На правах рукописи

ЛОМОВ
Александр Андреевич

МОДЕЛИ И МЕХАНИЗМЫ ДЛЯ АВТОМАТИЗАЦИИ ПРОГРАММИРОВАНИЯ КОСВЕННОГО ВЗАИМОДЕЙСТВИЯ АГЕНТОВ ИНТЕЛЛЕКТУАЛЬНЫХ ПРОСТРАНСТВ

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: **Андрей Николаевич Терехов**
доктор физико-математических наук, профессор
заведующий кафедрой системного программирования ФГБОУ ВПО «Санкт-Петербургского государственного Университета» (СПбГУ)

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 «18» ноября 2014 года в 15:30 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: _____

Ученый секретарь
диссертационного совета Д 002.199.01
доктор физико-математических наук,
профессор Баранов Сергей Николаевич

Общая характеристика работы

Актуальность темы. Развитие технологий Интернета физических устройств (Internet of Things, IoT) и сервисно-ориентированных систем для вычислительных сред «повсеместных вычислений» позволяет применять на практике парадигму интеллектуальных пространств. Интеллектуальное пространство (ИП) обеспечивает динамическое множество участников вычислительной среды контекстно-зависимой информацией, сервисами и персонифицированными рекомендациям. Участниками выступают автономные агенты на разнообразных устройствах вычислительной среды. Агент — это программа, прикладная логика которой основана на 1) наблюдении состояния среды и ее изменений, 2) интерпретации наблюдений для определения собственных действий и 3) внесении своих изменений в среду. Интеллектуальные пространства различаются по типу вычислительной среды, способу организации взаимодействия агентов и представлению разделяемого информационного содержимого.

Широкое применение получили интеллектуальные пространства, разворачиваемые в ограниченных физических окружениях (напр., в доме или офисе — системы Z-Wave, KNX, C-Bus). В случае открытых ИП состав участников, представленных агентами на разнообразных вычислительных устройствах, динамически меняется (напр., ИП в общественных местах — конференц-зал, зона отдыха и т.п.). Возникает проблема интероперабельности (возможности взаимодействия), включая вопросы сетевого взаимодействия агентов и разделения ими информационного содержимого.

Выделяют интеллектуальные пространства с косвенным взаимодействием агентов на основе онтолого-ориентированного представления и обработки разделяемого агентами информационного содержимого. Вычислительная среда для таких ИП удовлетворяет условиям IoT — разнообразные устройства локализованы в физическом окружении и могут выполнять сетевые коммуникации как друг с другом, так и с внешними системами. Взаимодействие агентов организуется на основе базовых моделей «классная доска» и «публикация/подписка». Разработчик использует примитивы доступа агента к ИП для реализации косвенного взаимодействия, определяя обработку разделяемого информационного содержимого ИП. Последнее представлено в общем хранилище на основе модели RDF (Resource Description Framework) и интерпретируется агентами при помощи онтологий проблемной области на языке OWL (Web Ontology Language). Модель RDF и OWL-онтологии наследуются из Семантического веб. Выделяют основные типы косвенного взаимодействия: интеллектуальное соединение (ожидание некоторых событий для начала действий), централизованное хранилище (доступ к хранилищам информации), концентратор информации (получение и

интеграция информации из распределенных источников) и глобальный посредник (организация обмена информацией между агентами).

Для программирования взаимодействия в логике агента в терминах OWL-онтологий (классы, свойства, индивиды) необходимо их представление в программном коде и функции для работы с этими представлениями. Создание такого представления возможно на основе модельно-ориентированного подхода, а функции могут быть реализованы на уровне промежуточного ПО. Онтологическая модель проблемной области определяет информационные объекты (далее — объекты), посредством которых агент взаимодействует с другими агентами. Промежуточное ПО позволяет связать такие объекты с конструкциями языка программирования и представить объекты в программном коде логики агента структурами данных. На уровне промежуточного ПО реализуются механизмы для автоматизации программирования косвенного взаимодействия агентов с использованием объектов онтологической модели. Разработчику для использования механизмов предоставляются шаблоны программного кода. Основную сложность для построения таких механизмов создает наблюдаемое разнообразие аппаратно-программных платформ в IoT-средах.

Целью диссертационной работы является повышение эффективности разработки агентов в терминах проблемной области за счет автоматизации программирования косвенного взаимодействия агентов на основе технологий Семантического веб и с учетом разнообразия аппаратно-программных вычислительных платформ.

