ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
ПЕТРОЗАВОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

На правах рукописи

Галов Иван Викторович

Модели проектирования программной инфраструктуры интеллектуального пространства для ресурсно-ограниченных вычислительных сред

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

Диссертация на соискание ученой степени
кандидата технических наук

Научный руководитель
к. ф.-м. н., доцент
Корзун Дмитрий Жоржевич

Петрозаводск – 2017

Оглавление

Введение 4

Глава 1. Особенности разработки интеллектуальных пространств 14

1.1. Интеллектуальные пространства с косвенным взаимодействием агентов на основе семантического представления общего информационного содержимого 14

1.2. Ресурсно-ограниченные среды Интернета вещей 20

1.3. Программная инфраструктура интеллектуального пространства 28

1.4. Управление сетевым доступом к информационному хранилищу . 33

1.5. Обеспечение устойчивости компонентов программной инфраструктуры к сбоям IoT-среды 40

1.6. Построение и доставка сервисов на основе взаимодействия программных агентов 42

1.7. Выводы 48

Глава 2. Проектирование программной инфраструктуры интеллектуального пространства 49

2.1. Метод разработки программной инфраструктуры 49

2.2. Модель управления сетевым доступом программных агентов к информационному хранилищу 54

2.3. Модель обеспечения устойчивости компонентов программной инфраструктуры к сбоям вычислительной среды 66

2.4. Выводы 72

Глава 3. Построение сервисов в интеллектуальном пространстве 74

3.1. Разработка агентов для совместного решения задачи на основе шаблонного подхода 74

3.2. Модель информационных уведомлений для программирования косвенного взаимодействия агентов 86
3.3. Программирование взаимодействия агентов на основе онтологического событийно-ориентированного описания взаимодействий . 97
3.4. Выводы 103

Глава 4. Экспериментальное исследование производительности 104
4.1. Трудоемкость обработки операций в программной реализации информационного хранилища CuteSIB 104
4.2. Время восстановления после сбоев на примере системы проведения мероприятий совместной деятельности 114
4.3. Время обработки информационных уведомлений в зависимости от количества параметров 117
4.4. Время обработки операций программной инфраструктурой в зависимости от числа агентов 121
4.5. Выводы 123

Заключение 124

Список сокращений и условных обозначений 126

Список литературы 128

Приложение А. Регистрация программ для ЭВМ 144

Приложение Б. Акты внедрения 146

Приложение В. Сводные характеристики комплекса программных средств 150

Введение

Актуальность темы. Развитие технологий Интернета вещей (Internet of Things, IoT) [26] привело к появлению нового класса вычислительных окружений — ресурсно-ограниченных IoT-сред, что позволяет реализовать новые цифровые решения для различных областей экономической и социальной деятельности человека [39, 53, 4]. Особенностью таких сред являются использование коммуникационных сетей с ограничениями (низкая скорость передачи данных) и вовлечение в работу периферийных устройств с ограниченными вычислительными ресурсами [28]. Возникает задача создания в такой среде интеллектуального пространства (ИП) как вычислительного окружения, обеспечивающего динамическое множество участников контекстно-зависимыми информационными сервисами. Участники представлены программными агентами на сетевых вычислительных устройствах (СВУ). Агенты взаимодействуют для совместного построения сервисов, используя доступные ресурсы IoT-среды. В работе рассматриваются ИП с косвенным взаимодействием агентов [10] через разделяемое информационное хранилище с семантическим представлением состояния IoT-среды и операциями сетевого доступа. При косвенном взаимодействии действия агента проявляются как изменения содержимого информационного хранилища, которые воспринимаются другими агентами для модификации своего поведения. Необходима программная инфраструктура, обеспечивающая организацию косвенного взаимодействия агентов для совместного построения информационных сервисов. Программная инфраструктура ИП разворачивается на имеющихся в IoT-среде СВУ в виде следующих компонент: а) информационное хранилище для организации косвенного взаимодействия; б) программные агенты для построения сервисов; в) системное программное обеспечение (СПО) для запуска исполнения и обеспечения работоспособности как самих агентов, так и информационного хранилища.

Такая организация косвенного взаимодействия является актуальной зада-

чей для создания ИП в ресурсно-ограниченных IoT-средах в силу следующих условий. Во-первых, известные программные реализации информационных хранилищ (например, Hydra, CoCaMAAL, OSGi SIB) ориентированы на специализированные среды, что затрудняет использование доступного аппаратно-сетевого разнообразия и не обеспечивает возможности развертывания программной инфраструктуры с учетом ограничений на ресурсы IoT-среды. Во-вторых, ресурсно-ограниченная IoT-среда содержит слабопроизводительные СВУ состав участников подвержен изменениям (многократные подключения и отключения различных СВУ) и работа беспроводных сетевых соединений нестабильна, что приводит к частым сбоям при построении сервисов и не обеспечивает возможности непрерывной работы сервиса (например, на беспроводном маршрутизаторе в режиме 24/7). В-третьих, при создании ИП требуется возможность интеграции с уже разработанными сервисами (внешние и из других ИП), без предопределения набора возможных комбинаций сервисов и используемых предметных областей. В указанных условиях существующие модели для организации косвенного взаимодействия агентов на уровне программной инфраструктуры (глобальное облако, общие модели классной доски и публикации/подписки) становятся неэффективными или неприменимы.

В диссертационной работе предлагаются новые модели для организации косвенного взаимодействия, на основе которых формируется новый метод разработки программной инфраструктуры ИП для указанных условий ресурсно-ограниченных IoT-сред.

Степень разработанности темы. Развитию методологических основ создания интеллектуальных пространств способствовали труды российских и зарубежных ученых: С. И. Баландина, С. Болдырева, В. И. Городецкого, А. М. Кашевника, Ю. Кильяндера, Д. Ж. Корзуна, Д. Кук, Я. Оливера, А. Л. Ронжина, А. В. Смирнова, Р. М. Юсупова, Т. Чинотти, Ю. Хонкколы, А. Д'Элиа и других. Вопросы разработки программного обеспечения для IoT-среды и организации вычислений на периферийных СВУ рассмотрены в работах Ч. Аг-

гарвала, М. А. Аксеновой, В. А. Александрова, Ф. Бономи, Б. Канга, Х. Кима, К. В. Кринкина, Д. И. Муромцева, О. Ю. Никифорова, М. Г. Пантелеева, М. Раззака, А. В. Рослякова, А. Н. Терехова, Д. Э. Трибушкова, А. Эспозито и других.

Целью работы являются обеспечение технической возможности и повышение эффективности построения информационных сервисов в интеллектуальном пространстве в условиях ресурсно-ограниченных IoT-сред за счет разработки моделей проектирования для организации косвенного взаимодействия программных агентов и формирования метода разработки программной инфраструктуры интеллектуального пространства.

Для достижения поставленной цели в работе решены следующие основные задачи.

1. Выполнить обзор моделей проектирования для организации косвенного взаимодействия программных агентов и методов разработки программных инфраструктур ИП.
2. Сформировать метод разработки программной инфраструктуры ИП для построения информационных сервисов на основе моделей для организации косвенного взаимодействия программных агентов в условиях ресурсно-ограниченных IoT-сред.
3. Разработать модель управления сетевым доступом программных агентов и обеспечения устойчивости программной инфраструктуры ИП к сбоям IoT-среды для организации косвенного взаимодействия агентов через информационное хранилище.
4. Разработать онтологическую модель информационных уведомлений для организации косвенного взаимодействия агентов в ИП на основе семантического описания требуемых вариантов взаимодействия при построении сервиса.
5. Провести экспериментальное исследование свойств программных инфра-

структур, разрабатываемых на основе предложенных метода разработки и моделей проектирования для организации косвенного взаимодействия программных агентов в ИП.

Объектом исследования является программная инфраструктура, обеспечивающая создание интеллектуального пространства в ресурсно-ограниченной IoT-среде для построения информационных сервисов на основе косвенного взаимодействия программных агентов.

Предметом исследования являются модели проектирования программного обеспечения для организации косвенного взаимодействия агентов в интеллектуальном пространстве.

Научная новизна работы. Разработаны три оригинальные модели проектирования ПО, формирующие в совокупности новый метод разработки программной инфраструктуры ИП для условий ресурсно-ограниченных IoT-сред. Учет этих условий позволяет создавать требуемое ИП в заданной ресурсно-ограниченной IoT-среде, что ранее было возможно лишь для частных случаев ИП и IoT-сред.

1. Предложен метод разработки программной инфраструктуры ИП на основе новых моделей организации косвенного взаимодействия программных агентов, отличающийся возможностью создания ИП с использованием широкого разнообразия слабопроизводительных СВУ в IoT-среде, при низкой производительности локальных сетевых коммуникаций и без привлечения множества дополнительных СВУ.

2. Предложена концептуальная модель управления сетевым доступом программных агентов к информационному хранилищу на основе модульной системной архитектуры, отличающаяся возможностью настройки информационного хранилища для выполнения на заданном СВУ IoT-среды за счет выбора поддерживаемых операций сетевого доступа и способа их параллельной обработки на СВУ с информационным хранилищем.

3. Предложена структурная модель обеспечения устойчивости компонентов программной инфраструктуры к сбоям IoT-среды на основе многоуровневой системы восстановления, отличающаяся возможностью использования имеющихся в IoT-среде ресурсов без привлечения множества дополнительных СВУ или программных агентов для восстановления на следующих уровнях: а) операции подписки; б) сетевых соединений между агентами и информационным хранилищем; в) вычислительных процессов работы агентов на СВУ; г) хранения данных с разделением семантической информации и объемных бинарных данных.

4. Предложена онтологическая модель информационных уведомлений на основе общей онтологии вариантов косвенного взаимодействия для заданного ИП, отличающаяся возможностью унифицированного событийно-ориентированного программирования индивидуального участия агентов во взаимодействии при совместном построении ими информационного сервиса, что позволяет упростить разработку и интеграцию в ИП информационных сервисов.

5. Реализован комплекс программных средств в соответствии с предложенными методом разработки программной инфраструктуры и моделями проектирования программного обеспечения для организации косвенного взаимодействия агентов, в составе: а) реализация информационного хранилища CuteSIB, отличающаяся возможностью создания и настройки ИП для существующего разнообразия ресурсно-ограниченных IoT-сред; б) реализация компонентов программной инфраструктуры на примере системы проведения мероприятий совместной деятельности, отличающаяся возможностями использования слабопроизводительных СВУ для построения опорных сервисов системы, восстановления после сбоев сетевых коммуникаций и расширения сервисов системы за счет интеграции сервисов из других ИП и сети Интернет; в) реализация ПО для проведения экспери-

ментального исследования по оценке возможностей применения предложенных моделей.

Практическая значимость работы. Предложенный метод позволяет разрабатывать программную инфраструктуру для создания требуемого ИП в заданной ресурсно-ограниченной IoT-среде с целью построения сервисов, реализующих новые цифровые решения прикладных задач для различных областей деятельности человека. Реализовано информационное хранилище CuteSIB как основное программное средство для создания ИП в разнообразных ресурсно-ограниченных IoT-средах. В качестве референтного прикладного случая исследована система проведения мероприятий совместной деятельности, реализующую автоматизированную поддержку проведения конференций, совещаний и других мероприятий. Для этого случая разработан вариант программной инфраструктуры, обеспечивающий построение опорных сервисов системы и служащий примером применения предложенных метода и моделей. Проведенное экспериментальное исследование предоставляет эмпирические оценки эффективности и границы возможности применения предложенных моделей для организации косвенного взаимодействия агентов в условиях ресурсно-ограниченных IoT-сред.

