На правах рукописи

ЛОМОВ
Александр Андреевич

# МЕТОД ОНТОЛОГО-ОРИЕНТИРОВАННОЙ РАЗРАБОТКИ ПРОГРАММНЫХ АГЕНТОВ С КОСВЕННЫМ ВЗАИМОДЕЙСТВИЕМ В ИНТЕЛЛЕКТУАЛЬНЫХ ПРОСТРАНСТВАХ

05.13.11 — Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

АВТОРЕФЕРАТ
диссертации на соискание ученой степени
кандидата технических наук

Петрозаводск
2014

Работа выполнена в Петрозаводском государственном университете.

Научный руководитель: **Корзун Дмитрий Жоржевич**, кандидат физико-математических наук, доцент кафедры информатики и математического обеспечения ФГБОУ ВПО «Петрозаводский государственный университет» (ПетрГУ)

Официальные оппоненты: звание, уч. степень ФИО

**Пантелеев Михаил Георгиевич**
кандидат технических наук, доцент кафедры вычислительной техники ФГБОУ ВПО «Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» (СПбГЭТУ)

Ведущая организация: **ФГБУН Институт прикладных математических исследований Карельского научного центра Российской академии наук (ИПМИ КарНЦ РАН, г. Петрозаводск).**

Защита состоится ... (число, месяц, год) в … часов на заседании диссертационного совета Д 002.199.01 при Федеральном государственном бюджетном учреждении науки Санкт-Петербургском институте информатики и автоматизации Российской академии наук по адресу: 199178, Санкт-Петербург, В.О., 14 линия, 39.

С диссертацией можно ознакомиться в библиотеке Федерального государственного бюджетного учреждения науки Санкт-Петербургского института информатики и автоматизации Российской академии наук

Автореферат разослан ... (число, месяц, год)

Ученый секретарь
диссертационного совета Д 002.199.01
кандидат технических наук, доцент Нестерук Филипп Геннадьевич

## Общая характеристика работы

**Актуальность темы.** Парадигма интеллектуальных пространств ориентирована на разработку прикладных сервисно-ориентированных многоагентных систем для вычислительных сред «повсеместных вычислений» и Интернета физических устройств (Internet of Things, IoT). Прикладная система строится как набор взаимодействующих программных агентов на окружающих и удаленных устройствах (напр., бытовая техника, мобильные ЭВМ). Интеллектуальное пространство (ИП) обеспечивает динамическое множество участников контекстно-зависимой информацией, сервисами и персонифицированными рекомендациям. Участниками выступают персональные агенты, агенты оборудования, сервисные агенты и др. Интеллектуальные пространства классифицируют по типу вычислительной среды, модели взаимодействия агентов и способу организации разделяемого информационного содержимого.

Широкое применение получили интеллектуальные пространства, разворачиваемые в локализованных физических окружениях (напр., в доме или офисе — системы Z-Wave, KNX, C-Bus). В случае открытых ИП участники представлены разнообразными вычислительными устройствами, а состав участников динамически меняется (напр., ИП в общественных местах — конференц-зал, зона отдыха и т.п.). Реализация таких ИП сталкивается с проблемой интероперабельности (возможности взаимодействия) множества динамических участников, включая вопросы как сетевого взаимодействия, так и разделения информационного содержимого.

В отдельный класс выделяют интеллектуальные пространства с косвенным взаимодействием программных агентов и онтолого-ориентированным представлением и обработкой разделяемого информационного содержимого. Такие ИП характеризуются следующими свойствами. Вычислительная среда удовлетворяет условиям IoT — разнообразные устройства локализованы в физическом окружении и выполняют сетевые коммуникации как друг с другом, так и с внешними системами. Взаимодействие агентов основано на моделях «классная доска» и «публикация/подписка». Информационное содержимое ИП представлено в общем хранилище на основе моделей и технологий Семантического веб, ориентированных на машинную обработку.

Разработка агентов для ИП включает задачу программирования логики агента (программный код, реализующий действия агента как участника системы). В логике агента программируется косвенное взаимодействие с другими агентами, обработка информационного содержимого на основе модели RDF (Resource Description Framework) и онтологий проблемной области на языке OWL (Web Ontology Language).

Для программирования логики агента в терминах OWL-онтологий (классы, свойства, индивиды) необходимо промежуточное ПО и инструментарий разработки,

которые предоставят уровень абстракции и механизмы программирования, скрывающие от разработчика нижележащую реализацию косвенного взаимодействия агентов. Моделирование проблемной области с помощью онтологий позволяет применять методы модельно-ориентированной разработки. Онтологическая модель проблемной области определяет информационные объекты (далее — объекты), посредством которых агент взаимодействует с другими агентами в ИП. Промежуточное ПО позволяет связать такие объекты с конструкциями языка программирования и представить объекты в программном коде логики агента структурами данных. Развитие таких методов в итоге приводит к онтолого-ориентированной разработке ПО, когда онтологии применяются на основных этапах разработки (анализ, проектирование, реализация).

**Целью диссертационной работы** является повышение эффективности разработки агентов с косвенным взаимодействием в интеллектуальных пространствах за счет автоматизированной онтолого-ориентированной разработки программного кода логики агента на основе онтологий проблемной области.

**Объектом** исследования являются программные агенты интеллектуального пространства, применяющие 1) модели «классная доска» и «публикация/подписка» для косвенного взаимодействия с другими агентами и 2) RDF-модель и OWL-онтологии для представления и совместной обработки информационного содержимого.

**Предметом** исследования являются модели косвенного взаимодействия агентов и механизмы программирования, обеспечивающие 1) автоматизированную онтолого-ориентированную разработку программного кода логики агента для разнообразных аппаратно-программных вычислительных платформ, 2) интеграцию, синхронизацию и обработку на стороне агента во время его выполнения информационного содержимого с использованием объектов онтологической модели проблемной области.

Для достижения поставленной цели в работе решаются следующие задачи.

1. Анализ существующих методов разработки программных агентов и моделей их взаимодействия для определения путей упрощения программирования логики агента за счет применения онтологической модели проблемной области.

2. Разработка метода автоматизированной онтолого-ориентированной разработки агентов для упрощения программирования логики агента с использованием объектов онтологической модели проблемной области.

3. Разработка моделей взаимодействия агентов для автоматической интеграции на стороне агента объектов из нескольких ИП, автоматической синхронизации на стороне агента объектов при их изменении в ИП и обработки на стороне агента группы объектов с автоматическим построением агентом запросов к ИП виде транзакций.

4. Разработка механизмов программирования для использования предлагаемых моделей взаимодействия при программировании логики агентов.

5. Реализация и апробация программных средств для применения предлагаемого метода онтолого-ориентированной разработки агентов и механизмов программирования с учетом разнообразия аппаратно-программных вычислительных платформ.