Объектом исследования являются программные агенты интеллектуального пространства, применяющие 1) базовые модели «классная доска», «публикация/подписка» для взаимодействия агентов и 2) RDF-модель и OWL-онтологии для представления и обработки разделяемого информационного содержимого.

Предметом исследования являются специализированные модели косвенного взаимодействия и механизмы программирования для основных типов взаимодействия агентов, обеспечивающие 1) автоматизированную разработку программного кода логики агента с учетом разнообразия аппаратно-программных вычислительных платформ, 2) выполнение агентом интеграции, синхронизации и обработки информационного содержимого при его взаимодействии с другими агентами в ИП на уровне объектов онтологической модели проблемной области.

Для достижения поставленной цели в работе решаются следующие задачи.

1. Анализ существующих методов разработки программных агентов и основных типов их взаимодействия для определения путей упрощения разработки за счет применения модельно-ориентированного подхода в условиях IoT-сред.

2. Разработка метода программирования косвенного взаимодействия агентов с использованием объектов онтологической модели проблемной области и механизмов программирования для упрощения реализации основных типов взаимодействия.

3. Разработка специализированных моделей косвенного взаимодействия агентов на основе базовых моделей для автоматической интеграции агентом объектов из ИП, автоматической синхронизации агентом объектов и локальной обработки агентом групп объектов с автоматическим построением запросов к ИП в виде транзакций.

4. Разработка механизмов программирования для использования предлагаемых моделей взаимодействия и автоматизации программирования логики агентов при реализации основных типов взаимодействия агентов.

5. Реализация и апробация программного инструмента для применения предлагаемых метода и механизмов программирования взаимодействия агентов с учетом разнообразия аппаратно-программных вычислительных платформ.

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

Основные положения, выносимые на защиту.

1. Метод программирования косвенного взаимодействия агентов на основе специализированных моделей взаимодействия с использованием технологий Семантического веб для автоматизации разработки программных агентов.

2. Модель взаимодействия агентов на основе многоэлементной сессии для организации параллельного сетевого доступа агента к нескольким ИП и автоматической интеграции агентом объектов онтологической модели из этих ИП.

3. Модель взаимодействия агентов на основе операции подписки для автоматизации отслеживания агентом происходящих в ИП изменений на уровне объектов онтологической модели.

4. Модель взаимодействия агентов на основе локальной обработки группы объектов онтологической модели для автоматического формирования агентом запроса на множественные изменения в ИП.

5. Механизмы программирования косвенного взаимодействия агентов на основе предложенных специализированных моделей и реализация этих механизмов в программном инструменте SmartSlog платформы Smart-M3.

Научная новизна работы состоит в следующем.

1. Разработан метод программирования косвенного взаимодействия агентов на основе онтологических библиотек, отличающийся от существующих методов поддержкой разнообразных аппаратно-программных вычислительных платформ для выполнения агентов и позволяющий автоматизировать программирование основных типов взаимодействия, реализуемых в логике агента.

2. Разработана модель взаимодействия агентов на основе многоэлементной сессии, отличающаяся от существующих моделей поддержкой параллельных сеансов сетевого доступа с автоматической интеграцией информации в локальном хранилище агента и позволяющая организовывать и управлять группами подписок в рамках сеансов для упрощения программирования сетевых операций доступа к нескольким ИП.

3. Разработана модель взаимодействия агентов на основе операции подписки, отличающаяся от существующих моделей поддержкой отслеживания изменений в ИП на уровне объектов онтологической модели с автоматической синхронизацией этих объектов в локальном хранилище агента и позволяющая уменьшить в логики агента объем программируемых сетевых операций доступа к ИП.

4. Разработана модель взаимодействия агентов на основе локальной обработки группы объектов, отличающаяся от существующих моделей возможностью программировать операции над группой объектов онтологической модели и позволяющая автоматически формировать агенту запрос к ИП на множественное изменение в виде одной транзакции и поддерживать семантическую целостность информационного содержимого.

5. Реализованы механизмы программирования косвенного взаимодействия для логики агента в созданном программном инструменте разработки SmartSlog платформы Smart-M3 для применения предложенного метода, отличающиеся от существующих механизмов автоматизацией программирования основных типов взаимодействия в ИП на уровне объектов онтологической модели и поддерживающие разработку агентов для разнообразных вычислительных устройств.