Методы исследования. Результаты выполненных в работе исследований и программных разработок основаны на методах системного и распределенного программирования, сервисно-ориентированных информационных систем, онтологического моделирования, а также на интернет-технологиях и технологиях Семантического Веба.

Положения, выносимые на защиту:

1. Метод разработки программной инфраструктуры интеллектуального пространства для построения информационных сервисов взаимодействующими агентами в условиях разнообразия и нестабильности ресурсно-ограниченных IoT-сред.

2. Концептуальная модель управления сетевым доступом программных агентов к информационному хранилищу для организации косвенного взаимодействия агентов на основе выбора поддерживаемых хранилищем операций доступа и настраиваемых способов параллельной обработки информации.
3. Структурная модель обеспечения устойчивости компонентов программной инфраструктуры к сбоям для поддержки взаимодействия агентов на основе многоуровневой системы восстановления после сбоев.
4. Онтологическая модель информационных уведомлений для организации косвенного взаимодействия агентов в ИП на основе семантического описания требуемых вариантов взаимодействия при построении сервиса.
5. Комплекс программных средств, разработанных в соответствии с предложенными методом разработки программной инфраструктуры и моделями для организации косвенного взаимодействия.

Степень достоверности и апробация. Достоверность научных положений, основных выводов и результатов диссертационной работы обеспечивается за счет анализа состояния исследований в данной области, согласованности теоретических свойств предложенных моделей проектирования программного обеспечения с результатами экспериментального исследования полученной программной реализации, а также апробацией основных положений диссертации в печатных трудах и докладах на научных конференциях. Новизна технических решений подтверждается полученными свидетельствами на программы для ЭВМ. Результаты работы представлялись на международных конференциях ассоциации открытых инноваций FRUCT в 2010–2016 гг., международных конференциях ruSMART (2011, 2012, 2014 гг.); международной конференции Information Technology Interfaces в 2012 г.; международных семинарах Annual International Workshop on Advances in Methods of Information and Communication Technology в 2013, 2014, 2015 гг.; всероссийской конференции «Современные

технологии в теории и практике программирования» в 2013, 2014, 2015 гг. Получены свидетельства о регистрации программы для ЭВМ № 2015615091 от 07.05.2015 «Программный комплекс опорных сервисов управления программой и материалами докладчиков для проведения мероприятий коллаборативной деятельности типа «конференция» в интеллектуальном зале SmartRoom в составе: Conference-service и Content-service» и № 2017615801 от 24.05.2017 «Программа, реализующая информационное хранилище CuteSIB, для организации косвенного взаимодействия агентов интеллектуального пространства в ресурсно-ограниченных вычислительных средах» (см. Приложение А).

Реализация результатов работы. Работа выполнена в рамках федеральной целевой программы «Исследования и разработки по приоритетным направлениям развития научно-технологического комплекса России на 2014–2020 годы» – соглашение № 14.574.21.0060 (RFMEFI57414X0060). Выполненное исследование поддержано: 1) Международной программой Karelia ENPI – грант KA179, 2012–2014 гг.; 2) Минобрнауки России – НИР № 2336 проектной части государственного задания № 2.2336.2014/К в сфере научной деятельности, 2014–2016 гг.; 3) РФФИ – научный проект № 14-07-00252, 2014–2016 гг.; 4) Минобрнауки России – НИР № 1481 базовой части государственного задания № 2014/154 в сфере научной деятельности, 2014–2016 гг.; 5) Программой стратегического развития ПетрГУ на 2012–2016 гг.

Результаты диссертационной работы внедрены в учебный процесс ПетрГУ, применяются в рабочем процессе ООО «Опти-Софт», используются для проведения международных конференций в Ассоциации Открытых Инноваций FRUCT (см. Приложение Б).

Публикации. По материалам диссертации опубликовано 19 печатных работ, включая 3 работы в российских журналах из списка ВАК и 10 работ в международных изданиях, индексируемых в реферативных базах Web of Science или Scopus.

Структура и объем диссертации. Диссертация объемом 150 машино-

писных страниц состоит из введения, 4 глав, заключения, списка использованной литературы (129 наименований) и 3 приложений. Текст диссертации содержит 39 рисунков и 13 таблиц.

Во введении обосновывается актуальность работы, формулируются ее цель и решаемые задачи, приводятся положения, выносимые на защиту, научная новизна и практическая значимость работы.

В первой главе раскрывается суть основных понятий интеллектуальных пространств с косвенным взаимодействием агентов и приводится обзор существующих исследований в данной области. Формулируются основные проблемы и решаемые задачи.

Основой ИП является его программная инфраструктура. Программная инфраструктура обеспечивает функционирование ИП, целью которого является построение и доставка информационных сервисов для конечного пользователя. Программная инфраструктура состоит из двух трех компонентов: 1) информационного хранилища, 2) программных агентов, 3) сервисного системного ПО (СПО) для обеспечения работоспособности хранилища и агентов. Информационное хранилище обеспечивает косвенное взаимодействие программных агентов через разделяемое содержимое хранилища I. С помощью операций доступа агенты могут выполнять манипуляции с содержимым информационного хранилища. За счет получения/изменения информации в хранилище агенты осуществляют взаимодействие друг с другом. На основе такого взаимодействия выполняется построение информационного сервиса.

Во второй главе предлагается метод разработки программной инфраструктуры ИП. Метод основывается на трех моделях: концептуальной модели управления сетевым доступом программных агентов к информационному хранилищу, структурной модели обеспечения устойчивости компонентов программной инфраструктуры к сбоям, онтологической модели информационных уведомлений. Концептуальная модель управления сетевым доступом программных агентов к информационному хранилищу предназначена для выполнения на-

стройки информационного хранилища под условия заданной IoT-среды. Структурная модель обеспечения устойчивости компонентов программной инфраструктуры к сбоям предназначена для построения многоуровневой системы восстановления работоспособности информационного хранилища и агентов после сбоев с использованием СПО в IoT-среде. Онтологическая модель информационных уведомлений предназначена для унификации программирования участия каждого агента в построении информационного сервиса с представлением взаимодействия агентов непосредственно в I.

В третьей главе рассматривается построение сервисов в ИП. Предлагаются шаблоны взаимодействия на основе которых можно организовать взаимодействие программных агентов для построения информационного сервиса. Описывается онтологическая информационных уведомлений с использованием семантического описания возможных событий и взаимодействий. За счет онтологического описания всех вариантов взаимодействия между агентами модель позволяет организовать событийно-ориентированные и инициирующие (по запросу) взаимодействия.

В четвертой главе проводится экспериментальное исследование предложенных моделей. Модели исследуются на примере реализации информационного хранилища CuteSIB и программная инфраструктура для построения опорных сервисов системы проведения мероприятий совместной деятельности. На основе экспериментального исследования с применением экспериментального стенда и методов имитационного моделирования получены оценки возможностей применения предложенных моделей для организации косвенного взаимодействия в условиях разнообразия, динамики и нестабильности ресурсно-ограниченных IoT-сред.

Глава 1

Особенности разработки интеллектуальных пространств

В главе приводится общая характеристика решаемых в диссертационной работе научных задач. Текущее развитие ИКТ обеспечивает технические возможности для разработки новых классов сервисно-ориентированных информационных систем — цифровые окружения совместной деятельности человека, индустриальный интернет, кибер-медицинские системы и ряд других. Подобные системы могут быть реализованы в виде интеллектуального пространства с косвенным взаимодействием программных агентов над семантическим представлением содержимого общего информационного хранилища. Для создания интеллектуального пространства целесообразно использовать ресурсно-ограниченные IoT-среды, чтобы задействовать вычислительные возможности имеющегося разнообразия СВУ. Рассматривается задача разработки программной инфраструктуры интеллектуального пространства для развертывания в такой среде. Показано, что программная инфраструктура должна обеспечивать взаимодействие агентов при следующих условиях: а) аппаратно-сетевое разнообразие и ограниченность вычислительных ресурсов участвующих СВУ; б) динамика состава участвующих СВУ и нестабильность сетевых соединений при совместном построении и доставке сервисов агентами; в) интероперабельность построения сервисов агентами без предопределения набора возможных комбинаций сервисов и используемых предметных областей.

1.1. Интеллектуальные пространства с косвенным взаимодействием агентов на основе семантического представления общего информационного содержимого

Развитие ИКТ обеспечило технические возможности для разработки широкого круга сервисно-ориентированных информационных систем. В частности,

ведутся активные исследования и разработки в таких областях как:

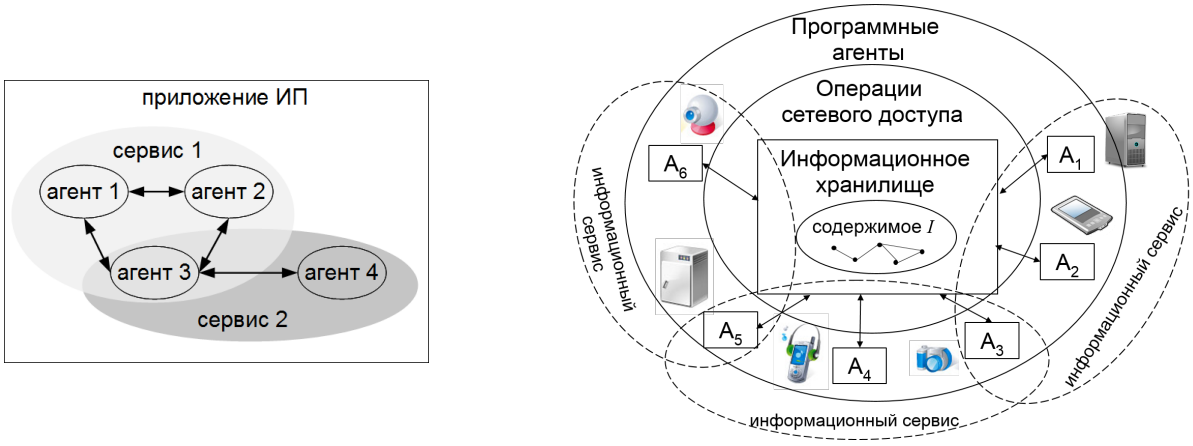
- цифровые окружения мобильного здравоохранения и поддержки здорового образа жизни, выполняющие построение персонализированных информационных сервисов на основе данных мониторинга множеством сенсоров, ассоциированных с человеком, см. напр. [116, 61, 104];

- цифровые окружения «промышленного (индустриального) интернета», выполняющие построение сервисов в виде контекстных рекомендаций при планировании и управлении технологическими процессами на производстве, см. напр. [35, 12, 38, 76, 73];

- цифровые окружения совместной деятельности человека, выполняющие построение сервисов для информационного обеспечения множества участников во время проведения мероприятия, см. напр. [32, 88, 129, 95].