Результаты выполненных в работе исследований и проектных работ основаны на автоматизированных системах онтологического моделирования, моделях взаимодействия и протоколах распределенных систем, моделях и технологиях Семантического веб, методах системного программирования, объектно-ориентированном подходе к проектированию и программированию.

**Основные положения, выносимые на защиту:**

1. Метод автоматизированной онтолого-ориентированной разработки для программирования логики агента с использованием объектов онтологической модели проблемной области и моделей косвенного взаимодействия агентов.

2. Модель взаимодействия агентов на основе сессии параллельных сеансов сетевого доступа для участия агента в нескольких ИП и автоматической интеграции на стороне агента объектов из этих ИП.

3. Модель взаимодействия агентов на основе операции подписки для отслеживания агентом происходящих в ИП изменений и автоматической синхронизации на стороне агента изменяемых в ИП объектов.

4. Модель взаимодействия агентов на основе обработки группы объектов на стороне агента с автоматическим формированием запроса на множественное изменение информационного содержимого ИП в виде транзакции.

5. Механизмы обеспечения онтолого-ориентированной разработки программных агентов на основе предложенных моделей взаимодействия агентов и реализация этих механизмов в инструментарии SmartSlog платформы Smart-M3.

**Научная новизна** работы состоит в следующем.

1. Разработан метод разработки программных агентов на основе онтологических библиотек как промежуточного ПО, отличающийся поддержкой выбора объектов онтологической модели проблемной области для косвенного взаимодействия с другими агентами и позволяющий генерировать программный код онтологической библиотеки для выбранных объектов на целевом языке программирования.

2. Разработана модель взаимодействия агентов на основе сессии, отличающаяся поддержкой параллельных сеансов сетевого доступа при программировании логики агента и позволяющая одновременно взаимодействовать в нескольких ИП. Агент соз-

дает сессию из сеансов сетевого доступа с локальным хранилищем информации для автоматической интеграции объектов, разделяемых агентами в нескольких ИП.

3. Разработана модель взаимодействия агентов на основе операции подписки, отличающаяся поддержкой использования в логике агента объектов онтологической модели при программировании для отслеживания изменений в ИП и позволяющая на стороне агента автоматически синхронизировать объекты с изменениями, вносимыми в ИП другими агентами.

4. Разработана модель взаимодействия агентов на основе обработки группы объектов, отличающаяся возможностью программировать правила обработки в логики агента и позволяющая на стороне агента автоматически формировать запрос на множественное изменение информационного содержимого ИП в виде одной транзакции, уменьшая число сетевых обращений к ИП.

5. Реализованы механизмы программирования в виде инструментария Smart-Slog для апробации предложенного метода, отличающиеся поддержкой моделей косвенного взаимодействия агентов и позволяющие разрабатывать программных агентов для ИП на платформе Smart-M3.

**Обоснованность и достоверность** научных положений обеспечены аналитическим обзором исследований и разработок в области интеллектуальных пространств, подтверждаются положительными итогами практического использования результатов диссертации в пилотных прикладных системах на базе платформы Smart-M3, а также апробацией основных научно-практических положений в печатных трудах и докладах на российских и международных конференциях.

**Практическая ценность работы.** Предложенные модели взаимодействия агентов используются в полученном методе разработки агентов. Механизмы программирования реализованы в инструментарии с открытым исходным кодом SmartSlog платформы Smart-M3. Инструментарий позволяет а) повысить качество разработки программного кода за счет связывания программных конструкций с онтологической моделью проблемной области, б) снизить трудозатраты на разработку и сопровождение программных агентов и в) разрабатывать агентов для различных аппаратно-программных вычислительных платформ. Инструментарий применяется при разработке системы интеллектуального зала SmartRoom, автоматизирующей проведение мероприятий коллективной деятельности (конференции, собрания и др.).

**Реализация результатов работы.** Исследования проводились в рамках реализации комплекса мероприятий Программы стратегического развития ПетрГУ на 2012-2016 г.г., были поддержаны грантом КА179 «Комплексное развитие регионального сотрудничества в области открытых инноваций в ИКТ» в рамках Karelia ENPI — совмест-

ной программы Европейского союза, Российской Федерации и Республики Финляндия, получили финансовую поддержку со стороны Минобрнауки России в рамках базовой части государственного задания №2014/154 в сфере научной деятельности, НИР № 1481.

**Апробация.** Основные положения и результаты диссертационной работы представлялись на международных конференциях Open Innovations Association FRUCT: 2011 г. (FRUCT 9, FRUCT 10), 2012 г. (FRUCT 12) и 2013 г. (FRUCT 13), семинаре «Проблемы современных информационно-вычислительных систем» в МГУ им М. В. Ломоносова, 2010 г.; международной конференции Annual International Workshop on Advances in Methods of Information and Communication Technology (AMICT), ПетрГУ, 2010 г.; международной конференции Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM), 2010 г.; конгрессе Information Systems and Technologies (IS&IT'11), 2011 г.; научном семинаре СПИИРАН, 2013 г.; научном семинаре ИИММ КНЦ РАН, 2013 г.

**Публикации.** Основные результаты по материалам диссертационной работы опубликованы в 10 печатных работах, среди них 3 работы из перечня ВАК, 1 работа индексируется в базе данных Scopus.

**Объем и структура работы.** Диссертация объемом 122 машинописных страниц состоит из введения, 4 глав, заключения и приложения. Список использованной литературы содержит 88 наименований. Текст диссертации содержит 108 страниц машинописного текста, включая 30 рисунков, 5 таблиц.

## Содержание работы

**Во введении** обосновывается актуальность работы и формулируется ее цель, приводятся основные положения, выносимые на защиту, отмечается их научная новизна и практическая значимость, кратко изложены основные результаты и приведено содержание работы по главам.

**В первой главе** рассмотрены интеллектуальные пространства как парадигма для построения многоагентных систем, обеспечивающих пользователей контекстно-зависимой информацией, сервисами и персонифицированными рекомендациями. В качестве иллюстративного примера вводится прикладная система «умный дом».

Интеллектуальные пространства классифицируют в зависимости от типа вычислительной среды, модели сетевого взаимодействия агентов и способа организации разделяемого агентами информационного содержимого. Выделяют интеллектуальные пространства с косвенным взаимодействием, в которых агенты динамически формируют информационное содержимое ИП, используя онтолого-ориентированные модели и технологии Семантического веб для представления и обработки информации.

Проблемная область описывается набором онтологий, позволяющих интерпретировать информационное содержимое ИП в терминах объектов и их семантических связей. Агент следует частной онтологической модели, определяющей, какое подмножество объектов требуется этому агенту для взаимодействия с другими агентами. Сетевой доступ любого агента к информационному содержимому ИП выполняется через информационного брокера, обрабатывающего запросы на чтение и запись.