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 и семинарах.

Практическая ценность работы. Предложенные специализированные модели взаимодействия агентов используются в полученном методе программирования косвенного взаимодействия агентов. Механизмы программирования реализованы авто-

ром в инструменте разработки с открытым исходным кодом SmartSlog платформы Smart-M3. Инструмент 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. В частности, инструмент SmartSlog 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 — совместной 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.; научных семинарах СПИИРАН, 2013-2014 гг.; научном семинаре ИИММ КНЦ РАН, 2013 г.; научном семинаре кафедры системного программирования СПбГУ, 2014 г.; научном семинаре ИПМИ КарНЦ РАН, 2014 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. Получено свидетельство о регистрации программы ЭВМ (выдано Федеральной службой по интеллектуальной собственности).

Объем и структура работы. Диссертация объемом 137 машинописных страниц состоит из введения, 4 глав, заключения, списка использованной литературы и приложения. Текст диссертации содержит 29 рисунков и 8 таблиц. Список использованной литературы содержит 98 наименований.

Содержание работы

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

В первой главе рассмотрены интеллектуальные пространства обеспечивающие пользователей контекстно-зависимой информацией, сервисами и персонифицированными рекомендациями. В качестве примера приведена система «умный дом».

Реализация интеллектуальных пространств зависит от типа вычислительной среды, модели сетевого взаимодействия агентов и способа организации разделяемого агентами информационного содержимого. Выделяют интеллектуальные пространства с косвенным взаимодействием, где агенты динамически формируют информационное содержимое ИП, используя технологий Семантического веб для представления и обработки информации. Проблемная область описывается набором онтологий, позволяющих интерпретировать информационное содержимое ИП в терминах объектов и их связей. Агент следует частной онтологической модели, определяющей подмножество объектов, требуемых агенту для взаимодействия с другими агентами.

Известны основные типы взаимодействия агентов в ИП: интеллектуальное соединение, централизованное хранилище, концентратор информации и глобальный посредник. При их программировании в логике агента разработчик сталкивается со следующими проблемами. 1. Программирование сетевых аспектов взаимодействия агента с другими агентами. 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 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 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 объектов онтологической модели в виде RDF-троек. Сете-

вой доступ к ИП — через RDF-ориентированный ИП-интерфейс. Разработчик реализует взаимодействие, используя примитивы доступа ИП. Высокоуровневая разработка (уровень OWL-объектов) 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 с реализованными моделями взаимодействия в виде механизмов. Механизмы дополняются шаблонами программного кода, которые разработчик использует при программировании логики агента. Промежуточное ПО увеличивает общий объем программного кода агента, но упрощает программирование логики агента за счет приближения к проблемной области и автоматизации косвенного взаимодействия. В результате, уменьшается программный код логики агента, который необходимо создать и протестировать прикладному разработчику.

