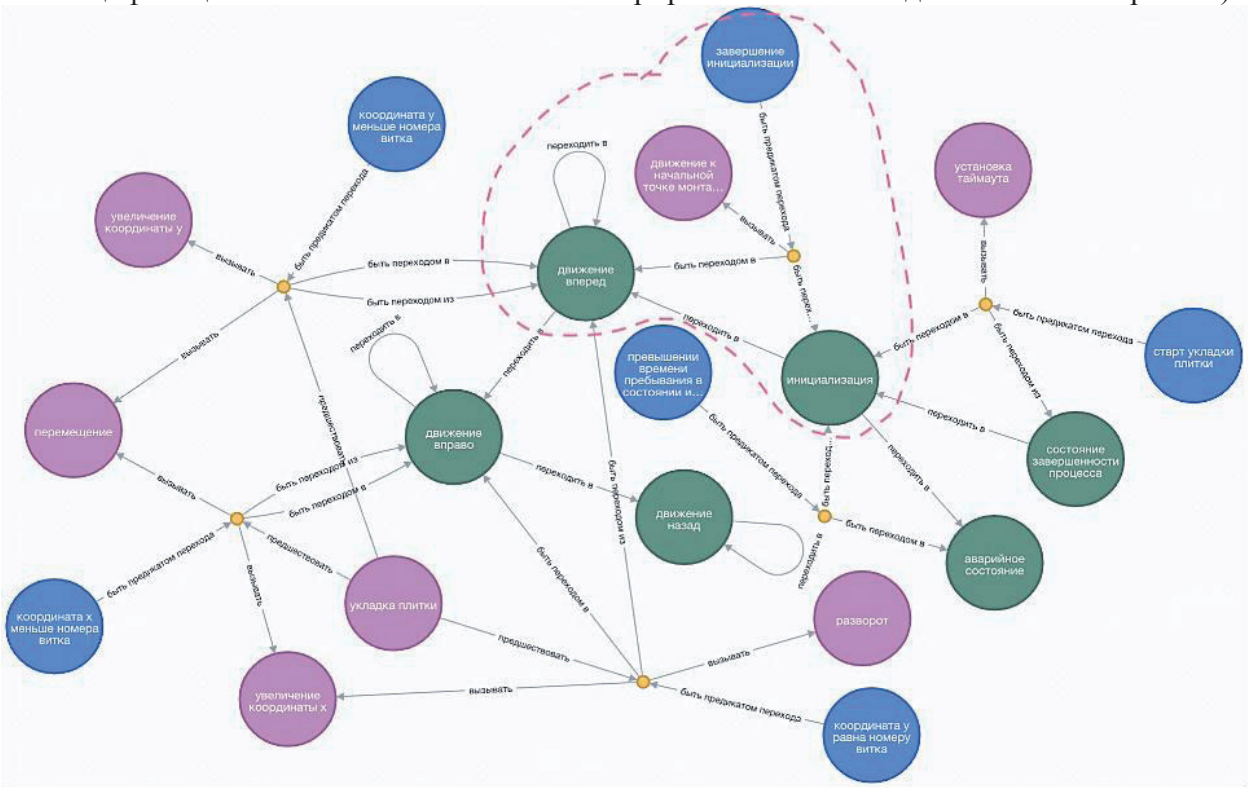


Рисунок 3 – Фрагмент онтологии проекта СЛУ роботами-плиткоукладчиками уровня проектных решений

ОС уровня проектных решений строятся с учётом заданной формальной структуры метаданных на основе спецификаций требований, к которым относятся в том числе дискурсы. Примером описания может служить следующий текст: *«Существует некоторый процесс инициализации, за которым могут скрываться последовательные подпроцессы загрузки контейнера с плитками и подпроцесс движения к начальной точке монтажа. Начало этого процесса задаётся некоторым внешним воздействием – старт укладки плитки»*. На рисунке 3 отмечены концепты и отношения, полученные на основе QA-анализа данного текста. Фрагмент QA-анализа данного дискурса:

Qn. Можно ли рассматривать процесс инициализации как состояние?
An. Процесс инициализации рассматривается как одно из состояний разрабатываемой СЛУ роботами-плиткоукладчиками. …

Qn.m. Что происходит с роботом после завершения инициализации?
An.m. После завершения инициализации робот начинает движение вперёд. ...