Разработчик сталкивается со следующими проблемами при программировании логики агента. 1. Программирование взаимодействия с другими агентами в ИП. В логике агента необходимо реализовать сетевой доступ к ИП (в общем случае, в виде набора параллельных вычислительных потоков к одному и более ИП). Программный код оперирует со структурами данных для представления на стороне агента объектов из информационного содержимого ИП. 2. Программирование реакции агента на изменения информационного содержимого ИП, вносимых другими агентами. В логике агента необходимо реализовать подписку на объекты в информационном содержимом ИП для отслеживания изменений. В программном коде задается реакция агента на поступающие изменения с внесением ответных изменений. На решение этих проблем влияют, во-первых, ограничения целевого устройства для программного агента: язык программирования, вычислительные ресурсы, средства сетевых коммуникаций, организация параллельных вычислительных потоков. Во-вторых, трудоемкость разработки зависит от уровня абстракции для представления в программном коде обрабатываемого информационного содержимого.

Два уровня разработки программного агента для ИП показаны на рис. 1. При низкоуровневой разработке (уровень RDF-представления) в программном коде используется представление в виде RDF-троек, т.е. без непосредственных манипуляций в коде с объектами онтологической модели. Сетевой доступ к ИП — через RDF-ориентированный ИП-интерфейс. Высокоуровневая разработка (уровень OWL-представления) скрывает от разработчика операции с RDF-тройками в промежуточном ПО. Разработчику предоставлен дополнительный уровень абстракции (структуры данных для представления в программном коде объектов онтологической модели — OWL классы, свойства и индивиды) и поддерживающий его интерфейс прикладного программирования. Промежуточное ПО увеличивает общий объем программного кода агента, но упрощает программирование логики агента за счет приближения к проблемной области. В результате, уменьшается программный код логики агента, который необходимо создать и протестировать прикладному разработчику.

Задачи программирования логики агента для реализации взаимодействия агентов приведены в табл. 1. Их решение на уровне RDF-представления требует от разра-

ботчика самостоятельно программировать сетевой доступ к ИП для обмена наборами троек и их обработку, в том числе доступ или синхронизацию обновлений в виде набора параллельных вычислительных потоков. На уровне OWL-представления решение допускает автоматизацию: промежуточное ПО предоставляет механизмы для программирования в логике агента взаимодействия с другими агентами через разделение в информационном содержимом ИП объектов онтологической модели проблемной области. Для разработки агентов выгодно использовать онтолого-ориентированный подход, повышающий уровень автоматизации разработки, надежности и снижающий трудозатраты на сопровождение агентов. На основе онтологий автоматизируется проверка корректности и проводится интеграция информации.

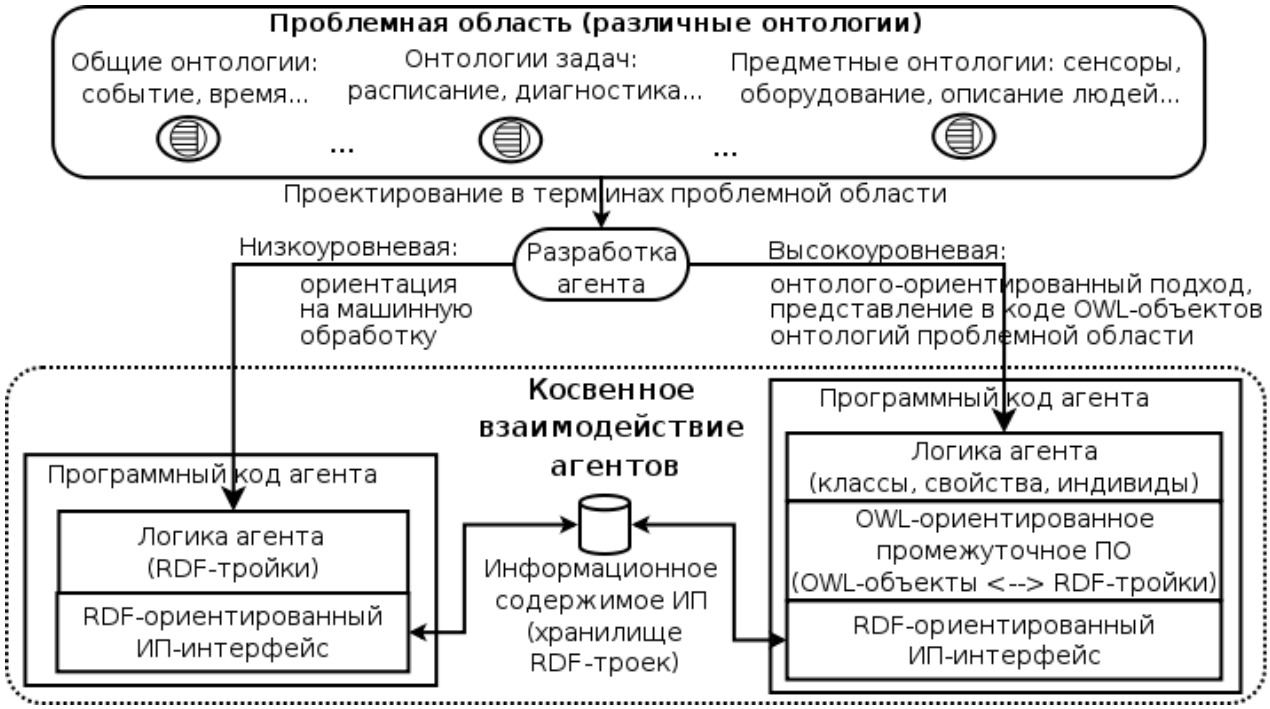


**Рисунок 1. Низкоуровневая и высокоуровневая разработка программных агентов для интеллектуальных пространств.**

**Во второй главе** предложен метод автоматизированной онтолого-ориентированной разработки агентов для ИП. Промежуточное ПО для высокоуровневой разработки агента реализуется как программная онтологическая библиотека функций и структур данных, предоставляющая разработчику логики агента интерфейс прикладного программирования, параметризируемый объектами онтологической модели. Разработка агента включает следующие шаги (рис. 2), каждый из которых поддерживает автоматизацию на основе онтолого-ориентированного подхода к разработке ПО: 1) онтологическое моделирование проблемной области, 2) выбор объектов для формирования частной онтологической модели проблемной области для взаимодействия агента с другими агентами и автоматическая генерация программного кода онтологи-

ческой библиотеки, 3) программирование логики агента и взаимодействия с другими агентами, используя структуры данных и функции из онтологической библиотеки.