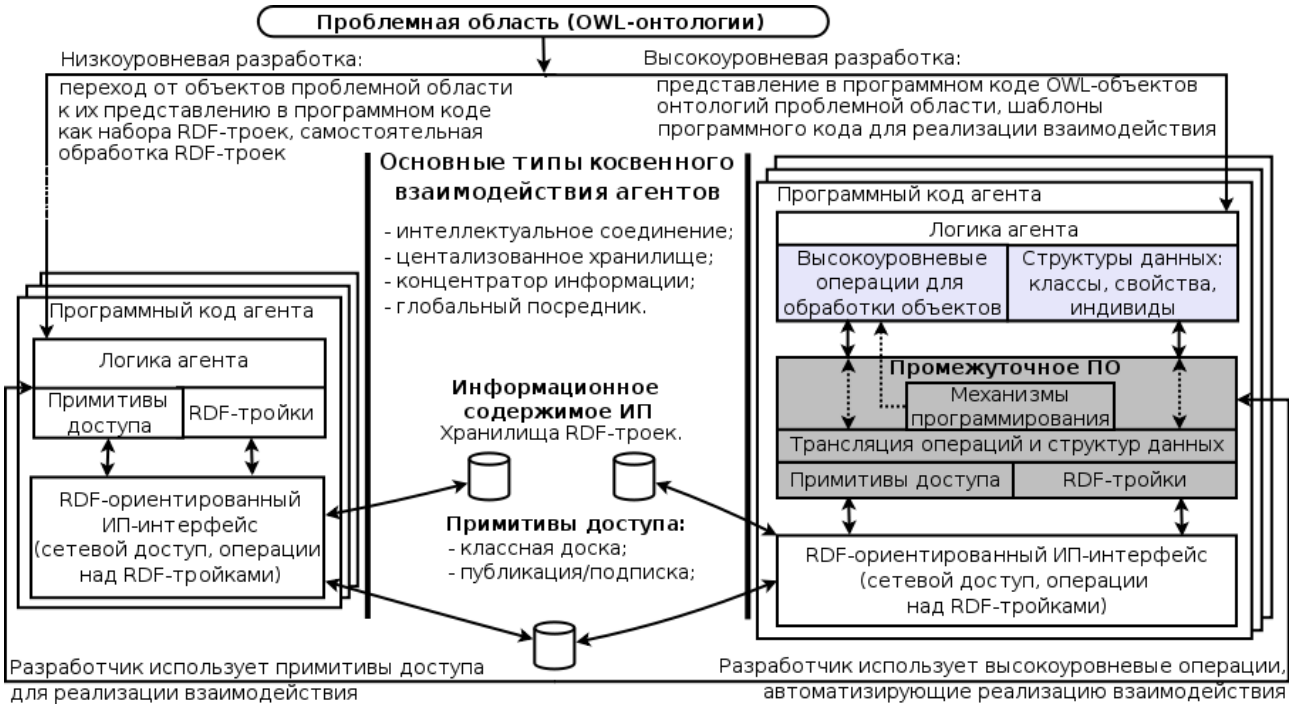

Рисунок 1. Низкоуровневая и высокоуровневая разработка программных агентов для интеллектуальных пространств.

Базовые задачи программирования взаимодействия в логике агента приведены на рис. 2. Их решение на уровне RDF-представления требует от разработчика 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 с использованием онтолого-ориентированного подхода и промежуточного ПО. На основе онтологий автоматизируется проверка корректности и проводится интеграция информации. В промежуточном ПО реализуются механизмы для

программирования взаимодействия и шаблоны их использования в логике агента на основе объектов онтологической модели проблемной области.

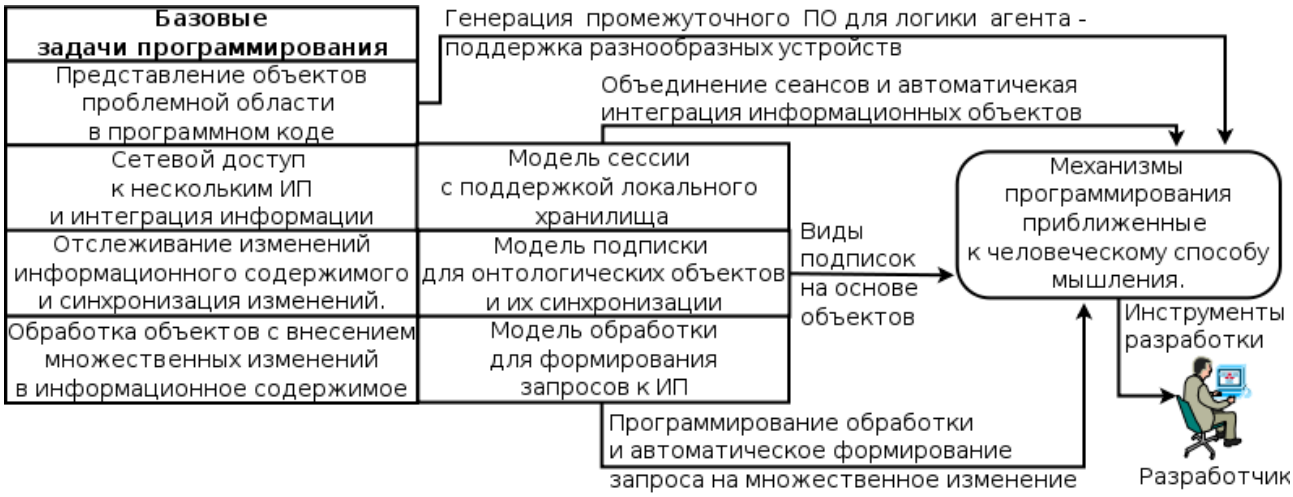

Рисунок 2. Базовые задачи программирования взаимодействия в логике агента.