На рисунке 5 представлен фрагмент кода программы автоматизации на языке С, сгенерированного на основе ОС уровня реализации.

ОС уровня реализации формируются автоматизированно, за счёт реализации функций интерпретации, ставящих понятия уровня проектных решений в соответствие понятиям уровня реализации.

Интерфейс формирования онтологии реализации управляется ОМ – при добавлении каждого нового понятия в онтологию реализации он автоматически связывается с соответствующим понятием онтологии проектирования отношением «наследоваться от».

Имена понятий уровня реализации генерируются средствами автоматического перевода в соответствии с выбранным типом (*CamelCase*, *lowerCamelCase* и *snake_case*), которые могут быть изменены.

Семантические связи между понятиями уровня реализации наследуются из ОС уровня проектных решений, часть из которых (например, связи «быть предикатом перехода», «переходить в», «вызывать», «предшествовать») служит логической основой для генерации кода.

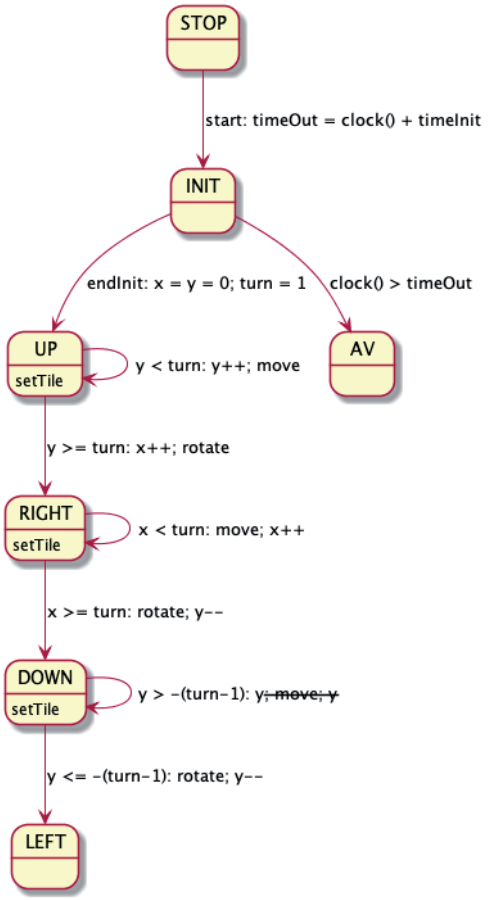
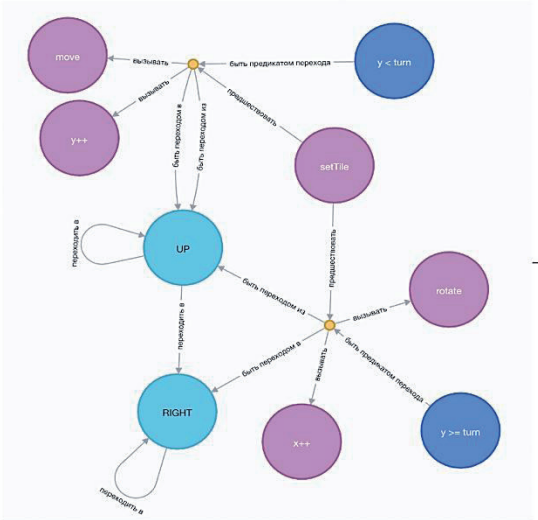