| Задача | RDF-представление | OWL-представление | Автоматизация |
|---|---|---|---|
| Представление объектов онтологической модели проблемной области. | Программный код оперирует наборами троек. | Структуры данных, приближенные к проблемной области. | Генерация промежуточного ПО для логики агента. |
| Сетевой доступ к нескольким ИП с интеграцией поступающих обновлений. | Сетевые сеансы доступа в параллельных вычислительных потоках с раздельной обработкой получаемых из ИП троек. | Сетевые сеансы объединены общим локальным хранилищем. | Интеграция поступающих обновлений из ИП. |
| Отслеживание изменений в ИП, вносимых другими агентами, и синхронизация обновлений. | Подписка на тройки и программирование обработки получаемых наборов троек. | Подписка на онтологические объекты (классы, индивиды, свойства). | Синхронизация поступающих обновлений из ИП. |
| Локальная обработка с последующим внесением множественных изменений в ИП. | Обработка наборов троек и формирование множества запросов на внесение изменений. | Обработка в терминах проблемной области и формирование запросов на свойств индивидов. | Формирование общего запроса на внесение изменений в ИП. |

**Таблица 1. Базовые задачи программирования логики агента.**

Онтологическая библиотека состоит из частей $l_{\text{data}}$ и $l_{\text{func}}$. В $l_{\text{data}}$ определены структуры данных для представления объектов частной онтологической модели агента. С учетом целевой для агента аппаратно-программной вычислительной платформы предлагается генерировать программный код $l_{\text{data}}$ с выбором языка программирования. Инвариантная функциональная часть $l_{\text{func}}$ содержит функции для программирования взаимодействия агента в общеонтологических терминах — «класс», «свойство», «индивид». Тем самым, программная реализация $l_{\text{func}}$ зависит от целевой аппаратно-программной платформы без привязки к конкретной проблемной области.

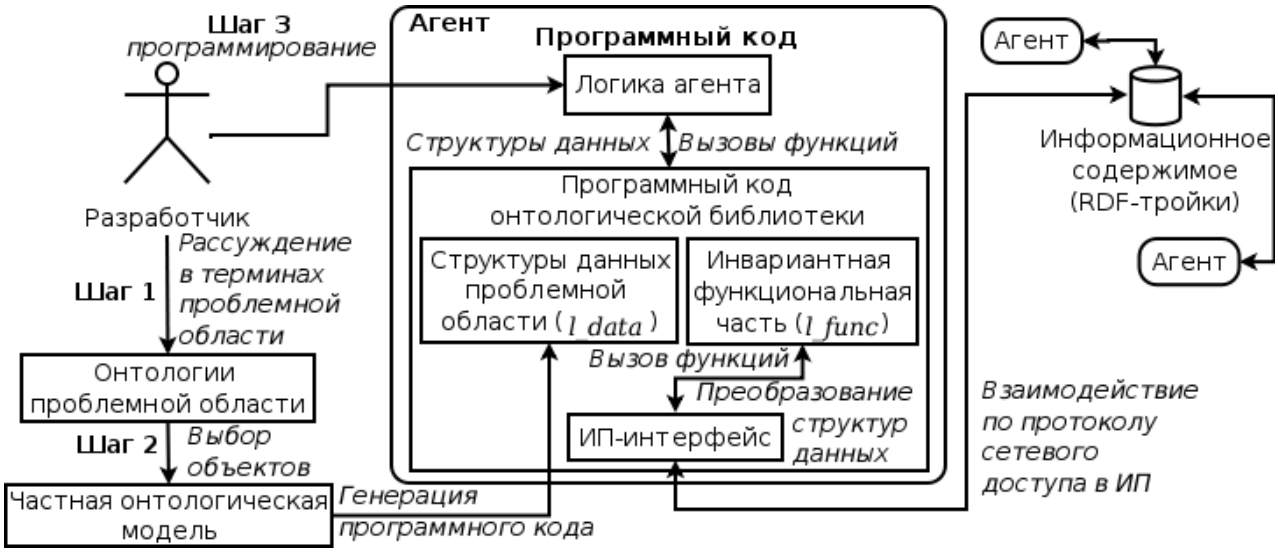


**Рисунок 2. Схема использования онтологической библиотеки в рамках метода автоматизированной разработки ПО.**

В логике агента используются вызовы функций из $l_{\text{func}}$ с параметрами на основе структур данных из $l_{\text{data}}$. Для сетевого доступа агента к ИП онтологическая библиотека выполняет преобразования между структурами данных $l_{\text{data}}$ и RDF-представлением. Часть $l_{\text{func}}$ поддерживает различные ИП-интерфейсы, чтобы обеспечить разработку агентов для широкого круга аппаратно-программных платформ.

Емкостная трудоемкость онтологической библиотеки оценивается как

$$R_{\text{mem}} = v_{\text{data}} + v_{\text{func}} + n_{\text{loc}} C_{\text{obj}} + n_{\text{ses}} C_{\text{ses}} + n_{\text{sub}} C_{\text{sub}}. \quad (1)$$

Величины $v_{\text{data}}$ и $v_{\text{func}}$ определяют объем памяти для загрузки частей $l_{\text{data}}$ и $l_{\text{func}}$ на устройство. Параметры $C_{\text{obj}}$, $C_{\text{ses}}$ и $C_{\text{sub}}$ отражают средние затраты по памяти для хранения на стороне агента одного индивида и его свойств, сеанса и подписки. Параметры $n_{\text{loc}}$, $n_{\text{ses}}$ и $n_{\text{sub}}$ — это число индивидов, сеансов и подписок.

Время выполнения отдельной функции из $l_{\text{func}}$ зависит от автоматических преобразований между структурами данных $l_{\text{data}}$ и RDF-форматом ИП-интерфейса. Временная трудоемкость онтологической библиотеки оценивается как

$$R_{\text{proc}} = \lambda n_{\text{th}} (C_{\text{obj-trp}} + C_{\text{trp-obj}}), \quad (2)$$

где $C_{\text{obj-trp}}$ и $C_{\text{trp-obj}}$ — средние затраты на преобразование (из OWL-объектов в RDF-тройки и обратно), $\lambda$ — интенсивность вызова функций онтологической библиотеки в вычислительном потоке, $n_{\text{th}}$ — число параллельных вычислительных потоков.

Оценки (1) и (2) определяют затраты, вызываемые переходом к разработке на уровне OWL-представления. Для маломощных устройств эти затраты должны быть ограничены. Применение частной онтологической модели для агента уменьшает значения $n_{\text{loc}}$, $C_{\text{obj-trp}}$ и $C_{\text{trp-obj}}$. Так, в системе «умный дом» онтологии описывают 1) людей, 2) сенсоры физического окружения и 3) бытовое климатическое оборудование. Поскольку агент сенсора оперирует с небольшим набором измеряемых параметров и операции доступа к ИП сводятся к простым операциям чтения/записи, то генерируемая для него часть $l_{\text{data}}$ не требует трудоемких программных конструкций.