Во второй главе предложен метод программирования косвенного взаимодействия агентов для упрощения реализации основных типов взаимодействия на уровне объектов онтологической модели. Промежуточное ПО для высокоуровневой разработки агента реализуется как программная онтологическая библиотека функций, предоставляющая разработчику логики агента интерфейс прикладного программирования, параметризируемый объектами онтологической модели. Метод включает следующие шаги (рис. 3), 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 онтологической библиотеки, 3) программирование в логике агента взаимодействия с другими агентами, используя структуры данных и функции из онтологической библиотеки.

Онтологическая библиотека состоит из частей l_{data} и l_{func}. В l_{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_{data} с выбором языка программирования. Инвариантная функциональная часть l_{func} содержит функции для программирования взаимодействия агента в общеонтологических терминах — «класс», «свойство», «индивид». Тем самым, программная реализация l_{func} зависит от целевой аппаратно-программной платформы без привязки к конкретной проблемной области.

В логике агента используются вызовы функций из l_{func} с параметрами на основе структур данных из l_{data}. При организации сетевого доступа агента к ИП онтологиче-

ская библиотека выполняет преобразования между структурами данных l_{data} и RDF-представлением. Часть l_{func} поддерживает различные ИП-интерфейсы, чтобы обеспечить разработку агентов для разнообразных аппаратно-программных платформ.

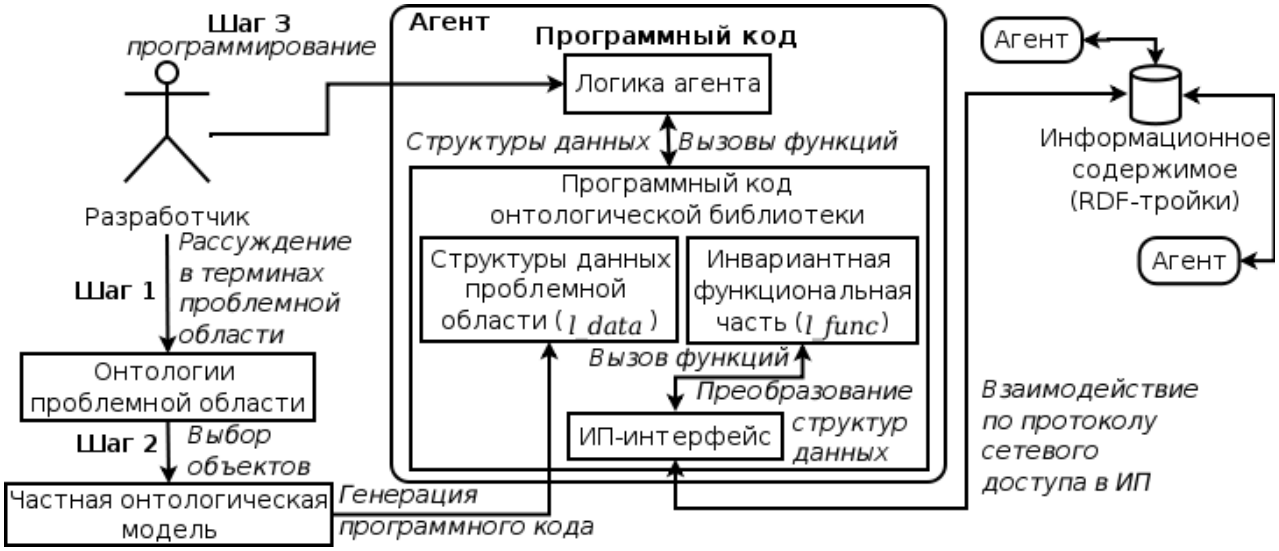

Рисунок 3. Схема использования онтологической библиотеки в рамках метода программирования косвенного взаимодействия.

Емкостная трудоемкость онтологической библиотеки оценивается как

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

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

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

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

где $C_{\text{obj-trp}}$ и $C_{\text{trp-obj}}$ — средние затраты на преобразование (из OWL-объектов в RDF-тройки и обратно), λ — интенсивность вызова функций онтологической библиотеки в вычислительном потоке, n_{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_{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 оборудова-

ние. Поскольку агент сенсора оперирует с небольшим набором измеряемых параметров и операции доступа к ИП сводятся к простым операциям чтения/записи, то генерируемая для агента часть l_{data} не включает трудоемких программных конструкций.

Предложенный метод требует специализированные модели взаимодействия для программирования логики агента (шаг 3) на уровне объектов онтологической модели. В работе предлагаются три таких модели.