Рисунок 4 – Фрагмент сгенерированной на основе ОС диаграммы состояний




```
void step(void){
 switch (state){
  case UP:
  setTile(); /* Укладывание плитки */
  if (y < turn) { /* Справа от плиткоукладчика есть уложенные плитки */
   state = UP;
   y++; /* Увеличение значения координаты y на одну единицу */
   move(); /* Перемещение к следующему плиткоместу */
  }
  if (y >= turn) { /* Справа от плиткоукладчика нет уложенных плиток */
   state = RIGHT;
   x++; /* Увеличение значения координаты x на одну единицу */
   rotate(); /* Поворот */
  }
  break;
 case STOP:
  if (!start) { /* Нет сигнала на старт укладки плитки */
   state = STOP;
  }
  if (start) { /* Получен сигнал на старт укладки плитки */
   state = INIT;
   init(); /* Выполнение инициализации */
   timeOut = clock() + timeInit; /* Установка ограничения на пребывание в
состоянии инициализации */
  }
  break;
```

Рисунок 5 – Фрагмент сгенерированного на основе ОС исходного кода на языке С

3.3 Сокращение трудозатрат на перепроектирование за счёт онтологического моделирования

Формирование ОС на каждом этапе проектирования приводит к дополнительным трудозатратам, но при этом удаётся уменьшить трудозатраты, порождаемые семантическими разрывами. Оценить эффект от сокращения объёма исправлений проектных решений, связанных с этими разрывами, в общем виде вряд ли возможно.

Можно утверждать, что сквозное проектирование АС на основе онтологического моделирования сокращает трудозатраты на перепроектирование, вызванное изменением требований. В ОМ проекта показан путь от агрегата с изменёнными требованиями до тех агрегатов, которые специфицируют проектные решения всех этапов проектирования, вплоть до реализации. На основе экспериментов с разработкой СЛУ можно оценить различные виды трудозатрат, в том числе связанные с внесением изменений. При этом сокращение трудозатрат выражается функцией от количества изменённых требований:

$$f(x) = 1 - \frac{E_O^P + E_O^R x}{E^P + E^R x}, \quad (1)$$

где E^P – трудозатраты на разработку СЛУ до внесения изменений в требования без включения в проектный процесс онтологического моделирования, E_O^P – трудозатраты на разработку СЛУ до внесения изменений с применением онтологического моделирования; E^R и E_O^R – соответственно трудозатраты на перепроектирование вследствие возникновения необходимости внести изменения в требования, касающиеся логики функционирования СЛУ.

Для сравнения трудозатрат оценены проектные процедуры, являющиеся частью проектного процесса по разработке СЛУ (описание требования к СЛУ, построение UML-диаграммы, написание программного модуля, онтологическое специфицирование требования к проекту и др.). Оценки трудозатрат выражены в условных единицах, где за единицу принята простейшая проектная операция (описание одного требования к СЛУ), а остальные оценивались относительно неё экспертно с использованием чисел из ряда Фибоначчи [18]. Отдельно вычислены трудозатраты на реализацию проектного процесса до внесения изменений в требования, а также трудозатраты на реализацию изменений требований – оба подпроцесса рассматривались как с использованием средств онтологического моделирования, так и без него. Каждому из указанных подпроцессов соответствовал собственный набор проектных процедур. По результатам оценки построены зависимости, показанные на рисунке 6. Точками показано сокращение трудозатрат в процентах в соответствии с формулой (1).

Из рисунка 6 видно, что на начальных этапах проектного процесса требуются дополнительные трудозатраты на онтологическое моделирование, которые впоследствии компенсируются сокращением их на перепроектирование. При внесении значительного количество изменений (50 и более) сокращение трудозатрат достигает 30%.

Приведённые результаты подтверждают, что применение онтологического моделирования может сократить трудозатраты на перепроектирование при возникновении необходимости внесения изменений в требования. При этом, чем больше требований необходимо изменить, тем заметнее позитивные эффекты.