Появление таких классов сервисно-ориентированных информационных систем соответствует общей тенденции «интеллектуализации» современных цифровых услуг в экономической, социальной, повседневной и других сферах деятельности человека. В частности, актуальность и высокое научное и народохозяйственное значение исследований и разработок в этой области подтверждаются направлениями, указанными в Стратегии научно-технологического развития Российской Федерации до 2035 года (утверждена Указом Президента РФ от 01.12.2016 N 642).

Известно, что подобные системы могут быть реализованы в виде интеллектуального пространства (ИП) с косвенным взаимодействием программных агентов над семантическим представлением содержимого общего информационного хранилища [3, 9, 14, 15, 16, 19, 23, 29, 37, 39, 41, 44, 52, 87]. В общем случае, под интеллектуальным пространством будем в работе понимать цифровую вычислительно-информационную среду, которая способна обеспечивать находящееся в ней динамическое множество программных агентов контекстно-



а) б)

Рисунок 1.1. Сервисы программного приложения ИП

зависимыми информационными сервисами тогда, когда они требуются и, по возможности, упреждающим (проактивным) образом [9, 44, 87]. Информационный сервис определяется как цифровая услуга, обеспечивающая поиск и предоставление пользователю (человек или СВУ) информации для последующей интерпретации при решении возникшей у пользователя задачи [44, 53, 16].

Программное приложение ИП реализует построение и доставку сервисов, как правило, для решения задач из определенной предметной области. Приложение разрабатывается как подсистема взаимодействующих агентов [16]. На рисунке 1.1а представлена схема, соотносящая понятия приложения ИП, информационных сервисов и агентов. В одном ИП может одновременно выполняться несколько приложений. Один и тот же программный агент может участвовать в построении нескольких сервисов и входить в состав нескольких приложений.

В настоящее время нет общепринятой архитектуры ИП (рисунок 1.1б). В таблице 1.1 представлены характеристические свойства ИП и общие подходы к созданию ИП. Вычислительная среда предоставляет аппаратную основу для создания ИП [87, 68]. По своей зоне действия вычислительная среда может быть глобально распределенной, локальной или локализованной. В случае глобально распределенной среды ИП распределяется по разным местоположениям (посредством сети Интернет), в каждом из которых может быть предо-

Таблица 1.1. Подходы к реализации интеллектуальных пространств

Свойство	**Подход/Модель**		
Организация информационного содержимого	Индивидуальные хранилища участников	Общее информационное хранилище	
Сетевое взаимодействие	Обмен сообщениями	Публикация/ подписка	Классная доска
	Прямое	Косвенное	
Вычислительная среда	Глобально распределенная	Локализованная	Локальная

ставлен информационный сервис. В случае локальной вычислительной среды ИП создается в ограниченном физическом пространстве (напр.: здание, отдельная комната) и является закрытой системой. Локализованная среда расширяет локальную среду в ограниченное физическое пространство за счет доступа к внешним информационным сервисам и ресурсам в сети Интернет.

По типу сетевого взаимодействия агентов различают ИП с прямым и косвенным взаимодействием [53]. При прямом взаимодействии каждый агент взаимодействует с другим путем непосредственного обмена сообщениями. Оба агента должны знать адрес доступа к другому агенту. Такое ИП, например, можно создавать на основе оверлейных P2P сетей [9]. В ИП с косвенным взаимодействием агенты взаимодействуют друг с другом через посредника — информационного брокера, управляющего общим информационным хранилищем [44, 115, 77]. Агентам для взаимодействия уже не требуется знать адрес доступа друг к другу, достаточно знать адрес доступа к информационному хранилищу.

Информационное хранилище содержит информацию, используемую агентами при взаимодействии. В случае прямого взаимодействия каждый агент имеет индивидуальное информационное хранилище, в котором он накапливает необходимые для него знания. Для организации косвенного взаимодействия между программными агентами в ИП используется общее информационное хранилище, через которое агенты могут обмениваться информацией. При косвенном взаимодействии не требуется установка попарных соединений между участ-

вующими СВУ. При этом, в современных вычислительных средах такие СВУ характеризуются не только большим числом, но и неоднородностью состава и имеющихся аппаратно-сетевых возможностей. В среде развертывается общее информационное хранилище, используемое агентами как сервер совместного потребления и производства информации.

В силу существенного роста числа СВУ, с одной стороны, и необходимости использования ресурсов глобальной сети Интернет, с другой стороны, особое значение приобретают локализованные ИП с косвенным взаимодействием агентов на основе общего информационного хранилища [52, 44, 94]. Информационное хранилище выступает не традиционной базой данных с предопределенной структурой, а связующим информационным центром (от англ. information hub), задача которого обеспечивать семантику происходящих процессов [90].

В диссертационной работе рассматривается класс локализованных ИП, характеризуемый следующими особенностями:

1. косвенное взаимодействие программных агентов с ограниченной автономностью на основе совместного накопления и обработки информации;

2. семантическое представление содержимого информационного хранилища для обеспечения ее связывания и интероперабельного использования;

3. организация взаимодействия на основе операций сетевого доступа агентов к информационному хранилищу.

Программные агенты с ограниченной автономностью [46] также называются процессорами знаний (от англ. knowledge processor — KP). Ограничение автономности агента проявляется как: а) ограничена возможность принятия самостоятельных решений, б) пользователь или администратор ИП имеет возможность влиять на участие агента во взаимодействии.

Для косвенного взаимодействия используются известные модели «классной доски» [54] и «публикации/подписки» [62]. Примером использования моде-

ли «классной доски» являются системы вычислений на пространствах кортежей [106]. Для семантического представления содержимого информационного хранилища используются модель данных RDF и онтологии, определяя возможности связывания и интероперабельного использования информации для взаимодействия агентов [29]. При использовании модели «публикации/подписки» агент может подписаться на часть содержимого информационного хранилища, получая затем уведомления о происходящих изменениях.

Многоагентные системы публикации/подписки и вычислений на пространствах кортежей позволяют выполнять построение информационных сервисов с использованием технологий Семантического веб [52]. Для построения сервисов агенты взаимодействуют друг с другом, совместно обрабатывая общую информацию [116]. Модель RDF используется для единообразного представления содержимого общего информационного хранилища [84]. Данные описываются с помощью RDF-троек, каждая имеет следующую структуру: «субъект–предикат–объект». Субъект и объект соответствуют понятиям в предметной области (человек, машина, дом и т.п.), а предикат (также называемый свойством) определяет отношение между ними (владелец, друг и т.п.). Имена для субъектов, предикатов и объектов определяются унифицированным идентификатором ресурсов (Uniform Resource Identifier, URI). Объект также может являться литералом (строкой), чтобы определять конкретные данные (напр., имя, возраст). Множество хранимых RDF-троек образует ориентированный граф, в котором узлами являются субъекты и объекты, а дугами — предикаты.

Модель RDF не определяет предметно-ориентированную структуру данных и не позволяет описывать семантические отношения между объектами предметной области. Для такого описания используются онтологии, см. напр. [74, 31, 30]. В различных предметных областях одни и те же понятия могут быть представлены разными терминами. Механизм онтологий в этих случаях позволяет формировать иерархические взаимосвязи между объектами, обобщать и совместно использовать глобальные сведения. Используется язык веб онтологий

(web ontology language, OWL) [49] для описания фактических данных (ресурсов) на основе структуры онтологических классов (концепций), отношений и ограничений в виде индивидов (экземпляры классов), их свойств данных (атрибуты индивида) и семантических связей (объектные свойства) между индивидами. Теоретической основой служат математические формализмы, называемые дескрипционными (описательными) логиками [43]. Формальная семантика OWL описывает, как получить логические выводы на основе онтологий, т.е. получить факты, которые не представлены буквально, а следуют из семантики. В языке OWL используется модель данных «объект–свойство», основанная на словаре RDF Schema и позволяющая определять разнообразные ограничения при моделировании предметной области [128]. Представление на основе OWL онтологии может быть автоматически преобразовано в модель RDF [17].

Таким образом, рассматриваемый класс ИП предназначен для разработки востребованных сервисно-ориентированных информационных систем для широкого круга предметных областей и возникающих там задач. Система разворачивается в вычислительной среде в виде интеллектуального пространства с косвенным взаимодействием программных агентов над семантическим представлением содержимого общего информационного хранилища. Применяемые технологии (в первую очередь, Семантического веб) ориентированы на использование традиционных Интернет-компьютеров (серверы, настольные, мобильные). К настоящему времени слабо исследованы возможности использования других СВУ (периферийных и промежуточных) для реализации особенностей рассматриваемого класса ИП. Исследование таких возможностей необходимо для развития технологий туманных вычислений (от англ. fog computing) [33, 51, 57].

1.2. Ресурсно-ограниченные среды Интернета вещей

Широкое распространение получают сейчас среды Интернета вещей (IoT-среды), см. напр. [40, 1, 36, 23, 2]. Такой среде обычно соответствует некоторое

ограниченное пространство (комната, дом, офис), оснащенная множеством СВУ (сенсоры, актуаторы, встроенные ЭВМ, персональные мобильные устройства и т.п.). Все СВУ связаны в локальную сеть, которая также имеет доступ в глобальную сеть Интернет. Каждое устройство IoT-среды является автономным участником и называется «умным объектом» в терминах Интернета вещей или агентом (с ограниченной автономностью) в терминах многоагентных систем. В случае IoT-среды ключевым фактором являются беспроводные коммуникации и, как следствие, возможности по динамике участия (вход/выход СВУ).

Современным направлением в IoT-технологиях являются «туманные вычисления» (от англ. fog computing) в которых вычисления переносятся (частично) с глобального облака в локализованные среды (периферия сети Интернет) с использованием маршрутизаторов или шлюзов [51, 57]. Большинство исследований в этой области посвящены сбору и передаче информации с устройств на шлюзы, дальнейшей ее агрегации и отправке в облако. С помощью концепции ИП возможно вовлечение имеющихся в среде устройств в совместные вычисления для построения и предоставления информационных сервисов. Отдельно выделяются ресурсно-ограниченные IoT-среды [111], особенностью которых является не столько большое количество участвующих СВУ, сколько их разнородность и разнообразие способов коммуникации между ними [51]. Ресурсная-ограниченность таких сред заключается в использовании коммуникационных сетей с ограничениями (низкая скорость передачи данных) и СВУ с ограниченными вычислительными ресурсами [28].

В качестве примера приложения ИП, развертываемого в ресурсно-ограниченной IoT-среды, рассмотрим переносной вариант системы проведения мероприятий совместной деятельности для развертывания в помещении [95, 32].