Модель взаимодействия агентов на основе многоэлементной 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 вычислительного потока на стороне агента. Подписка привязана к сетевому сеансу. Локальное хранилище разделено между сеансами и подписками, что автоматизирует 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 отслеживаются изменения в ИП). В состоянии ожидания сеанс или подписка определены только на стороне агента (сетевое соединение 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 операции подписки используется агентом как $(O_{\text{src}}, D_{\text{sub}}, D_{\text{ind}})$, где O_{src} — множество объектов онтологической модели для отслеживания в ИП, D_{sub} — множество RDF-представлений отслеживаемых объектов в формате ИП-интерфейса, D_{ind} — множество RDF- представлений поступаю-

щих изменений для отслеживаемых объектов в формате ИП-интерфейса. При программировании логики агента разработчик оперирует только объектами из O_{src}, обновление которых выполняется автоматически. Множества RDF-представлений D_{sub} и D_{ind} скрыты от разработчика. Модель определяет операции $SUBDATA(O_{\text{src}})$ и $UPDATE(D_{\text{ind}})$. Первая транслирует объекты O_{src} в представление D_{sub} для подписки на уровне ИП-интерфейса. Вторая обновляет объекты O_{src} на основе D_{ind}. Параметрами операции подписки являются объекты из O_{src}. В результате, разработчик оперирует небольшим количеством объектов O_{src} вместо RDF-троек ($|D_{\text{src}}| << |D_{\text{ind}}|$).

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 локальной 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).

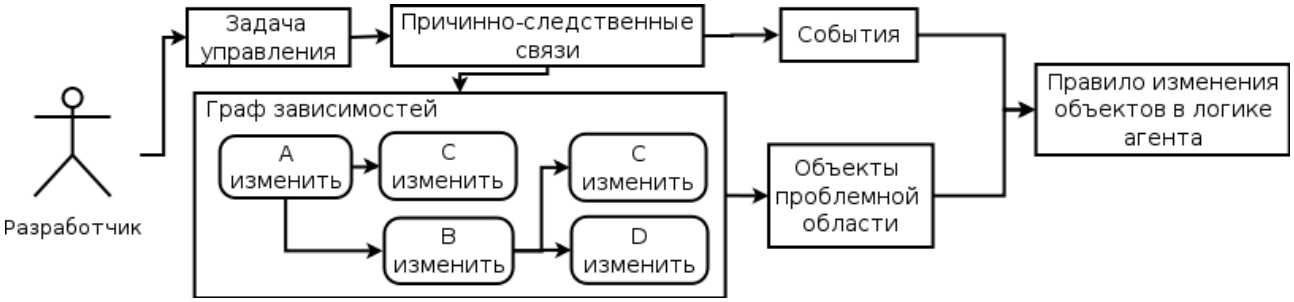

Рисунок 4. Построение правила обработки объектов проблемной области.

Сценарии использования ИП основаны на причинно-следственных связях проблемной области (рис. 4). Определяются события (причины) и граф зависимостей, описывающий цепочки обработки группы объектов проблемной области (следствия). Для заданного события e в логике агента программируется правило обработки q. Последнее реализует обработку локальной 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.

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, число людей в помещении, параметры климатического оборудования. Выполнение операций неделимым образом (транзакция) гарантирует семантическую целостность информационного содержимого ИП.

В третьей главе приведены разработанные автором механизмы программирования косвенного взаимодействия агентов, поддерживающие предложенный метод программирования на основе специализированных моделей взаимодействия агентов.

Механизм генерации программного кода онтологической библиотеки формирует для агента частную онтологическую модель на основе 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_{data} онтологической библиотеки для реализации работы с m в логике агента. При генерации анализируется иерархия классов, кардинальность свойств, тип свойств, ограничения на значения свойств.

Выбор объектов выполняется на уровне OWL классов и свойств (включать, не включать), позволяя уменьшить объем программного кода части l_{data}, емкостную трудоемкость в оценке (1). В 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. Отсутствие выбора объектов привело бы к дополнительному использованию ресурсов (память для хранения структур данных).

Механизм программирования взаимодействия на основе многоэлементной сессии определяет схему использования сессии и операций управления для управления сессией, сеансами, подписками и локальным хранилищем (рис. 5).

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*).