Заключение

Наличие последовательных связей между ОС разных уровней позволяет осуществить *автоматизацию управления изменениями*: изменения, внесённые в онтологию требований, автоматически вносятся в онтологии проектных решений и реализации за счёт реализации функций интерпретации. Наличие программ автоматизации, обслуживающих генерирование артефактов проектирования на основе ОС, значительно *сокращает трудозатраты* специа-

листов на перепроектирование в ситуациях, когда после разработки части проектных решений и реализации исходного кода АС возникает потребность внести изменения в логику её функционирования.

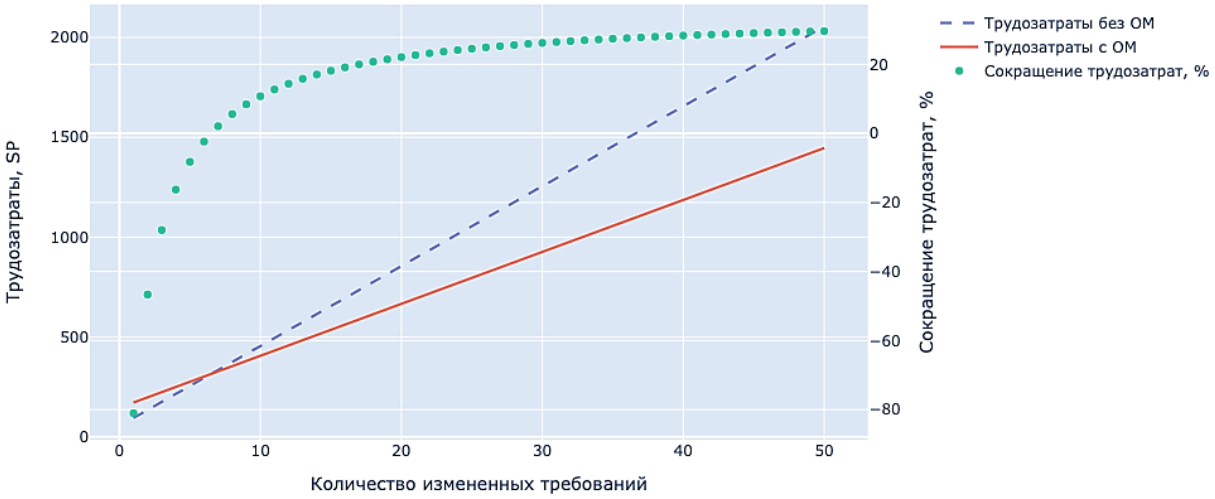


Рисунок 6 – Трудозатраты на проектирование СЛУ в условиях изменения требований

В результате достигается: *сокращение семантических разрывов*; обеспечение *концептуальной целостности*, *непротиворечивости* и *полноты* проектных решений; обеспечивается согласованность всех спецификаций и минимизируются ошибки, вызванные человеческим фактором.

Благодарности

Авторы считают своим долгом отметить большой вклад д.т.н., профессора Петра Ивановича Соснина, ушедшего из жизни в 2020 году, инициатора работ и активного участника развития изложенных в статье идей.

Работа выполнена при частичной поддержке РФФИ (гранты №18-47-732012 и 18-47-730016).

Список источников

[1] ГОСТ 34.601-90. Информационная технология. Комплекс стандартов на Автоматизированные системы. Автоматизированные системы. Стадии создания. М.: Издательство стандартов. – 1991.

[2] ГОСТ 34.602-89. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы. М.: Издательство стандартов. – 1990.

[3] ***Hein, Andreas M.*** Identification and Bridging of Semantic Gaps in the Context of Multi-Domain Engineering / Andreas M. Hein // Proceedings of 2010 Forum on Philosophy, Engineering & Technology. 2010. P.57-58.

[4] ***Minagawaa, M.*** Study on BIM Utilization for Design Improvement of Infrastructure Project / M. Minagawaa, Sh. Kusayanagi // Proceedings of the 5th International Conference of Euro Asia Civil Engineering Forum (EACEF-5). Procedia Engineering. 2015. P.431-437.

[5] ***Sikos, Leslie F.*** The Semantic Gap / Leslie F. Sikos // Description Logics in Multimedia Reasoning. Springer, 2017. P.51-66.