Пример 1.1 («Система проведения мероприятий совместной деятельности»). Предназначена для информационного обеспечения при проведении таких мероприятий как конференции, собрания, лекции (рисунок 1.2). Для режима конференции выделены программные агенты, реализующие опорные сервисы управ-

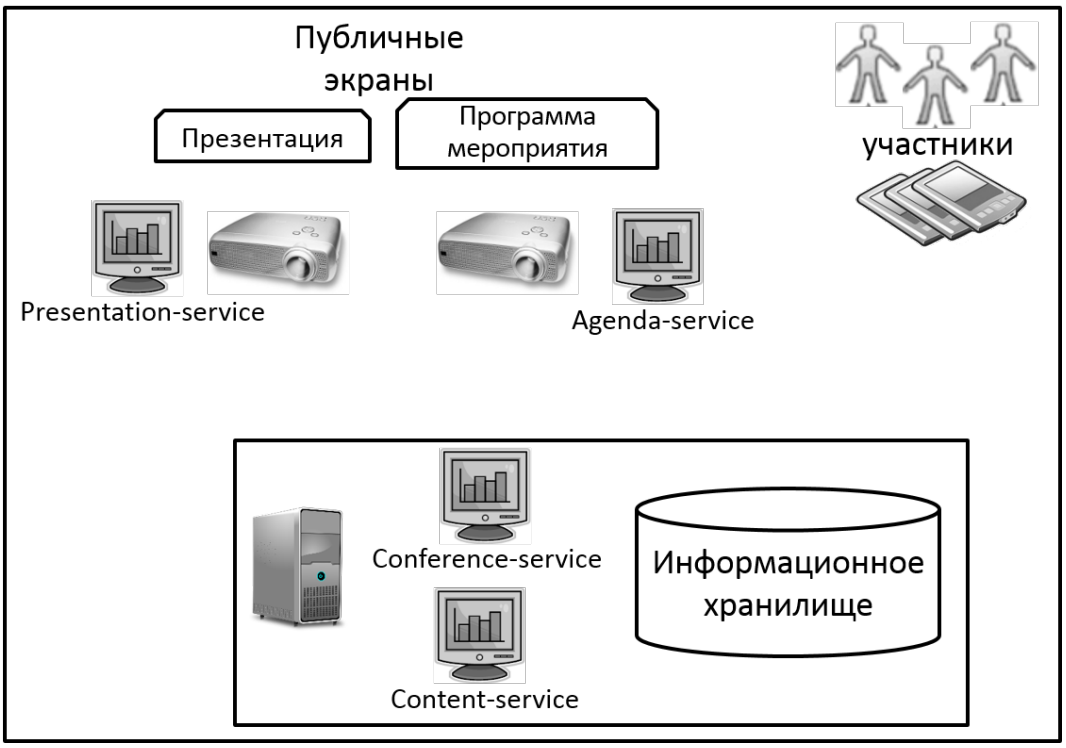


Рисунок 1.2. Архитектура системы проведения мероприятий совместной деятельности

ления конференцией (Conference-service), программой мероприятия (Agenda-service), презентациями докладчиков (Presentation-service) и содержимым (Content-service). Вычислительная среда системы локализована в помещении (напр., конференц-зал), оснащенном вычислительной и медиа аппаратурой. Может отсутствовать традиционный персональный компьютер и для запуска основных агентов используются другие имеющиеся СВУ (маршрутизатор, проектор и т.п.) В помещении могут быть установлены два проектора и экрана: один отображает слайды выступающего, другой — программу мероприятия. Множество участников конференции используют персональные мобильные СВУ (смартфоны, планшеты) с запущенным на них клиентским агентом для участия в конференции.

Для создания ИП из рассматриваемого класса разработаны различные платформы. Некоторые используют разделяемое содержимое, представленное моделью RDF, для обмена данными между агентами. Платформа JTangPS [114, 110] применяет модель публикации/подписки, где отслеживаемые события представлены в виде графа RDF. Используются онтологии и механизм вывода для

поиска и интерпретации хранимой информации. Подписки осуществляются с помощью сопоставления шаблонов графах RDF. В работе [123] предлагается онтолого-ориентированный подход для создания системы на основе модели публикации/подписки, где используется собственный язык для подписок и онтологическое представление событий (событие от публикующего агента преобразуется в RDF-тройки и доставляется подписчику). Выделяется два вида систем по структуре событий: основанные на отображениях (множество пар атрибут-значение) и основанные на XML. В работе [124] авторы предлагают инфраструктуру системы для создания ИП как некоторого "семантического пространства знаний", в котором информация представляется с помощью OWL-онтологий. Для запроса знаний используется специальный язык RDF data query language (RDQL). В то же время, перечисленные выше платформы не предназначены для развертывания ИП в ресурсно-ограниченных IoT-средах, появляющихся сейчас массово на периферии глобальной сети Интернет. Эти платформы ориентируются на традиционные персональные ЭВМ (настольные ЭВМ, ноутбуки), не поддерживая имеющегося разнообразия СВУ в таких IoT-средах.

В работе [111] выполнен обзор множества платформ для создания ИП в IoT-среде. Выделяются такие классы платформ, как событийно-ориентированные, сервисно-ориентированные, виртуальные, многоагентные, основанные на пространствах кортежей. Платформа Hydra [61] позволяет создавать ИП на основе сервисно-ориентированной архитектуры. Она состоит из модулей для управления сервисами, событиями, устройствами, информационным хранилищем, контекстом и безопасностью. Среди этих компонентов выделяются семантический, сервисный, сетевой уровни и уровень безопасности. Платформа поддерживает динамическую реконфигурацию и самонастройку. Платформа A3-TAG [47] создает ИП на основе разделяемого пространства кортежей. Множество участников обмениваются информацией через это пространство. Для возможности работы большого количества устройств в платформе осуществляется объединение устройств в группы и в дальнейшем взаимодействие осуществ-

ляется между группами устройств. Платформа KSpot$^+$ [42] позволяет создавать распределенную датацентрированную архитектуру сервиса для сенсорных сетей. Платформа поддерживает расширенные семантические запросы, а также обеспечивает расширяемость и устойчивость к сбоям. Рассмотренные платформы предназначены для конкретных IoT-сред — беспроводных сенсорных сетей (Wireless sensor network, WSN), что не позволяет создавать ИП для существующего широкого разнообразия IoT-сред.

Другой класс формируют платформы, которые поддерживают большее разнообразие СВУ, но имеют ограничение по сфере своей применимости. В работе [65] рассматривается платформа CoCaMAAL, представляющая класс ИП для помощи в повседневной жизни, которая основана на облачных технологиях. Собирая информацию с множества сенсоров и сохраняя ее в облаке, платформа позволяет организовать доступ к этой информации различным приложениям для построения и предоставления информационных сервисов. Платформа meSchup [96] предлагает создание ИП в IoT-среде на периферии сети Интернет. Взаимодействие организуется через сервер-посредник, который запущен на специализированном устройстве, позволяющем подключать к себе различные мобильные устройства. Для подключения мобильного устройства необходима установка на него специального клиента платформы meSchup. Возможности использования ограничены узким классом устройств, поддерживающих такого клиента.

Распространение получили платформы, предназначенные для конкретных предметных областей (например, умный дом, умный офис, автомобиль). В работе [104] рассматривается медицинская система мобильного мониторинга для отслеживания состояния здоровья пациента с помощью датчиков и с возможностью привлечения других (носимых) медицинских СВУ. Взаимодействие между СВУ осуществляется на основе развертывания IoT-шлюза, который реализует медицинское ИП и сохраняет данные в облаке. Применимость таких платформ существенно ограничивается классами СВУ, применяемыми в конкретной при-

кладной области.

Перспективной для создания ИП в разнообразных ресурсно-ограниченных IoT-средах является известная архитектура M3, реализованная, в частности, в виде технологической платформы Smart-M3 [77]. Платформа позволяет создать ИП с косвенным взаимодействием программных агентов на основе модели публикации/подписки. Разделяемая между агентами информация представляется с помощью модели RDF и содержится в информационном хранилище, управляемом брокером семантической информации (semantic information broker, SIB). Существует несколько программных реализаций брокеров, развивающих исходную реализацию брокера платформы Smart-M3. Брокер RedSIB [103] ориентирован на запуск на персональных ЭВМ и предлагает улучшенный алгоритм обработки подписки. В работе представлен брокер OSGi SIB [59], который ориентируется на запуск в IoT-средах с поддержкой языка Java. На основе платформы OSGi достигается модульность и расширяемость брокера, что позволяет добавлять в него новый функциональные возможности. Реализация брокера PySIB [120] представляет «легковесный» модульный брокер, предназначенный для запуска на ресурсно-ограниченных СВУ с поддержкой языка Python. В то же время, текущие реализации этих брокеров имеют статус исследовательского прототипа, не акцентируя, в частности, такие практически важные свойства как настройка под имеющиеся ресурсы IoT-среды, обеспечение устойчивости к сбоям, возможности унифицированного программирования агентов.

Разнообразие существующих платформ и систем для развертывания ИП связано с аппаратно-сетевым разнообразием самих IoT-сред [111, 51, 127]. Многие существующие реализации информационных хранилищ предназначены только для узкого класса IoT-сред. Таким образом, в условиях разнообразия IoT-сред возникает проблема, что информационный сервис из одного ИП не может быть перенесен или интегрирован в ИП другой IoT-среды, т.к. в ней не может быть развернуто ИП на основе заданного информационного хранилища.

Другой проблемой является нестабильность IoT-сред и динамика участия

СВУ [57, 73]. Нестабильность проявляется в периодических разрывах сетевого соединения между информационным хранилищем и программными агентами. Число участников (в виде СВУ) динамически меняется: мобильные СВУ могут подключаться (вход в среду) и отключаться (выход из среды). Таким образом, в этих условиях нестабильности и динамики IoT-среды возникает проблема разнообразных сбоев, связанных как с информационным хранилищем, так и с программными агентами. Наличие сбоев препятствует построению и доставке сервиса, что приводит к необходимости выполнения восстановления после сбоев при организации взаимодействия агентов.

Разнообразие IoT-сред и существующих реализаций информационных хранилищ привело к появлению разнообразных способов программирования взаимодействия агентов, необходимого для совместного построения информационного сервиса. Способы программирования взаимодействия могут различаться и при использовании одного информационного хранилища вследствие гибкости онтолого-ориентированного описания содержимого информационного хранилища. Таким образом, в условиях разнообразия способов программирования взаимодействия возникают проблемы интеграции информационных сервисов или переноса сервиса из одного ИП в другое, что приводит к необходимости реализации разных способов взаимодействия между агентами для поддержки интеграции или переноса [109].

В примере 1.1 «Система проведения мероприятий совместной деятельности» (с. 21) перечисленные проблемы проявляются следующим образом. В условиях разнообразия IoT-сред возможен широкий спектр СВУ для запуска агентов системы. Информационное хранилище может быть запущено как на персональной ЭВМ, так и на встроенных СВУ типа маршрутизатор [24]. Участники конференции, использующие персональные мобильные устройства, повышают динамику среды из-за регулярных подключений и отключений. При такой работе участников также часто происходят сбои сетевых соединений.

Перечисленные выше проблемы, связанные с условиями разнообразия, неста-

Таблица 1.2. Платформы для создания ИП

Класс платформ	Особенности
Платформы не поддерживающие IoT-среды: JTangPS [114, 110], подход онтолого-ориентированной системы публикации/подписки [123], подход семантического пространства [124]	Ориентация на персональные ЭВМ, не поддерживаются различные СВУ, в том числе ресурсно-ограниченные. Нет встроенных механизмов обеспечения устойчивости. к сбоям.
Платформы ориентированные на сенсорные сети: Hydra [61], A3-TAG [47], KSpot$^+$ [42]	Поддерживаются только сенсорные сети WSN. Специализированные механизмы устойчивости к сбоям присутствуют только у KSpot$^+$.
Проблемно-ориентированные платформы: CoCaMAAL [65], meSchup [96], подход IoT-шлюза [104]	Ориентация на специализированные устройства в рамках локализованных IoT-сред типа умный дом. Нет встроенных механизмов обеспечения устойчивости к сбоям.
M3-архитектура: Smart-M3 SIB [77], RedSIB [103], PySIB [120], OSGi SIB [59]	Поддержка широкого круга СВУ, но сама платформа может быть развернута на конкретном типе устройств. Разработки имеют статус исследовательского прототипа, поэтому вопросам обеспечения устойчивости к сбоям отдан меньший приоритет.