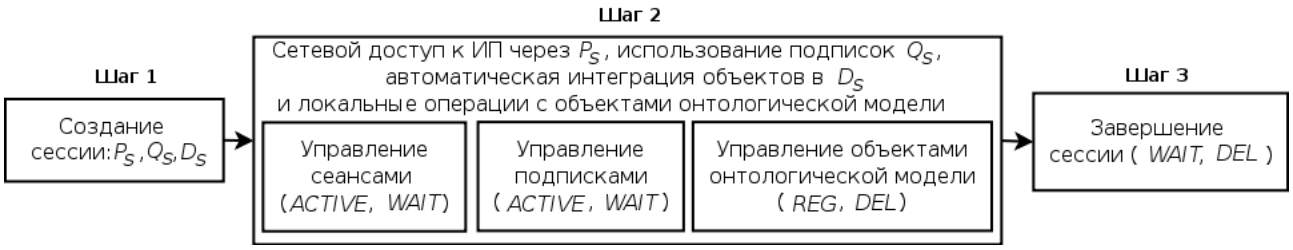

Рисунок 5. Схема использования сессии при программировании логики агента.

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 взаимодействия на основе 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_{src} в ИП: 1) подписка на свойства (изменение свойств индивидов) и 2) подписка на классы (появление/удаление индивидов). В первом случае объекты в O_{src} — наборы «индивид – список свойств», во втором — наборы классов. Допускается смешанный вариант с заданием в O_{src} различных наборов. Поддерживается синхронный и асинхронный способ работы операции подписки, позволяющий реализовать логику агента в виде набора параллельных вычислительных потоков. Данных видов подписки достаточно для взаимодействия, так как они позволяют реализовать отслеживание объектов, которое можно реализовать на уровне RDF-троек.

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_{src}) в набор Т-шаблонов (D_{sub}). Операция *UPDATE*: 1) интерпретирует RDF-тройки (D_{ind}), определяя URI индивидов, классов и свойств и 2) синхронизирует объекты (O_{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» таких параметры как температура, влажность (публикуются в ИП агентами сенсорного оборудования). Аналогично, подписка на класс «человек» позволяет отслеживать появление человека.

Механизм программирования взаимодействия на основе обработки локальной группы объектов реализует дополнительное свойство, устанавливаемое объекту онтологической модели (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. Обновление свойства означает наступление события, значением выступает само событие e. При наступлении события e агентом автоматически выполняются следующие шаги (рис. 6).

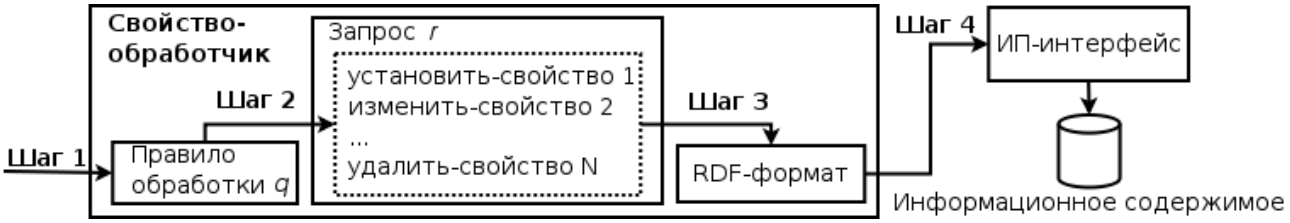

Рисунок 6. Использование свойства-обработчика для группы объектов.

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.

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 индиви-

да 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 известная платформа Smart-M3 для развертывания ИП и, разрабатываемый при участии автора, программный инструмент SmartSlog. Последний включает реализацию предложенных в работе механизмов программирования взаимодействия агентов в 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.

Инструмент 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, представленных в табл. 1. 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.

<table>
<tr><th></th><th>Windows</th><th>Windows Phone</th><th>На основе Linux</th><th>Mac OS</th><th>Android</th></tr>
<tr><td>ANSI C
(C KPI)</td><td>+</td><td>–</td><td>+</td><td>+</td><td>+ (взаимодействие ANSI C и Java кода)</td></tr>
<tr><td>C# (C KPI)</td><td>+ (адаптер)</td><td>+ (адаптер)</td><td colspan="2" rowspan="2">Mono (.NET Framework на базе свободного ПО)</td><td>–</td></tr>
<tr><td>C# (C# KPI)</td><td>+</td><td>+</td><td>–</td></tr>
</table>

Таблица 1. Платформы, поддерживаемые инструментом SmartSlog.