[6] ***Bjarnason, E.*** Requirements are slipping through the gaps — A case study on causes & effects of communication gaps in large-scale software development / E. Bjarnason, K. Wnuk, B. Regnell // Proceedings of the 2011 IEEE 19th International Requirements Engineering Conference. 2011. P.37-46.

[7] ***Bennett, R.*** Designed for change: End-to-End arguments, Internet Innovation, and the Net Neutrality debate / R. Bennett // Information Technology and Innovation Foundation. P.7,11. Retrieved 11 September 2017.

[8] ***Негода В.Н.*** Онтологическое моделирование в проектировании средств логического управления / В.Н. Негода, А.А. Куликова // Труды Международного научно-технического конгресса «Интеллектуальные системы и информационные технологии – 2021» («ИС & ИТ-2021», «IS&IT'21»). Научное издание. – Таганрог: Изд-во Ступина С.А., 2021. С.93-98.

[9] ***Шалыто, А.А.*** Логическое управление: Методы аппаратной и программной реализации алгоритмов / А.А. Шалыто. – СПб.: Наука. 2000. 780 с.

[10] ***Dorst, K.*** The Nature of Design Thinking / K. Dorst // DTRS8 Interpreting Design Thinking: Design Thinking Research Symposium Proceedings. 2009. P.131–139.

[11] ***Razavian, M.*** Reflective Approach for Software Design Decision Making / M. Razavian, A. Tang, R. Capilla, P. Lago // Proc. of the first symposium "Qualitative Reasoning about Software Architectures". 2016. P.19-26.

[12] ***Häger, F,*** DT@Scrum: Integrating Design Thinking with Software Development Processes / F. Häger, T. Kowark, J. Krüger, Ch. Vetterli, F. Übernickel, M. Uflacker // Design Thinking. Understanding Innovation. 2014. P.263-289.

[13] ***Sosnin, P.*** Ontological Support of Design Thinking in Developments of Software Intensive Systems / P. Sosnin, A. Pushkareva, V. Negoda // Abraham A., Kovalev S., Tarassov V., Snasel V., Vasileva M., Sukhanov A. (eds) Proceedings of the Second International Scientific Conference "Intelligent Information Technologies for Industry" (IITI'17). IITI 2017. Advances in Intelligent Systems and Computing, v.679. Springer, Cham. P.159-168.

[14] ***Соснин, П.И.*** Управление знаниями и опытом в проектной организации / П.И. Соснин, В.А. Маклаев, А.А. Перцев. – Ульяновск: УлГТУ, 2018. 213 с.

[15] ***Соснин, П.И.*** Вопросно-ответное программирование человеко-компьютерной деятельности / П.И. Соснин. – Ульяновск: УлГТУ, 2010. 240 с.

[16] ***Брукс, Ф.*** Проектирование процесса проектирования: записки компьютерного эксперта. Пер. с англ. — М.: ООО "И.Д. Вильямс", 2013. 464 с.

[17] ***Sosnin, P.*** An Architectural Approach to the Precedent-Oriented Solution of Tasks in Designing a Software Intensive System / P. Sosnin, S. Shumilov, A. Ivasev // Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2019, 11623 LNCS. P.97–107.

[18] ***Городецкий, В.И.*** Искусственный интеллект – наука и информационная технология: настоящее и будущее / В.И. Городецкий // Материалы конференции «Информационные технологии в управлении». 2020. С.3-14.

[19] ***Grenning, J.*** Planning Poker or How to avoid analysis paralysis while release planning / J. Grenning. https://wingman-sw.com/articles/planning-poker.