Предложенный метод применяет модели взаимодействия агентов, где объекты онтологической модели выступают базовыми элементами при программировании логики агента. В работе предлагаются три таких модели.

*Модель взаимодействия агентов на основе сессии* используется агентом как $(P, Q, D)$, где $P$ — множество сеансов сетевого доступа агента к ИП, $Q$ — множество подписок для отслеживания изменений в ИП, $D$ — множество объектов онтологической модели в локальном хранилище. Сетевой сеанс требует отдельного вычис-

лительного потока на стороне агента. Подписка привязана к сетевому сеансу. Локальное хранилище разделено между сеансами и подписками, что позволяет автоматизировать интеграцию обновлений, поступающих из нескольких ИП. Так, свойства индивида могут разделяться в разных ИП, обновления поступают по параллельным сетевым сеансам, но в логике агента индивид представлен как один целостный объект.

Каждый сеанс (и подписка) находится в активном состоянии или ожидает. В активном состоянии установлено сетевое соединение с ИП (для подписки в отслеживаются изменения в ИП). В состоянии ожидания сеанс или подписка определены только на стороне агента (сетевое соединения отсутствует). Следовательно, при программировании логики агента вместо явного завершения сетевого сеанса и последующей установки нового можно использовать компактные программные конструкции переключения состояний сеансов. Для этих целей онтологическая библиотека включает операции активации ( *ACTIVE* ) и ожидания ( *WAIT* ). Для сеансов, подписок и объектов в локальном хранилище сессии введены операции: регистрация ( *REG* ) — ассоциирует объект с сессией, удаление ( *DEL* ) — удаляет ассоциацию объекта.

Пример использования модели — взаимодействие агента в системе «умный дом», где развернуты два ИП (внутри дома и во дворе). Персональный агент на мобильном телефоне использует сетевые сеансы доступа к двум ИП. В логике агента переключаются состояния сеансов и подписок в зависимости от местоположения человека. Поступающие по подпискам изменения из этих ИП интегрируются агентом в локальном хранилище информации в виде объектов (индивидов и их свойств).

*Модель взаимодействия агентов на основе онтолого-ориентированной операции подписки* используется агентом как $(O_{\text{src}}, D_{\text{sub}}, D_{\text{ind}})$, где $O_{\text{src}}$ — множество объектов онтологической модели для отслеживания в ИП, $D_{\text{sub}}$ — множество RDF-троек, представляющие отслеживаемые объекты в формате ИП-интерфейса, $D_{\text{ind}}$ — множество RDF-троек для представления изменений отслеживаемых объектов в формате ИП-интерфейса. При программировании логики агента разработчику нужно оперировать непосредственно только с объектами из $O_{\text{src}}$ — множества троек $D_{\text{sub}}$ и $D_{\text{ind}}$ скрыты от разработчика, обновление $O_{\text{src}}$ выполняется на стороне агента автоматически. Для этих целей в онтологическую библиотеку включают операции $SUBDATA(O_{\text{src}})$ и $UPDATE(D_{\text{ind}})$. Первая транслирует объекты $O_{\text{src}}$ в представление $D_{\text{sub}}$ для подписки на уровне ИП-интерфейса. Вторая обновляет объекты $O_{\text{src}}$ на основе $D_{\text{ind}}$. Параметры операции подписки — объекты из $O_{\text{src}}$. В результате, разработчик оперирует небольшим количеством объектов $O_{\text{src}}$ вместо RDF-троек ($\left|D_{\text{src}}\right| << \left|D_{\text{ind}}\right|$).

Пример использования модели — отслеживание присутствия людей в системе «умный дом». При входе/выходе человека агент сенсора присутствия обновляет в ИП свойства индивидов, представляющих людей. В зависимости от числа присутствующих агент управления климатом изменяет в ИП параметры требуемого в помещении климата, и агенты климатического оборудования настраивают работу оборудования.

*Модель взаимодействия агентов на основе обработки группы объектов* используется агентом как $(e, F, q, r)$, где $e$ — отслеживаемое событие (обнаруживается локально или как изменение в ИП), $F$ — множество возможных операции над объектами, $q$ — правило обработки, привязывающее операции к объектам, $r$ — набор {операция – объект} (запрос на обработку информационного содержимого ИП).

Сценарии использования ИП основаны на причинно-следственных связях в проблемной области (рис. 3). Определяются события (причины) и граф зависимостей, в котором представлены цепочки обработки группы объектов проблемной области (следствия). Для заданного события $e$ в логике агента программируется правило обработки $q$, которое реализует обработку группы объектов (на стороне агента), формируя в итоге запрос $r$ с операциями из $F$ для изменения объектов в ИП. С учетом параллельной активности других агентов операции из $r$ объединяются в один сетевой запрос от агента к ИП для выполнения транзакции на множественное изменение.

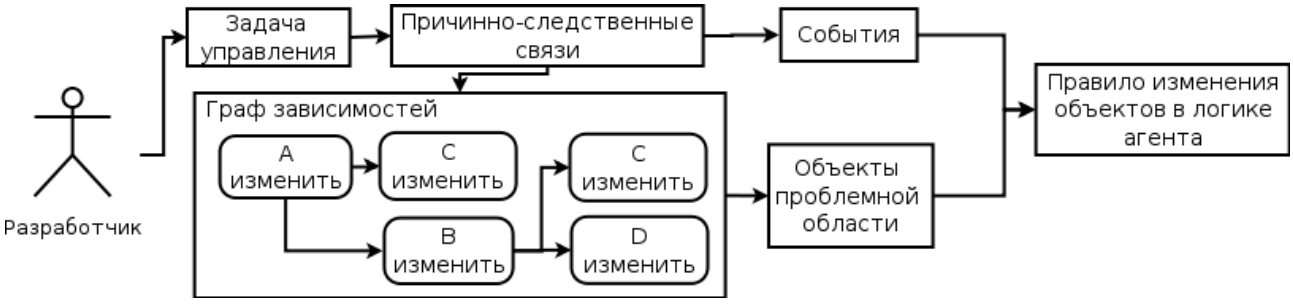


**Рисунок 3. Построение правила обработки объектов проблемной области.**

Пример использования модели — отслеживание присутствия людей в помещении для управления климатом в системе «умный дом». Обнаружение входа/выхода человека вызывает обновление группы объектов: персональная информация о человеке, текущее число людей в помещении, параметры климатического оборудования. Выполнении операции неделимым образом (транзакция) гарантирует семантическую целостность изменений, поступающих в информационного содержимого ИП.