Уменьшение объема программного кода логики агента показано на рис. 7 в зависимости от действий, выполняемых при взаимодействии. 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 39%, 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. 2. 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 незначителен — хранение индивида и свойства требует 48 и 20 байт для структур данных (напр., для 10^5 индивидов с пятью свойствами у каждого нужно около 14 МБ). Исходные RDF-тройки не хранятся на стороне агента КР после их преобразования в индивиды и свойства.

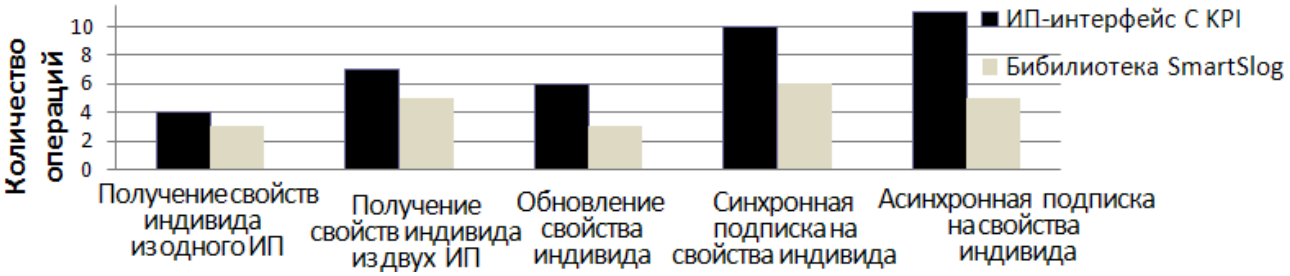

Рисунок 7. Количество операций для реализации действий при взаимодействии агента в ИП.

Практическая эффективность инструмента SmartSlog показана на примере разработки агентов КР для системы интеллектуального зала SmartRoom для ООО "Опти-Софт" и переносного варианта системы интеллектуально зала 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). На этих примерах показано, что возможностей библиотеки SmartSlog достаточно для программирования основных типов взаимодействия. При этом, уменьшается объем создаваемого вручную программного кода. 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 2. 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 решения базовых задач программирования взаимодействия в логике агента (см. рис. 2) без существенных потерь производительности и требуемых затрат памяти.

ЗАКЛЮЧЕНИЕ

Полученные в диссертационном исследовании результаты представляют собой решение актуальной задачи повышения эффективности разработки программных агентов интеллектуальных пространств. Внедрение результатов вносит вклад в развитие методов разработки программного обеспечения для интеллектуальных пространств. В ходе исследования получены следующие основные результаты:

1) Разработан метод программирования косвенного взаимодействия на основе специализированных моделей взаимодействия с использованием технологий Семантического веб. Метод позволяет повысить эффективность разработки программных агентов в условиях разнообразия аппаратно-программных платформ в IoT-средах.
2) Разработана модель взаимодействия агентов на основе многоэлементной сессии для поддержки параллельных сеансов сетевого доступа и интеграции информационных объектов из нескольких ИП. Модель позволяет организовать группы сеансов в сессии и управлять их сетевым взаимодействием с ИП.
3) Разработана модель взаимодействия агентов на основе операции подписки для автоматизации отслеживания агентом происходящих в ИП изменений на уровне объектов онтологической модели. Модель позволяет автоматически синхронизировать подписанные объекты с их изменениями в ИП.
4) Разработана модель взаимодействия агентов на основе локальной обработки группы объектов для программирования в логике агента операций над объектами онтологической модели как реакции на отслеживаемое событие. Модель позволяет агенту вносить множественные изменения в ИП в виде одной транзакцию, что уменьшает количество сетевых обращений агента к ИП и поддерживает семантическую целостность ИП.
5) Реализованы механизмы программирования косвенного взаимодействия и апробированы в программном инструменте SmartSlog платформы Smart-M3. Механизмы позволяют уменьшить объем создаваемого вручную разработчиком программного кода агента, поддерживают различные языки программирования, обеспечивают возможность программирования агентов для маломощных устройств.

Полученные результаты соответствуют п. 1 «Модели, методы и алгоритмы проектирования и анализа программ и программных систем, их эквивалентных пре-

образований, верификации и тестирования», п. 3 «Модели, методы, алгоритмы, языки и программные инструменты для организации взаимодействия программ и программных систем» паспорта специальности 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 системе 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. 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, No 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.