Сведения об авторах

Негода Виктор Николаевич, 1949 г. рождения. Окончил радиотехнический факультет Ульяновского политехнического института, профессор кафедры «Вычислительная техника» Ульяновского государственного технического университета. Имеет статьи, монографии и авторские свидетельства в области проектирования встроенных систем контроля и управления. Область научных интересов – автоматизация проектирования логического управления техническими системами. Author ID (RSCI): 393758; Author ID (Scopus): 57196052478. *nvn@ulstu.ru.*

Куликова Анна Александровна, 1992 г. рождения. Окончила Ульяновский государственный технический университет в 2014 г. и аспирантуру по специальности «Системы автоматизации проектирования (промышленность)» в УлГТУ в 2020 г. Ассистент кафедры «Вычислительная техника» Ульяновского государственного технического университета. Область научных интересов – проектирование систем с программным обеспечением, использование онтологий в проектировании. Author ID (RSCI): 974649; Author ID (Scopus): 57206900994; Researcher ID (WoS): T-4662-2017. *a.push1206@gmail.com.*

Поступила в редакцию 30.10.2021, после рецензирования 10.11.2021. Принята к публикации 10.12.2021.

End-to-end design of automated systems based on ontologies

V.N. Negoda, A.A. Kulikova

Ulyanovsk State Technical University, Ulyanovsk, Russia

Abstract

Currently, a large number of studies are being carried out on the use of ontologies in the development of automated systems. Ontological modeling improves the efficiency of the software development process. The division of labor in the development of automated systems contributes to the appearance of semantic gaps between the phases of the design process. The practice of using a wide range of reusable design artifacts is a risk factor for violation of conceptual integrity, consistency and completeness of the developed design solutions. The article proposes a new approach to ontological modeling of automated systems, which serves the design process for their development at all stages of design up to implementation; and a new structure of metadata of ontological specifications of automated systems, which allows taking into account semantically important entities and features of the system being developed. It is shown that the use of this approach significantly reduces labor costs in the design of a complex automated system in the face of changing requirements at different stages of system creation. The use of ontological models in the design process helps to increase the conceptual integrity, consistency and completeness of the developed design solutions.

Keywords: *design automation, automated systems, end-to-end design, question-answer modeling, project ontology.*

Citation: *Negoda VN, Kulikova AA.* End-to-end design of automated systems based on ontologies [In Russian]. *Ontology of designing*. 2021; 11(4): 450-463. DOI: 10.18287/2223-9537-2021-11-4-450-463.

Acknowledgment: The authors consider it their duty to note the great contribution of Doctor of Technical Sciences, Professor Petr Sosnin, who passed away in 2020, the initiator of the work and an active participant in the development of the ideas outlined in the article. This work was supported by the Russian Foundation for Basic Research (projects 18-47-732012 и 18-47-730016).

List of figures

Figure 1 – IDEF0 diagram of the design process for the development of automated systems
Figure 2 – Trajectory diagram of a tile-laying robot
Figure 3 – Project ontology fragment of the tile-laying robot logical control system at the level of design solutions
Figure 4 – State diagram fragment generated based on ontological specifications
Figure 5 – C code fragment generated based on ontological specifications
Figure 6 – Labor costs on control system design in the context of changing requirements

References

[1] State Standard 34.601-90. Information technology. Set of standards for automated systems. Automated systems. Stages of development [In Russian]. Moscow, Standartinform Publ., 1991.
[2] State Standard 34.602-89. Information technology. Set of standards for automated systems. Technical directions for automated system making [In Russian]. Moscow, Standartinform Publ., 1990.
[3] ***Hein Andreas M.*** Identification and Bridging of Semantic Gaps in the Context of Multi-Domain Engineering. Proceedings of 2010 Forum on Philosophy, Engineering & Technology. 2010. P.57-58.
[4] ***Minagawaa M, Kusayanagi Sh.*** Study on BIM Utilization for Design Improvement of Infrastructure Project. Proceedings of the 5th International Conference of Euro Asia Civil Engineering Forum (EACEF-5). Procedia Engineering. 2015. P. 431-437.
[5] ***Sikos Leslie F.*** The Semantic Gap. Description Logics in Multimedia Reasoning. Springer, 2017. P.51-66.
[6] ***Bjarnason E, Wnuk K, Regnell B.*** Requirements are slipping through the gaps — A case study on causes & effects of communication gaps in large-scale software development. Proceedings of the 2011 IEEE 19th International Requirements Engineering Conference. 2011. P.37-46.
[7] ***Bennett R.*** Designed for change: End-to-End arguments, Internet Innovation, and the Net Neutrality debate. *Information Technology and Innovation Foundation*. P.7,11. Retrieved 11 September 2017.