бильности и динамикой IoT-сред не в полной мере решаются в существующих платформах (таблица 1.2). Решение этих проблем возможно на уровне программной инфраструктуры ИП, которая включает в себя информационное хранилище и программных агентов [7, 71]. Таким образом, необходима разработка такой инфраструктуры ИП, которая будет поддерживать разнообразные IoT-среды, а также обеспечит устойчивость к сбоям в условиях разнообразия, нестабильности и динамики сред.

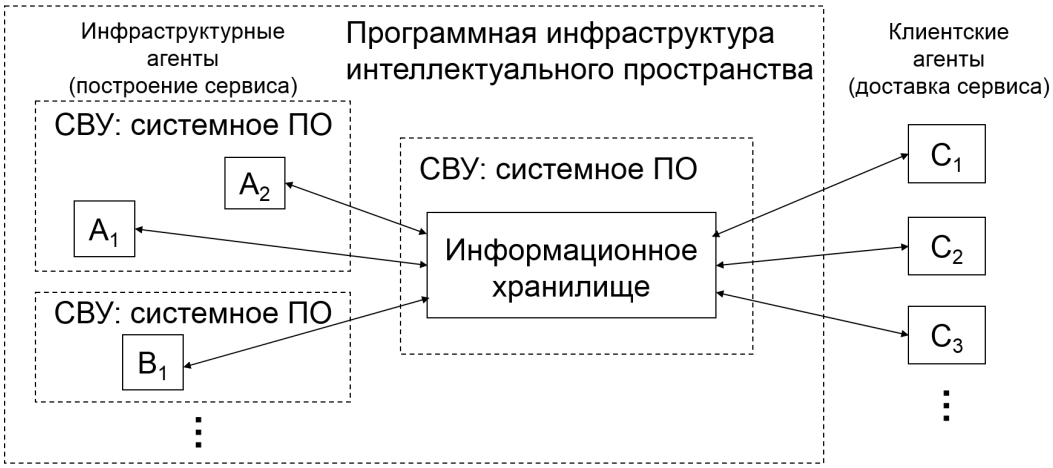


Рисунок 1.3. Инфраструктура интеллектуального пространства

1.3. Программная инфраструктура интеллектуального пространства

В общем случае, под инфраструктурой понимаются средства, которые делают возможным функционирование информационного сервиса [27]. Она включает в себя программно-аппаратное обеспечение и вычислительную среду. В состав программной инфраструктуры входят программные средства, обеспечивающие функционирование сервиса. В случае рассматриваемого класса ИП программная инфраструктура включает в себя следующие компоненты: а) информационное хранилище для обеспечения косвенного взаимодействия, б) инфраструктурные агенты для совместного построения сервисов, в) системное программное обеспечение (СПО) для запуска и обеспечения функционирования хранилища и агентов (рисунок 1.3). Программные агенты, которые непосредственно отвечают за построение и доставку информационных сервисов называются инфраструктурными [7].

Процесс установки, настройки и запуска основных компонентов инфраструктуры на устройствах вычислительной среды называется развертыванием. Существуют разнообразные классы СВУ: персональные (телефоны, планшеты), одноплатные компьютеры (бытовые устройства и устройства для контроля и управления), настольные ЭВМ, серверные ЭВМ, маршрутизаторы. Возможность развертывания инфраструктуры зависит от того, поддерживают ли

Таблица 1.3. Варианты развертывания инфраструктурных агентов

Вариант развертывания	**Свойства**
1. Рядом с информационным хранилищем	Быстрый доступ к разделяемой информации. Простота установки и обслуживания. Пример: базовые агенты сервиса.
2. На устройстве	Эффективное взаимодействие с устройством, необходимым для сервиса. Пример: агенты для сенсоров, камеры, климат-контроль.
3. На сервере	Ресурсоемкая обработка данных. Пример: посредник между внешними источниками данных и сервисами.

компоненты инфраструктуры СВУ среды.

Выделим три варианта развертывания инфраструктурных агентов (таблица 1.3). В варианте 1 запуск агента происходит на том же СВУ, где запущено информационное хранилище. Если хранилище запущено на серверной ЭВМ, то агент может работать постоянно, выполнять ресурсоемкие операции и получать доступ к информационному хранилищу с небольшой сетевой задержкой.

В варианте 2 агент запускается на отдельном СВУ, который непосредственно взаимодействует со специализированной аппаратурой, используя ее ресурсы для предоставления сервиса (проектор, интерактивная доска, сенсор, камера, микрофон и т.п.). Такие агенты могут служить посредниками для маломощных устройств, на которых нельзя запустить полноценных агентов (напр., сенсоры).

Вариант 3 заключается в запуске агента на отдельной серверной ЭВМ (отличной от СВУ с информационным хранилищем). Вариант предназначен для сервисов, построение которых требует ресурсоемких вычислений. Частный случай — это агенты-посредники, которые устанавливают связь между внешними источниками данных и ИП [89]. Здесь преобразование и синхронизация данных часто являются вычислительно ресурсоемкими. Развертывание на серверной

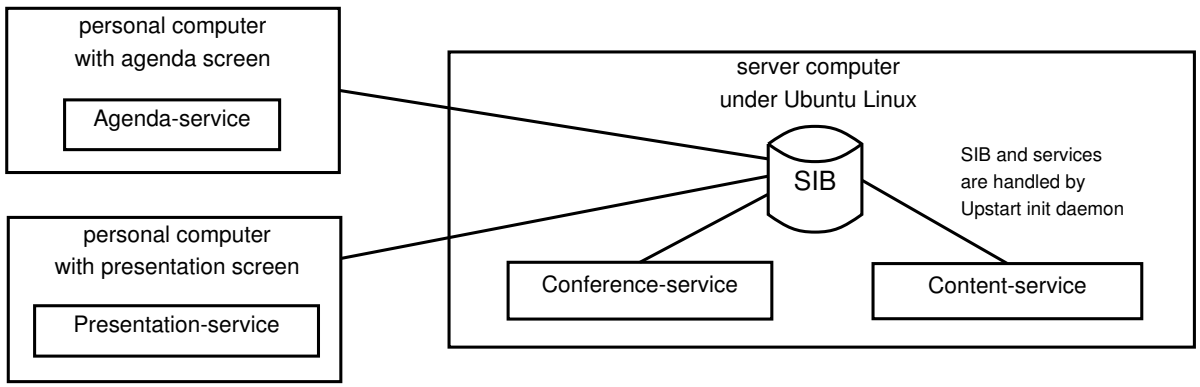


Рисунок 1.4. Программная инфраструктура системы проведения мероприятий совместной деятельности

ЭВМ подразумевает широкий диапазон используемых компьютеров (имеющихся в заданной IoT-среде): от локальных и корпоративных серверов до облачных систем в Интернет.

На рисунке 1.4 приведен вариант развертывания инфраструктуры на примере системы проведения мероприятий совместной деятельности [88, 95].

Пример 1.2 («Система проведения мероприятий совместной деятельности: развертывание инфраструктуры»). Рассмотрим вариант развертывания инфраструктуры на основе примера 1.1 «Система проведения мероприятий совместной деятельности» (с. 21). Информационное хранилище (брокер SIB) запускается на серверной ЭВМ. Агенты для сервисов Conference-service и Content-service запускаются на ЭВМ вместе с информационным хранилищем, чтобы уменьшить время сетевой передачи информации между агентами и ИП (вариант развертывания 1 в таблице 1.3). Агенты Agenda-service и Presentation-service запускаются на компьютерах, соединенных с проекторами (вариант развертывания 2).

Вопросы построения инфраструктуры для приложений ИП рассматриваются, например, в [126, 124, 113, 56]. Примером платформы для создания приложений на основе многоагентных систем является платформа SISS [126]. Платформа определяет многоуровневую программную инфраструктуру: уровень коммуникации обеспечивает качество сервиса и бесперебойность работы при сетевом взаимодействии агентов, уровень координации поддерживает совместную

работу агентов, а уровень сервисов содержит общие разделяемые сервисы.

В работе Wang и др. [124] предложен класс инфраструктур для ИП, основанных на технологиях Семантического Веб. Такие инфраструктуры ориентированы на явное представление данных и их семантики, на возможности поисковых запросов и выполнения на их основе рассуждений в различных контекстах.

В работе Sathish и di Flora [113] представлен программный каркас для разработки инфраструктур в ИП. Инфраструктура поддерживает динамическое построение сервисов, а также учитывает вопросы безопасности и конфиденциальности. Инфраструктура состоит из нескольких модулей: представление и обеспечение доступа к данным, хранилище и менеджер данных, репозиторий внешних данных и поддержка безопасности и конфиденциальности.

В работе da Costa и др. [56] рассмотрены инфраструктуры для сред повсеместных вычислений и сформулированы ключевые особенности их разработки. Предложена архитектурная модель, в которой работоспособность приложения позиционируется как одно из ключевых свойств, наравне с неоднородностью, расширяемостью и интероперабельностью. Следует отметить важное заключение авторов, что в системах повсеместных вычислений должны применяться такие стратегии обнаружения сбоев и восстановления, как контрольные точки, компенсация, изоляция и реконфигурация.

В платформе Hydra [61] (LinkSmart) рассматривается вопрос связывания IoT-сред между собой, а внутри среды предлагается использовать специальные программы-коннекторы для связи устройств, взаимодействие которых осуществляется через сторонний брокер сообщений.

Платформа A3-TAG [47] позволяет разрабатывать такую инфраструктуру, где взаимодействие устройств осуществляется за счет объединения их в группы. Между группами устанавливается сетевая связь и организуется взаимодействие СВУ. Недостатком платформы является ориентированность только на достаточно производительные устройства (напр., персональные компьютеры типа десктопы и ноутбуки).

Платформа CoCaMAAL [65] реализует инфраструктуру для IoT-среды с использованием глобального облака, через которое осуществляется накопление и обработка данных. Платформа поддерживает работу приложения с множеством сервисов, однако ориентировано только на сети телесных сенсоров BSN (body sensor network). В силу вовлечения множества таких слабопризводительных СВУ возможно возникновение частых сбоев, но платформа не предоставляет специальных средств для восстановления.

Платформа meSchup [96] включает в инфраструктуру специализированное устройство-хаб для организации переносного варианта ИП. Все устройства такого ИП взаимодействуют через этот хаб. Однако, платформой поддерживается только ограниченный круг СВУ — мобильные Android-устройства, для которых разработан специальный клиент.