**В третьей главе** приведены полученные механизмы программирования, поддерживающие предложенные метод автоматизированной онтолого-ориентированной разработки ПО и модели взаимодействия агентов.

*Механизм генерации программного кода онтологической библиотеки* формирует частную онтологическую модель для агента на основе интеграции онтологий проблемной области и выбора объектов для взаимодействия с другими агентами. Определяются следующие шаги: 1) построение общей модели проблемной области $O = o_1 + o_2 + ... + o_n$ при помощи редактора Protégé, 2) формирование частной онтологической модели $m$ для агента на основе выбранных разработчиком объектов из $O$, 3) генерация программного кода части $l_{\text{data}}$ онтологической библиотеки для реализации работы с $m$ в логике агента.

Выбор объектов выполняется на уровне OWL классов и свойств (включать, не включать). В результате можно уменьшить объем программного кода части $l_{\text{data}}$, снижая емкостную трудоемкость в оценке (1). В предлагаемом механизме результат выбора объектов сохраняется модулем расширения функциональности (плагином) редактора Protégé в фильтр-файл, содержащий URI объектов. Исходные онтологии не изменяются и можно создать нескольких фильтр-файлов. На шаге 3 можно выбрать язык программирования, на котором генерировать программный код.

Пример использования механизма — система «умный дом», где применяются онтологии для описания людей (FOAF), их присутствия (Online Presence Ontology), сенсоров (OntoSensor) и т.д. Агентам климатического оборудования не нужна информация о социальных связях людей. Такая информация нужна персональным агентам на мобильных телефонах. Отсутствие выбора объектов привело бы к дополнительному использованию ресурсов (память для хранения структур данных).

*Механизм программирования логики агента на основе модели сессии* определяет схему использования сессии и операций управления, определенных в модели, для управления сессией, сеансами, подписками и локальным хранилищем (рис. 4).

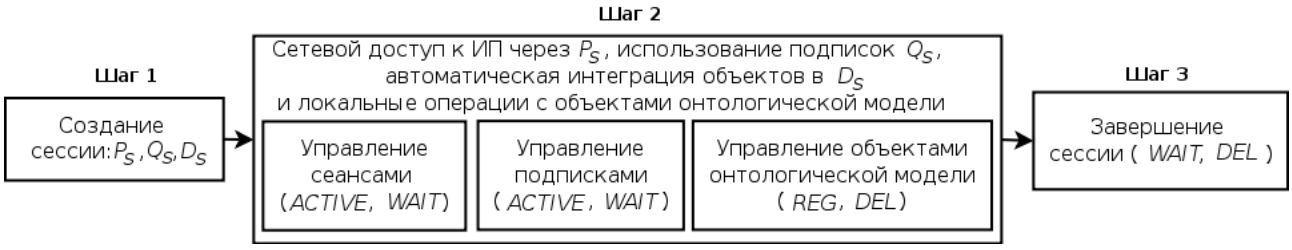


**Рисунок 4. Схема использования сессии при программировании логики агента.**

Шаг 1. Создание сессии с необходимым набором сеансов и подписок. Структура локального хранилища агента определяется частной онтологической моделью, объекты которой регистрируются в сессии (операция *REG*).

Шаг 2. Программирование логики агента с использованием операций управления: активация, ожидание сеансов и подписок, регистрация и удаление объектов из

локального хранилища. Запросы к ИП используют зарегистрированные в сессии объекты. Интеграция поступающих из ИП объектов выполняется автоматически в локальном хранилище на основе семантических связей объектов.

Шаг 3. Завершение сессии, переводя в состояние ожидания все подписки, сеансы и удаляя их (операции *WAIT* и *DEL*).

Пример использования сессии — управление подписками персональных агентов в системе «умный дом». При выходе человека из дома во двор часть подписок переводится в состояние ожидания (напр., отслеживание текущей мелодии на музыкальном проигрывателе). При возвращении человека в дом подписки активируются.

*Механизм программирования логики агента на основе модели онтолого-ориентированной операции подписки* определяет, какие объекты могут быть заданы для отслеживания изменений и формат RDF-представлений для подписываемых объектов и получаемых уведомлений. Поддерживаются два вида отслеживания объектов $O_{\text{src}}$ в ИП: 1) подписка на свойства (изменение свойств индивидов) и 2) подписка на классы (появление/удаление индивидов). В первом случае объекты в $O_{\text{src}}$ — наборы «индивид – список свойств», во втором — наборы классов. Допускается смешанный вариант с заданием в $O_{\text{src}}$ различных наборов. Поддерживается как синхронный, так и асинхронный способ работы операции подписки, позволяющий реализовать логику агента в виде набора параллельных вычислительных потоков. Данных видов подписки достаточно для базовых действий агента при взаимодействии в ИП, они могут быть расширены в зависимости от возможностей конкретного ИП и ИП-интерфейса.

Онтологическая библиотека автоматически сводит операцию подписки к RDF-ориентированным операциям ИП-интерфейса. Последний выполняет запросы на основе Т-шаблонов (RDF-тройки с масками на месте элементов). Операция *SUBDATA* транслирует объекты для отслеживания изменений ($O_{\text{src}}$) в набор Т-шаблонов ($D_{\text{sub}}$). Операция *UPDATE*: 1) интерпретирует RDF-тройки ($D_{\text{ind}}$), определяя URI индивидов, классов и свойств и 2) синхронизирует объекты ($O_{\text{src}}$) с содержимым ИП, создавая, удаляя или обновляя индивидов (изменяя свойства) на стороне агента.

Пример использования подписки — отслеживание агентом управления климатом в системе «умный дом» такие параметры как температура, влажность (публикуются в ИП агентами сенсорного оборудования). Аналогично, подписка на класс «человек» позволяет отслеживать появление человека.

*Механизм программирования логики агента на основе модели для обработки группы объектов* реализует дополнительное свойство, ассоциируемое с объектом онтологической модели (только на стороне агента). Механизм позволяет определить

требуемую обработку группы объектов, представленных OWL-индивидами, с автоматическим построением запроса на множественные изменения в ИП как транзакции.

Свойство-обработчик объявляется в логике агента для основного объекта из группы, изменение которого приводит к изменению других объектов. Программируется правило обработки $q$ как процедура $f(e)$. Программируется отслеживание события $e$ и вызов $q = f(e)$. Такое отслеживание удобно реализовать на основе подписки на свойства (см. предыдущий механизм), что позволит автоматически запускать обработку группы объектов. Работа со свойством-обработчиком сходна с работой с онтологическими свойствами индивида. Обновление свойства означает наступление события, значением выступает само событие $e$. При наступлении $e$ агентом автоматически выполняются следующие шаги (рис. 5).

Шаг 1. Вызов правила обработки $q$.