[8] ***Negoda VN, Kulikova AA.*** Ontological Modeling in Logical Control Systems Design [In Russian]. Proc. of the International Congress on Intelligent Systems and Information Technologies IS&IT'21. Taganrog, 2021. P. 93-98.
[9] ***Shalygo AA.*** Logical Controlling: Methods for Hardware and Software Implementation of Algorithms [In Russian]. Saint Petersburg: Scrience. 2000. 780 p.
[10] ***Dorst K.*** The Nature of Design Thinking. DTRS8 Interpreting Design Thinking: Design Thinking Research Symposium Proceedings. 2009. P.131–139.
[11] ***Razavian M, Tang A, Capilla R, Lago P.*** Reflective Approach for Software Design Decision Making. Proc. of the first symposium "Qualitative Reasoning about Software Architectures". 2016. P.19-26.
[12] ***Häger F, Kowark T, Krüger J, Vetterli Ch, Übernickel F, Uflacker M.*** DT@Scrum: Integrating Design Thinking with Software Development Processes. Design Thinking. Understanding Innovation. 2014. P.263-289.
[13] ***Sosnin P, Pushkareva A, Negoda V.*** Ontological Support of Design Thinking in Developments of Software Intensive Systems. Abraham A., Kovalev S., Tarassov V., Snasel V., Vasileva M., Sukhanov A. (eds). Proceedings of the Second International Scientific Conference "Intelligent Information Technologies for Industry" (IITI'17). IITI 2017. Advances in Intelligent Systems and Computing, v.679. Springer, Cham. P.159-168.
[14] ***Sosnin PI, Maklaev VA, Pertsev AA.*** Knowledge and Experience Management in Project Organization [In Russian]. Ulyanovsk: UlSTU, 2018. 213 p.
[15] ***Sosnin PI.*** Question-and-answer Programming of Human-Computer Activity [In Russian]. Ulyanovsk: UlSTU, 2010. 240 p.
[16] ***Brooks F.*** The Design of Design: Essays from a Computer Scientist [In Russian]. Moscow, 2013. 464 p.
[17] ***Sosnin P, Shumilov S, Ivasev A.*** An Architectural Approach to the Precedent-Oriented Solution of Tasks in Designing a Software Intensive System. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2019, 11623 LNCS. P.97–107.
[18] ***Gorodetsky VI.*** Artificial Intelligence - Science and Information Technology: Present and Future [In Russian]. Materials of the conference "Information technologies in management". 2020. P.3-14.
[19] ***Grenning J.*** Planning Poker or How to avoid analysis paralysis while release planning. https://wingman-sw.com/articles/planning-poker.

About the authors

Viktor Nikolaevich Negoda (b. 1949) PhD, graduated from the Radio Engineering department of the Ulyanovsk Technical Institute, is a professor at the Computer Science Department in Ulyanovsk State Technical University. He has publications in the area of computer-aided design of embedded systems. Computer-aided design of technical systems with logical control is his area of expertise. Author ID (RSCI): 393758, Author ID (Scopus): 57196052478. *nvn@ulstu.ru.*

Anna Aleksandrovna Kulikova (b. 1992), graduated from the Ulyanovsk State Technical University in 2014. She is an assistant at the Computer Science Department in Ulyanovsk State Technical University. Automated design of software systems, using ontologies in design is the area of expertise. Author ID (RSCI): 974649; Author ID (Scopus): 57206900994; Researcher ID (WoS): T-4662-2017. *a.push1206@gmail.com.*

Received October 30, 2021. Revised November 10, 2021. Accepted December 10, 2021.