В работе [80] предлагается включение в инфраструктуру дополнительного СВУ, реализующего IoT-шлюз. Он может автоматически определять, подключать и отключать доступные СВУ, а также организовать их взаимодействие и управление некоторыми параметрами этого взаимодействия. Платформа предназначена только для передачи информации от СВУ в облако и не предполагает запуск шлюза на одном из имеющихся в IoT-среде ресурсно-ограниченном СВУ.

В силу условия аппаратно-сетевого разнообразия IoT-сред, при развертывании программной инфраструктуры ИП в ресурсно-ограниченной IoT-среде возникает ряд проблем. Существующие реализации информационных хранилищ ориентированы на конкретную вычислительную среду, и развертывание в другой IoT-среде становится затруднительным. Актуальной является задача разработки такого информационного хранилища, которое может быть использовано для широкого класса IoT-сред, включая возможность настройки с учетом ограничений выбранной IoT-среды. Настройка возможна за счет управления сетевым доступом к хранилищу, основывающимся на последовательной или параллельной обработке операций и выборе множества доступных операций.

Другим условием, влияющим на развертывание программной инфраструк-

туры, выступает высокая динамика и нестабильность ресурсно-ограниченной IoT-среды. Так, число одновременно участвующих в ИП агентов (и СВУ) не фиксировано, они могут часто подключаться и отключаться. Сетевые соединения являются нестабильными, особенно в случае мобильных коммуникаций. В результате, взаимодействие агентов при совместном построении информационного сервиса может нарушаться из-за динамики участия агентов и множества разнообразных сбоев. Необходимы дополнительные программные механизмы, позволяющие обеспечить устойчивость компонентов программной инфраструктуры к сбоям.

Еще одно условие следует из наблюдаемого в настоящее время значительного разнообразия способов программирования косвенного взаимодействия для существующих реализаций информационных хранилищ. Такое разнообразие приводит к трудностям при переносе или интеграции информационного сервиса из одного ИП в другое. Разработка способа единообразного описания и программирования косвенного взаимодействия для группы агентов в заданном ИП позволило бы упростить разработку прикладного ПО, обеспечивая, в частности, эффективное повторное использование программных компонент.

1.4. Управление сетевым доступом к информационному хранилищу

В указанных условиях аппаратно-сетевого разнообразия IoT-сред актуальной становится задача разработки информационного хранилища, которое может быть запущено в различных IoT-средах (на широком круге СВУ, уже имеющихся в среде). Данную задачу предлагается решать на основе предоставления возможностей управления сетевым доступом к информационному хранилищу.

Доступ участников ИП к информационному хранилищу основан на получении или изменении содержимого информационного хранилища программными агентами в ходе их коллективного взаимодействия. Целью обеспечения доступа участника к информационному хранилищу является построение и до-

ставка информационного сервиса. Наиболее распространенной моделью организации доступа к содержимому информационного хранилища является модель «классной доски» [54]. Модель подразумевает наличие единого информационного пространства, в котором агенты могут сохранять данные для совместного использования или получать их. Взаимодействие между агентами осуществляется косвенным образом через разделяемое пространство данных. Например, в платформе Smart-M3 единое информационное пространство представлено с помощью модели данных RDF [77]. Взаимодействие на основе модели «классной доски» расширяется с помощью модели «публикации/подписки» [62]. В результате агенты могут не только опубликовать информацию в информационном хранилище, но и подписаться на обновления этой информации. При каждом изменении интересующей агента информации, он будет получать оповещения с обновленными данными.

Под управлением сетевым доступом имеются ввиду способы выполнения и виды доступных операций доступа для получения и изменения содержимого информационного хранилища. Существует два способа выполнения операций доступа: последовательный и параллельный. Последовательность или параллельность доступа к информационному хранилищу осуществляется как на сетевом уровне при взаимодействии агентов с информационным хранилищем, так и на уровне выполнения операций над содержимым информационного хранилища. Последовательный доступ позволят сохранить очередность операций в ходе их выполнения от одного агента. Параллельный доступ предоставляет возможность эффективнее обрабатывать операции от множества агентов. В то же время, возможности для организации параллельности в ресурсно-ограниченных СВУ существенно ограничены. Таким образом, возникает задача настройки использования последовательного способа или вариантов параллельной обработки операций доступа к информационному хранилищу с учетом ограничений выбранного СВУ в заданной IoT-среде.

Косвенное взаимодействие между агентами основывается на операциях

доступа к содержимому информационного хранилища. Основные виды операций над информацией, используемые в большинстве существующих платформ, представлены в [105]. В таблице 1.4 перечислены основные операции, поддерживаемые в наиболее распространенных платформах.

Операции publish и read являются традиционными для баз данных операциями добавления и чтения данных. Операция take получает, а затем удаляет фрагмент данных. Операции с префиксом b- являются блокирующими, т.е. выполняют ожидание, пока хотя бы один результат не будет получен. Операция subscribe является постоянным запросом, отслеживающим изменения в информационном хранилище и оповещающая об этих изменениях. Для реализации косвенного взаимодействия между агентами и информационным хранилищем были реализованы специальные протоколы. Одним из примеров является протокол Smart Spaces Access Protocol (SSAP) [77] и его вариации [82], а также протокол Knowledge Sharing Protocol [81].

Рассмотрим существующие подходы к управлению и обеспечению доступа к информационному хранилищу. В работах [110, 123] представлен класс систем публикации/подписки в которых информация только передается, но не хранится, а также не поддерживаются ресурсно-ограниченные СВУ. В таких системах существует только два вида операций — это публикация и подписка.

В работе [124] для представления содержимого информационного хранилища используется модель классной доски на основе онтологического подхода. Используются различные операции для сопоставления онтологий. Однако в подходе отсутствует возможность настройка параллельности и набора операций, что не позволяет запускать систему на слабопроизводительных СВУ.

Платформа A3-tag [47] обеспечивает управление сетевым доступом за счет балансировки нагрузки. Она имеет модульную архитектуру состоящую из модуля балансировки деревьев (топологии), модуля балансировки загрузки (каждого сенсора), модуля обработки запросов, модуля кеширования, модуль управления группами. Однако данная платформа поддерживает работ только с бес-

Таблица 1.4. Операции доступа к информационному хранилищу в различных платформах вычислений на пространствах

	TripCom	**TSC**	**SENS**	**SemWebSpaces**	**CSpaces**	**Smart-M3 (SSAP)**
publish	out	create	publish	outr, claim	write	insert
read	rda, rd, rdg without timeout parameter	read	tryReceive	endorse, extract (non-blocking mode)	read	query
b-read	rda, rd, rdg with timeout parameter	waitTo-Read	receive	endorse, extract (blocking mode)	waitTo-Read	{not available}
take	ina, in, ing without timeout parameter	take	tryDelete	in (non-blocking mode)	take	remove (only delete)
b-take	ina, in, ing with timeout parameter	waitTo-Take	delete	in (blocking mode)	waitTo-Take	{not available}
subscribe	subscribe	subscribe	subscribe	{not available}	subscribe	subscribe

проводными сенсорными сетями WSN.

В работе [117] для организации доступа к информационному хранилищу используются три модуля: модуль обнаружения сервисов, модуль оценки и композиции, база знаний. Для организации сервисов используются онтологии трех уровней: предметной области, оценочная онтология и онтология устройств.

В платформе Smart-M3 информационное хранилище реализуется в виде брокера SIB. Основным протоколом, обеспечивающим доступ к содержимому информационного хранилища, является протокол SSAP [77]. Протокол использует сообщения в формате XML и поддерживает следующие операции: операции

присоединения к ИП (join) и отсоединения (leave), запроса (получения) определенных RDF-троек (query), добавления RDF-троек (insert), удаления RDF-троек (remove), обновления RDF-троек (update), в одной транзакции происходит удаление и добавление); подписки (subscribe) и отписки (unsubscribe) на получение изменений определенных троек.

Для обеспечения доступа к содержимому информационного хранилища существует также протокол KSP [81]. В отличии от протокола SSAP протокол KSP имеет бинарный формат. Бинарный формат является компактным и быстрым для обработки, но не настолько гибок как формат XML, используемый в протоколе SSAP. Бинарный формат подходит для ресурно-ограниченных малопроизводительных устройств. Все операции в протоколе KSP выполняются только с помощью языка запросов SPARQL 1.1 (и SPARQL UPDATE). В SSAP язык запросов SPARQL является одним из двух доступных способов формирования запроса. Имеется возможность указать максимальный размер ответа от информационного хранилища. Таким образом может быть уменьшен объем сетевого трафика. В протоколе KSP отсутствуют операции join и leave. Также добавлены дополнительные постоянные операции помимо подписки, которые производят изменения в содержимом информационного хранилища.

В таблице 1.5 систематизированы все операции, поддерживаемые информационным хранилищем. Основными операциями являются операции, реализуемые протоколом SSAP. Они включают в себя операции для работы с сессиями, доступа и управления данными и операцию постоянного запроса (подписки). Операции работы с сессиями подразумевают, что каждый агент в начале своей работы выполняет операцию присоединения (join) к ИП, которая инициирует его сессию взаимодействия с ИП. В конце работы агент выполняет операцию отсоединения (leave) и его сессия завершается. Сессии позволяют отслеживать какие действия каким агентом были выполнены, что может использоваться, например, для разграничения прав доступа к различным фрагментам разделяемых данных. Операции доступа и управления данными необходимы для за-

Таблица 1.5. Систематизация операций поддерживаемых информационным хранилищем

Класс	Тип	Операции
Основные операции	Работа с сессией	Join, Leave
	Доступ и управление данными	Instant Query, Insert, Remove, Update
	Постоянные операции	(Un)Subscribe
Расширенные операции		Persistent Insert, Remove, Update
		конфигурационные правила
SPARQL запросы	SPARQL	SELECT, CONSTRUCT, ASK, DESCRIBE
	SPARQL Update	INSERT, DELETE, INSERT DATA, DELETE DATA

проса (query) определенной информации из информационного хранилища или для модификации информации: добавления (insert), удаления (remove) или обновления (update). Такие операции являются атомарными и обрабатываются информационным хранилищем мгновенно.

В информационном хранилище также поддерживаются постоянные операции — операции, которые устанавливаются по запросу от агента, а далее выполняются брокером каждый раз при наступлении определенного условия и до запроса отмены такой операции от агента. Основной постоянной операцией является операция подписки (постоянный запрос). При выполнении операции подписки агент получает от информационного хранилища все изменения данных, соответствующие запросу отправленному при подписке.

Расширенные постоянные операции включают в себя постоянное добавление, удаление и обновление. При отправке такой операции агентом она выполняется информационным хранилищем каждый раз при выполнении определенного условия. Другой расширенной операцией являются конфигурационные пра-

вила информационного хранилища. Они схожи с постоянными операциями, но устанавливаются не агентом, а администратором информационного хранилища при его настройке и запуске.

Отдельный тип операций, обрабатываемых информационным хранилищем — это SPARQL запросы. Они включают в себя непосредственно запросы на языке запросов SPARQL и SPARQL Update. SPARQL запросы позволяют формулировать более сложные запросы с условиями, по сравнению с основной операцией query. SPARQL Update запросы делают возможным добавление и удаление информации так, как это делают операции insert и remove.