Шаг 2. Формирование правилом $q$ запроса $r$ на основе операций над свойствами индивидов: $F = \{INS, DEL, UPD, QRY\}$ (установить, удалить, обновить, получить).

Шаг 3. Преобразование запроса $r$ в RDF-формат ИП-интерфейса.

Шаг 4. Выполнение транзакции к ИП, объединяющей операции из $r$.


**Рисунок 5. Использование свойства-обработчика для группы объектов.**

Пример использования обработки группы объектов — отслеживание агентом управления климатом в системе «умный дом» состояния физического окружения и людей (описываются группой OWL-индивидов). Изменение свойства одного индивида влечет изменение свойств у других индивидов. Так, изменение числа присутствующих людей требует обновления уровня поддерживаемой влажности и скорости работы вентилятора. Агент локально изменяет свойства индивидов и вносит все изменения в ИП в виде одной транзакции.

**В четвертой главе** рассматривается платформа Smart-M3 для развертывания ИП и авторский инструментарий SmartSlog для разработки агентов для таких ИП.

Платформа Smart-M3 позволяет развернуть ИП в IoT-среде на основе разделяемого RDF-хранилища. Программные агенты в терминах Smart-M3 называются процессорами знаний (KP, далее — агенты KP). Сетевой доступ использует протокол SSAP (Smart Space Access Protocol), его клиентская часть реализует ИП-интерфейс.

Инструментарий SmartSlog включает представленные в гл. 3 механизмы программирования. Онтологическую библиотеку (далее — библиотека SmartSlog) можно генерировать на языках ANSI C (для различных устройств, в т.ч. маломощных) и C# (для Windows-устройств). Для библиотеки SmartSlog реализован интерфейс программного адаптера, через который подключаются различные ИП-интерфейсы (KPI).

Разработка для разных аппаратно-программных платформ поддерживается за счет генерации библиотеки SmartSlog, выбора языка программирования, использования программных адаптеров для ИП-интерфейсов и снижения затрат в оценках (1) и (2). Применимость проверена для платформ из табл. 2. ANSI C вариант библиотеки SmartSlog зависит только от библиотеки ИП-интерфейса и POSIX-библиотеки вычислительных потоков. В агенте KP для ОС Android разработчику достаточно реализовать взаимодействие интерфейса пользователя (Java) и логики агента (ANSI C). Для C# варианта библиотеки SmartSlog реализованы адаптеры для C KPI, позволяя разрабатывать агентов KP и для ОС Windows Phone.

| | **Windows** | **Windows Phone** | **На основе Linux** | **Mac OS** | **Android** |
|---|---|---|---|---|---|
| **ANSI C** (C KPI) | **+** | **–** | **+** | **+** | + (взаимодействие ANSI C и Java кода) |
| **C#** (C KPI) | **+** (адаптер) | **+** (адаптер) | Mono (.NET Framework на базе свободного ПО) | | **–** |
| **C#** (C# KPI) | **+** | **+** | | | **–** |

**Таблица 2. Платформы, поддерживаемые инструментарием SmartSlog.**

Уменьшение объема программного кода логики агента показано на рис. 6 в зависимости от шаблона реализации доступа агента к ИП. Сравнивается объем программного кода, требуемый для реализации разработчиком с помощью библиотеки SmartSlog и непосредственно на уровне RDF-представления через ИП-интерфейс (в экспериментах используется интерфейс C KPI). Выигрыш по количеству операций в логике агента в среднем достигает 35%, т.к. при работе через ИП-интерфейс разработчику необходимо самостоятельно программировать обработку наборов RDF-троек (провести их поиск, проверку и обновление). Уменьшение объема программного кода логики агента также приводит к уменьшению цикломатической сложности, в результате требуется меньшее число тестов для покрытия кода.

Потеря производительности из-за использования библиотеки SmartSlog в сравнении с ИП-интерфейсом показана в табл. 3. Экспериментально обнаружено отсутствие значимых затрат. Для персональной ЭВМ (i5-2520M, 256 Мб, Linux) средние затраты времени растут на 0.15 мс на каждые 15000 RDF-троек, представляющих объекты онтологической модели. Дополнительный объем памяти также незначителен — хранение индивида и свойства, требуется 48 и 20 байт для структур данных (напр.,

для $10^5$ индивидов и свойств нужно около 6640 Кб). Исходные RDF-тройки не хранятся на стороне агента КР после их преобразования в индивиды и свойства.

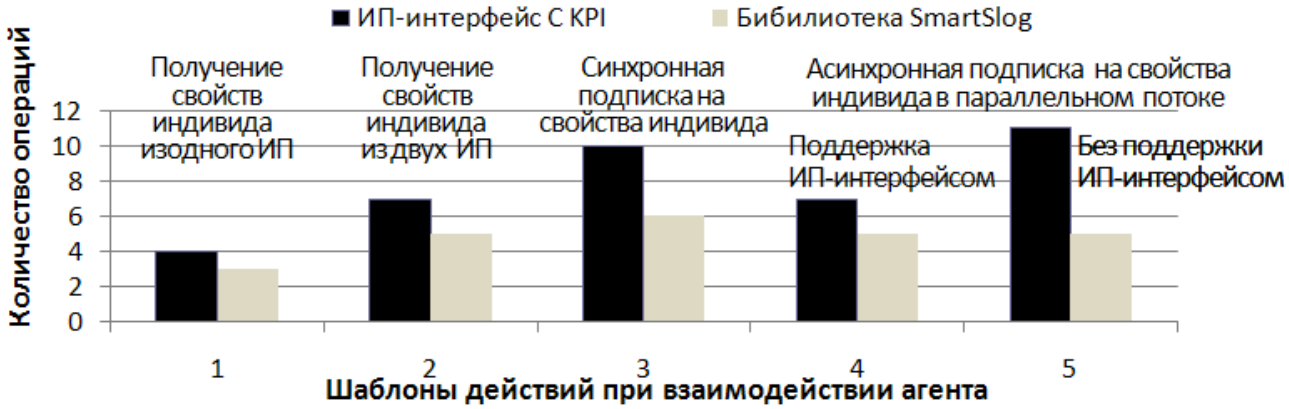


**Рисунок 6. Количество операций для реализации взаимодействия агента.**

Практическая эффективность инструментария SmartSlog показана на примере разработки агентов КР для системы интеллектуального зала SmartRoom, внедряемой в ИТ-парке ПетрГУ. Портативный вариант этой системы также используется для проведения секций международных конференций Ассоциации открытых инноваций FRUCT. Агенты КР разрабатываются для таких ограниченных устройств как персональные мобильные компьютеры (телефоны, планшеты), одноплатные компьютеры с сенсорами (Raspberry Pi), презентационные устройства (мультимедийные доски). Для агентов КР системы SmartRoom используются все типовые варианты работы, в особенности 1 и 5 (см. рис. 6), т.е. применение инструментария SmartSlog уменьшает примерно в два раза объем программного кода в логике агента. Использование сессий позволяет автоматизировать проверку работоспособности сеансов сетевого доступа от мобильных устройств к ИП и восстановление сеансов, когда связь возобновляется. Свойство-обработчик позволяет объединять запросы к ИП, снижая сетевую нагрузку.