Существующие подходы управления сетевым доступом к информационному хранилищу предлагают использование различных протоколов доступа. Например, в работе [81] предлагается бинарный протокол доступа, предоставляющий операции для работы с небольшими объемами информации в хранилище. Однако, рассмотренные выше модели управления сетевым доступом не предполагают настройки информационного хранилища и реализуют конкретный вариант под конкретную IoT-среду.

Для того, чтобы развернуть информационное хранилище в разнообразных IoT-средах требуется иметь возможность настройки данного хранилища под конкретную среду. Такая настройка может быть осуществлена за счет управления сетевым доступом к информационному хранилищу. Возможность управления подразумевает выбор способа обработки операций доступа и выбор поддерживаемых операций доступа. Существует два способа обработки операций доступа: последовательный и параллельный. В условиях динамики IoT-сред параллельный доступ предоставляет возможность более быстрого обслуживания множества агентов за счет больших требований к ресурсам [18]. Выбор операций доступа предоставляет агентам возможность выполнять более сложные операции при построении информационного сервиса, но также увеличивает требования к ресурсам среды. Модель управления сетевым доступом к информационному хранилищу представлена в разделе 2.2.

1.5. Обеспечение устойчивости компонентов программной инфраструктуры к сбоям IoT-среды

В условиях нестабильности и динамики IoT-сред актуальной является задача обеспечения устойчивости к сбоям среды. Устойчивость к сбоям может быть реализована на уровне программной инфраструктуры ИП.

Программная инфраструктура предоставляет средства для функционирования приложения и обеспечивает корректность его работы. Важной характеристикой выступает работоспособность (от англ. термина dependability) приложения. Работоспособное приложение предоставляет услуги, которым можно обоснованно доверять [101]. Инфраструктура должна обеспечивать следующие общие свойства работоспособности программного приложения:

- доступность (availability): готовность к использованию;
- безотказность (reliability): продолжительное предоставление сервиса;
- безопасность (safety): отсутствие катастрофических последствий на окружающую среду;
- конфиденциальность (confidentiality): предотвращение неавторизованного доступа;
- целостность (integrity) используемых и обрабатываемых данных;
- восстановление работоспособности (maintainability).

Сбои во время выполнения приложения снижают его работоспособность [21]. Обеспечение устойчивости к сбоям включает методы, позволяющие приложению предоставлять свои сервисы в условиях возникновения сбоев.

В настоящее время для приложений, основанных на платформе Smart-M3, характерны проблемы с безотказностью работы и сохранением целостности данных, что приводит к снижению доступности сервисов. Например, сбои в операции подписки приводят к некорректной работе приложения — от перерывов до полного прекращения предоставления сервисов.

Рассмотрим три причины возникновения сбоев, влияющих на работу информационного сервиса: 1) сбои информационного хранилища, 2) сбои инфраструктурного агента, 3) сбои сети передачи данных.

Сбои информационного хранилища являются наиболее серьезными, т.к. влияют на работу всех агентов, взаимодействующих через него. Такие сбои связаны как с внутренними ошибками текущей реализации хранилища (статус исследовательского прототипа), так и влиянием нагрузки на информационное хранилище при его работе с множеством параллельных агентов, частыми запросами и с большими объемами обрабатываемой информации. Сбои инфраструктурного агента происходят как из-за наличия внутренних ошибок, так и из-за динамичного участия агента в ИП. Сбои сети передачи данных связаны с ее загруженностью, приводящей к длительным задержкам или потерям данных, что особенно характерно для беспроводных коммуникаций.

Из всех операций доступа наиболее подверженной сбоям является операция подписки, вследствие наличия отдельного сетевого соединения для подписок. Сбои в сети приводят к потерям оповещений от информационного хранилища к агентам по подписке (оповещения об изменении данных, на которые подписан агент). При потере оповещения подписки агент не отреагирует должным образом, следовательно, качество предоставляемого информационного сервиса будет нарушено. Разрыв сетевого соединения для подписки требует от агента выполнения повторной подписки (перезапуска).

Распространенным подходом для повышения устойчивости к сбоям в многоагентных системах является реплицирование [75]. В систему добавляются копии агентов, которые используются при возникновении сбоев в оригинальных агентах. В работе [102] предлагается использование дополнительных агентов, которые контролируют остальных агентов и обнаруживают сбои. В платформе A3-TAG [47] реплицирование используется для замены супервизоров при их отказе (при этом сохраняется дополнительная информация для нового супервизора). Для платформы Smart-M3 в работе Васильева и др. [118] предложен

механизм для замены агента, в котором произошел сбой, на другого. Очевидно, что такие механизмы требуют увеличения инфраструктуры приложения, пропорционального числу агентов. В диссертационной работе предлагаются решения, не требующие такого значительного расширения инфраструктуры.

В работе [98] рассматриваются различные подходы для повышения устойчивости к сбоям в грид-системах. В частности, помимо репликации, выделяется стратегия реконфигурации, которая заключается в обнаружении сбоя и восстановлении состояния системы после сбоя. В работе [78] предлагается использовать специальные компоненты-агенты для мониторинга процесса, сети и восстановления их при сбое. Данные подходы для грид-систем используется в предлагаемом нами далее решении восстановления подписки после сбоя в ИП ресурсно-ограниченной IoT-среды.

В условиях нестабильности и динамики IoT-среды актуальной является задача обеспечения устойчивости компонентов программной инфраструктуры к сбоям. В диссертационной работе исследуются решения этой задачи на уровне программной инфраструктуры ИП [7]. В разделе 2.3 представлена модель обеспечения устойчивости компонентов программной инфраструктуры к сбоям IoT-среды. Предлагается многоуровневая система для восстановления компонентов программной инфраструктуры после сбоев на уровне данных, операции подписки, сетевого соединения и вычислительного процесса, запущенного на конкретном устройстве.

1.6. Построение и доставка сервисов на основе взаимодействия программных агентов

В условиях разнообразия IoT-сред и платформ, для развертывания ИП в таких средах наблюдается разнообразие способов программирования взаимодействия программных агентов. При разработке информационных сервисов используются различные частные решения для программной реализации косвенного взаимодействия, что не позволяет при создании требуемого ИП простым

образом интегрировать в него сервисы, ранее реализованные для других ИП. Разнообразие способов программирования взаимодействия приводит к трудностям при переносе или интеграции информационного сервиса из одного ИП в другое. Унифицированный способ программирования участия каждого агента в построении информационного сервиса позволит сократить трудоемкость прикладной программной разработки, в том числе и обеспечивая повторное использование программных компонент.

Информационный сервис определяет последовательность действий (взаимодействий между агентами) по построению сервиса, инициируемую со стороны участника ИП и завершающуюся доставкой сервиса конечному пользователю. Инициация может быть явной (команда пользователя) или неявной (обнаружение потребности на основе контекста). Доставка сервиса осуществляется в форме визуального представления информации пользователю или пользователь непосредственно наблюдает изменения в физическом мире.

Построение и развертывание в IoT-среде прикладных сервисно-ориентированных систем требует новых технологий, поддерживающих как построение информационных сервисов, так и предоставление этих сервисов пользователям в условиях разнообразия, нестабильности и динамики современных IoT-сред. Исследования и разработки в этой области ведутся сейчас на мировом уровне как зарубежными, так и российскими исследователями, см. напр. [53, 86, 44, 122, 125, 129, 91].

В связи с большим разнообразием СВУ, взаимодействующих в IoT-среде, и неоднородностью данных для обеспечения интероперабельности в различных приложениях требуется обеспечить совместимость таких устройств на уровне данных. Одним из актуальных исследований в этом направлении является изучение применения технологий Семантического веб в условиях IoT-сред [41, 40, 63]. Использование семантических описаний данных с различных устройств позволяет осуществлять моделирование и интеграцию знаний, используемых в приложениях. Например, в работе [58] рассматривается создание

платформы, позволяющей на основе семантического моделирования и связанных данных описывать ресурсы IoT-сред и устанавливать связи между ними для обмена данными.

Семантический веб может применяться для построения информационных сервисов в IoT-средах. Вопросам применения технологий Семантического веб и, в частности, применению онтологий при разработке интеллектуальных систем посвящены работы [30, 37, 49, 74, 45, 72]. Информация в Семантическом веб представляется с помощью модели данных RDF. Представление данных с помощью RDF позволяет описывать семантику информации из различных разнородных источников и связывать эту информацию, а онтологии позволяют структурировать семантические данные. В работе [55] рассмотрены различные методы доступа к RDF-данным. Одним из наиболее распространенных и эффективных подходов является запрос RDF-данных из специализированных точек доступа — централизованных хранилищ. Модель RDF также может применяться в отдельных системах для организации сбора и обмена информацией.

Системы, использующие вычисления на пространствах кортежей, позволяют организовать координационную инфраструктуру для построения и доставки сервисов. В работе [106] приводится обзор технологии вычислений на пространствах кортежей для работы с семантическими данными. За счет использования централизованного разделяемого пространства семантической информации такие системы особенно удобны для использования в локализованных вычислительных IoT-средах.

Еще одним подходом для осуществления построения и доставки информационных сервисов на основе обработки и обмена семантической информацией выступают системы публикации/подписки. Так, в работах [123, 108, 114] рассматриваются системы публикации/подписки, использующие модель RDF для обмена сообщениями между публикующими агентами и подписчиками. Использование модели RDF для представления данных позволяет осуществлять взаимодействие между множеством разнородных участников вычислительной

среды.

В работах [124, 52] предлагается создавать общее пространство семантической информации в рамках подхода ИП. Это позволяет применять технологии Семантического веб в средах повсеместных вычислений, таких как IoT-среды. Ключевым свойством является локализация — переход от глобального графа ресурсов Семантического веб к локальным подграфам ресурсов, формируемых в заданной IoT-среде.

Применение сервисно-ориентированного подхода позволяет обеспечивать обнаружение доступных сервисов. Например, в платформе Hydra [61] используются списки сервисов для предоставления доступа к различным сервисам. Однако, обеспечение интеграции различных сервисов остается задачей для прикладного разработчика.

В работе [83] разработка и композиция сервисов обеспечивается за счет использования готовых компонентов из которых строится сервис. Схожий подход применяется в платформе [96], где разработчик использует веб-сайт для описания правил передачи и обработки данных между устройствами.

Технологии Семантического веб позволяют строить информационные сервисы. Например, в работе [60] анализируются вопросы интеграции сервис-ориентированной архитектуры в Семантический веб. Такой подход позволяет создавать информационные сервисы, поддерживающие автоматизированное обнаружение и композицию сервисов. В работе [100] рассматриваются методы упреждающего обнаружения и доставки семантически-ориентированных информационных сервисов. Также возможна семантическая интеграция сервисов, как показано в работе [13]. Такая интеграция позволяет обеспечить интероперабельность (возможность взаимодействия) сервисов, работающими с разными представлениями данных.

Прикладные системы, использующие сбор и обработку данных в IoT-средах, создаются на основе специализированных программных платформ. Такие программные платформы обычно используют механизм публикации/подписки

для организации взаимодействия между участниками среды. Одной из перспективных платформ для создания подобного рода приложений является платформа Smart-M3 [77], разрабатываемая в режиме открытого кода международным исследовательским сообществом. Она уже экспериментально применяется в ряде европейских и российских проектов, направленных на задачи мобильного здравоохранения, туризма, поддержки бытовых и общественных окружений человека. В настоящее время ряд разработка коммерческих вариантов таких платформ ведется в закрытом режиме несколькими крупными корпорациями (напр., Google и IBM).