| **Число RDF-троек** | **1** | **15000** | **30000** | **45000** | **60000** | **75000** | **90000** | **100000** |
|---|---|---|---|---|---|---|---|---|
| **Интерфейс С KPI (мс)** | 0,00 | 1,88 | 3,26 | 5,26 | 7,35 | 9,48 | 10,74 | 11,85 |
| **Библиотека SmartSlog (мс)** | 0,00 | 2,02 | 3,55 | 5,77 | 8,03 | 10,24 | 11,66 | 12,93 |
| **Разница (мс)** | 0,00 | 0,14 | 0,29 | 0,51 | 0,68 | 0,76 | 0,92 | 1,08 |

**Таблица 3. Время обработки на стороне агента КР уведомлений по подписке**

Полученные результаты подтверждают эффективность разработки агентов ИП с использованием библиотек SmartSlog, которые включают реализацию механизмов на основе предложенных моделей взаимодействия. Автоматизируются задачи программирования логики агента (см. табл. 1) без существенной потери производительности и затрат памяти.

## ЗАКЛЮЧЕНИЕ

Совокупность полученных в диссертационном исследовании результатов представляет собой решение актуальной задачи повышения эффективности разработки

программных агентов для интеллектуальных пространств. Внедрение результатов вносит вклад в развитие онтолого-ориентированного подхода для автоматизации разработки программного обеспечения интеллектуальных пространств. В ходе исследования получены следующие основные результаты:

1) Разработан метод автоматизированной онтолого-ориентированной разработки агентов, позволяющий программировать логику агента на основе моделей косвенного взаимодействия агентов с использованием объектов частной онтологической модели проблемной области.
2) Разработана модель взаимодействия агентов на основе сессии, позволяющая агенту использовать параллельные сеансы сетевого доступа к ИП и автоматически интегрировать объекты из нескольких ИП.
3) Разработана модель взаимодействия агентов на основе операции подписки позволяющая агенту отслеживать изменения объектов в ИП и автоматически их синхронизировать.
4) Разработана модель взаимодействия агентов на основе обработки группы объектов, позволяющая программировать правило обработки объектов как реакцию агента на наступление событий в ИП и вносить в виде транзакций множественные изменения в ИП по результатам обработки.
5) Предложены программные механизмы онтолого-ориентированной разработки агентов и апробированы в инструментарии SmartSlog платформы Smart-M3 для генерации программного кода онтологических библиотек и программирования логики агентов. Механизмы позволяют уменьшить объем создаваемого прикладным разработчиком вручную программного кода агента, поддерживают использование различных языков программирования, сохраняют возможность программирования агентов для маломощных устройств.

Полученные результаты соответствуют п. 5 «Теория построения программ, пакетов, прикладных программ, программных комплексов, а также сетевых программ, в том числе, поддерживающих сетевые протоколы», п. 9 «Модели и методы разработки программных средств обработки данных и знаний в вычислительных машинах, вычислительных комплексов и компьютерных сетях» и п. 12 «Программные инструментальные средства разработки интеллектуальных систем, в том числе экспертных систем, систем поддержки принятия решений, обучающих систем и др.» и п. 16 «Теория и практика технологических аспектов программирования, изготовления и эксплуатации программ, ПК, ППП и СП, а также программных и инструментальных технологических комплексов поддержки разработки программных средств» паспорта специаль-

ности 05.13.11 «Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей».

**Работы автора по теме диссертации в рецензируемых журналах из перечня ВАК**

1. Корзун Д.Ж. Автоматизированная модельно-ориентированная разработка программных агентов для интеллектуальных пространств на платформе Smart-M3 / Д. Ж. Корзун, **А. А Ломов**, П. И. Ванаг // Теоретический и прикладной научно-технический журнал Программная инженерия. №5. 2012 г. С. 6-14.
2. **Ломов А. А.** Операция подписки для приложений в интеллектуальных пространствах платформы Smart-M3 / А. А. Ломов, Д. Ж. Корзун // Труды СПИИРАН. Вып. 4(23). 2012 г. С.439-458.
3. **Ломов А.А.** Взаимодействие программного агента на уровне сессии с интеллектуальным пространством // Ученые записки ПетрГУ. Вып. 8 (137). 2013 г. С. 118-122.

**Работы автора по теме диссертации в других изданиях**

4. Korzun D. G. Generating Modest High-Level Ontology Libraries for Smart-M3 / D. G. Korzun, **A. A. Lomov**, P. I.Vanag, S. I. Balandin, J. Honkola // Proc. 4th Int'l Conf. on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM 2010). October 25 – 30. 2010. Florence. Italy. Pp. 103-109 (индексируется в базе данных Scopus)
5. Korzun D. G. SmartSlog 0.3x: New Ontology Library API and Optimization / D. G. Korzun, **A. A. Lomov**, P. I. Vanag // Proc. 8th Conf. Open Innovations Framework Program FRUCT. November 9–12. 2010. Lappeenranta. Finland. Pp. 79-83.
6. Korzun D. G. Multilingual Ontology Library Generator for Smart-M3 Application Development / D. G. Korzun, **A. A. Lomov**, P. I.Vanag // Proc. 9th Conf. Open Innovations Framework Program FRUCT. Petrozavodsk. Russia. 26-29 April 2011. Pp. 82-92.
7. **Lomov A. A**. Subscription Operation in Smart-M3. / A. A. Lomov, D. G. Korzun // Proc. 10th Conf. Open Innovations Association FRUCT and 2nd Finnish-Russian Mobile Linux Summit. Tampere. Finland. 7-11 Nov. 2011. pp.83-94.
8. Korzun D. G. Multilingual Ontology Library Generator for Smart-M3 Information Sharing Platform. / D. G. Korzun, **A. A. Lomov**, P. I.Vanag, S. I. Balandin, J. Honkola // International Journal On Advances in Intelligent Systems, 2011, vol 4, nr 3&4, pp.68-81.
9. **Lomov A. A.** SmartSlog Session Scheme for Smart-M3 Applications. // Proc. 12th Conf. Open Innovations Association FRUCT. Oulu. Finland. 5-9 Nov. 2012. pp.66-71.
10. **Lomov A. A.** Ontology-based KP Development for Smart-M3 Applications. // Proc. 13th Conf. Open Innovations Association FRUCT, Petrozavodsk, Russia, 22-26 Apr. 2013. pp.94-100.