Вопросам построения и разработки информационных сервисов на платформе Smart-M3 в рамках подхода ИП посвящен ряд работ, см. напр. [44, 91]. Взаимодействие между участниками IoT-среды осуществляется с помощью механизма публикации/подписки через разделение информации в общем информационном содержимом. Отдельной задачей является изучение особенностей взаимодействия агентов [115]. Например, в работе [66] предлагается модель уведомлений, позволяющая организовать динамическое взаимодействие участников ИП на основе операции подписки. Работающие в общем ИП приложения могут создавать отдельные пространства информации.

В настоящее время существует множество работ, рассматривающих применение концепции ИП и платформы Smart-M3 для реализации различных информационных сервисов и приложений. Например, в [87] изучаются вопросы создания и использования сервисов на платформе Smart-M3 в IoT-средах. Для использования низкопроизводительных устройств в сервисах Smart-M3 предлагается применять бинарный протокол [81], требующий небольшой объем вычислительных ресурсов для осуществления взаимодействия с платформой. Программирование таких устройств, как правило, использует язык программирования C. Вопросам интеллектуализации приложений на Smart-M3 посвящены работы [79, 99]. В них рассматриваются способы использования логического программирования для вывода новых знаний и описания реакции программ-

ных агентов, которая основывается на семантической информации в ИП.

Построение информационного сервиса сводится к совместному получению агентами нужного фрагмента информации, доставляемого затем пользователю в удобном для последнего виде. Для того, чтобы в результате взаимодействия агентов был построен требуемый информационный сервис, при проектировании необходимо определить последовательность действий агентов [16]. В частности, в работе [115] исследуется задача координации одновременного доступа агентов к разделяемой информации и предлагается использовать такие механизмы, как семафоры, агенты-координаторы и ряд других дополнительных сущностей. Эти механизмы поддерживают организацию взаимодействия, но не определяют конкретных вариантов организации взаимодействия агентов в ИП для совместного построения информационного сервиса. Несмотря на то, что в работе [16] предложена концептуальная модель построения сервиса в ИП за счет взаимодействия агентов, нет общего метода проектирования взаимодействия агентов в ИП. При разработке информационного сервиса прикладному разработчику приходится определять собственные варианты: а) декомпозиции сервиса на агентов, б) действия каждого агента для участия в требуемом взаимодействии.

Таким образом, существует множество способов программирования косвенного взаимодействия агентов в ИП. Даже при использовании одного информационного хранилища в каждом информационном сервисе при его реализации часто используется свой подход для организации взаимодействия между агентами. Для того, чтобы осуществить перенос этого сервиса на другое информационное хранилище и ИП или чтобы интегрировать данный сервис с другим информационным сервисом требуется понимать каким образом выполняется взаимодействие между агентами. Таким образом, актуальной является задача разработки единообразного способа программирования взаимодействия агентов, позволяющего упростить организацию взаимодействия как между агентами одного сервиса, так и между агентами разных информационных сервисов. В разделе 3.2 представлена модель информационных уведомлений для програм-

мирования косвенного взаимодействия агентов в ИП.

1.7. Выводы

Рассмотрены интеллектуальные пространства с косвенным взаимодействием на основе онтолого-ориентированного представления содержимого информационного хранилища и проблемы развертывания ИП в ресурсно-ограниченных IoT-средах. Дано определение программной инфраструктуры ИП и ее базовые задачи для организации функционирования сервисов в ИП. Рассмотрены основные операции доступа к информационному хранилищу, построение и доставка информационных сервисов на основе взаимодействия агентов. Взаимодействие является базовым принципом построения сервисов и включает в себя выполнение различных операций над содержимым информационного хранилища.

Определены основные проблемы развертывания IoT-сред, а также организации взаимодействия агентов в этих средах, такие как 1) разнообразие сред, приводящая к сложностям развертывания информационного хранилища и интеграции информационных сервисов, 2) нестабильность, проявляющаяся в сбоях в компонентах программной инфраструктуры, 3) динамика, заключающаяся в динамическом количестве участников среды. Решение проблем организации взаимодействия агентов может быть осуществлено на уровне программной инфраструктуры ИП за счет управления сетевым доступом программных агентов к информационному хранилищу, обеспечением устойчивости компонентов программной инфраструктуры к сбоям и организацией событийно-ориентированного взаимодействия агентов.

Далее в работе предлагается метод разработки программной инфраструктуры интеллектуального пространства для построения информационных сервисов на основе моделей организации косвенного взаимодействия программных агентов для построения информационных сервисов в условиях разнообразия, динамики и нестабильности IoT-сред.

Глава 2

Проектирование программной инфраструктуры интеллектуального пространства

В главе предлагается метод разработки программной инфраструктуры, позволяющий создать программную инфраструктуру интеллектуального пространства с использованием доступного разнообразия слабопроизводительных СВУ, при низкой производительности локальных сетевых коммуникаций и без привлечения дополнительных СВУ. Метод основывается на двух базовых моделях проектирования, которые применяются при разработке программной инфраструктуры, и одной дополнительной модели, которая применяется при проектировании информационных сервисов ИП. Базовые модели описывают организацию взаимодействия на уровне инфраструктуры ИП и представлены в данной главе. Дополнительная модель относится к прикладной области — организации взаимодействия между агентами при проектировании информационного сервиса, поэтому она описана отдельно в главе 3. Результаты автора, представленные в данной главе, опубликованы в [71, 70, 7, 68, 93, 121].

2.1. Метод разработки программной инфраструктуры

В данном методе рассматривается разработка программной инфраструктуры ИП, которая обеспечивает организацию косвенного взаимодействия программных агентов для совместного построения информационных сервисов. Программная инфраструктура создает основу ИП, на базе которой реализуются информационные сервисы для конкретных предметных областей. Участники ИП — программные агенты, отвечающие за построение сервиса, также являются частью инфраструктуры в рамках определенной предметной области.

Метод предназначен для разработки программной инфраструктуры ИП, обладающей следующими характеристиками:

- возможностью настройки информационного хранилища для заданной ресурсно-ограниченной IoT-среды в условиях аппаратно-сетевого разнообразия,
- устойчивостью к сбоям в условиях нестабильности IoT-сред,
- возможностью событийно-ориентированного программирования участия каждого агента в построении информационного сервиса.

Метод определяет следующие этапы, на каждом из которых выполняется решение ряда задач по разработке программной инфраструктуры ИП:

1. разработка информационного хранилища (системный программист);
2. настройка и запуск информационного хранилища с созданием требуемого ИП в выбранной IoT-среде (администратор ИП);
3. обеспечение взаимодействия программных агентов сервиса на основе описания требуемых вариантов взаимодействия (прикладной программист);
4. обеспечение устойчивости компонентов программной инфраструктуры к сбоям IoT-среды (системный программист, прикладной программист, администратор ИП).

На этапе разработки информационного хранилища основными задачами являются проектирование архитектуры и реализация модулей хранилища для обработки сетевых запросов, управления и обработки операций сетевого доступа к содержимому информационного хранилища. Особенностями разрабатываемого в рамках метода информационного хранилища является его модульность, возможность изменения и расширения функциональности, в зависимости от требований создаваемого ИП и ограничений IoT-среды. Разработанное информационное хранилище предоставляет возможность его настройки для запуска на заданном СВУ среды.

На этапе настройки информационного хранилища решается задача настройки информационного хранилища для возможности запуска на заданном СВУ ресурсно-ограниченной IoT-среды. Настройка хранилища осуществляется за счет выбора доступного набора операций сетевого доступа и способов их параллельной обработки. При этом учитываются доступные ресурсы СВУ на котором запускается хранилище. Такая настройка позволяет развернуть ИП, которое способно полноценно функционировать в заданной IoT-среде и использовать все доступные ресурсы для построения информационных сервисов в условиях разнообразия и нестабильности ресурсно-ограниченной IoT-среды.

На этапе обеспечения взаимодействия агентов решаются задачи по определению необходимых взаимодействий между агентами при разработке информационного сервиса и описанию всех вариантов взаимодействий для организации косвенного взаимодействия агентов сервиса. Реализация взаимодействия осуществляется на стороне программных агентов в соответствии с разработанным описанием.

На этапе обеспечения устойчивости компонентов программной инфраструктуры к сбоям решается задача разработки механизмов восстановления после сбоев для информационного хранилища и инфраструктурных агентов, а также запуск этих механизмов на СВУ. Одной из проблем ресурсно-ограниченных IoT-сред являются разного рода сбои, связанные с пропаданием сетевого соединения, включением/выключением различных устройств вычислительной среды, используемых при построении сервиса. В этом случае устойчивость к сбоям является важной характеристикой всех элементов программной инфраструктуры ИП: как информационного хранилища, так и инфраструктурных агентов сервиса. Метод обеспечивает устойчивость к сбоям на основе программных механизмов восстановления после сбоев.

За счет решения этих задач становится возможным учет следующих факторов при развертывании программной инфраструктуры для заданной предметной области, для которой создается ИП в IoT-среде:

1. локализация участников: какими рамками физического окружения ограничено ИП, с использованием какой аппаратуры нужно развертывать ИП;

2. типы участников: взаимодействие каких СВУ требуется, кто является поставщиком/получателем информации, наличие динамически или постоянно присутствующих участников, сетевые протоколы взаимодействия участников с ИП;

3. количество участников: какое количество участников одновременно взаимодействует, в каких пределах их количество может изменяться и какова скорость этих изменений;

4. разделяемая информация: какой информацией (текстовая/бинарная) обмениваются участники в ходе взаимодействия в едином информационном пространстве, каков объем информации и динамика обмена.

При этом, становится возможным выполнение требований на переносимость ПО для работы на различных СВУ и потребностей по расширяемости ПО для нужд заданной предметной области [34]. Вышеприведенные факторы учитываются в ходе следующий действий, выполняемых разработчиками ИП с помощью предложенного метода, по развертыванию компонентов программной инфраструктуры:

1. информационное хранилище: выбор подходящего СВУ для развертывания, выбор используемой реализации базы данных RDF-троек и СВУ для ее развертывания, настройка требуемых операций сетевого доступа и способа их обработки, частичный перенос вычислительной нагрузки по обработке операции на самих агентов;

2. программные агенты: выбор подходящего СВУ для развертывания агента, разработка интерфейса взаимодействия агента с остальными, делегируемая агенту вычислительная нагрузка;

3. системное ПО: запуск комплекса программных механизмов восстановления после сбоев на СВУ с информационным хранилищем и агентами.

Для решения поставленных задач в методе разработки программной инфраструктуры применяются следующие три модели проектирования программного обеспечения (рисунок 2.1), полученные в данной диссертационной работе:

- концептуальная модель управления сетевым доступом программных агентов к информационному хранилищу (M_{mng});
- структурная модель обеспечения устойчивости компонентов программной инфраструктуры к сбоям IoT-среды (M_{rec});
- онтологическая модель информационных уведомлений (M_{ntf}).

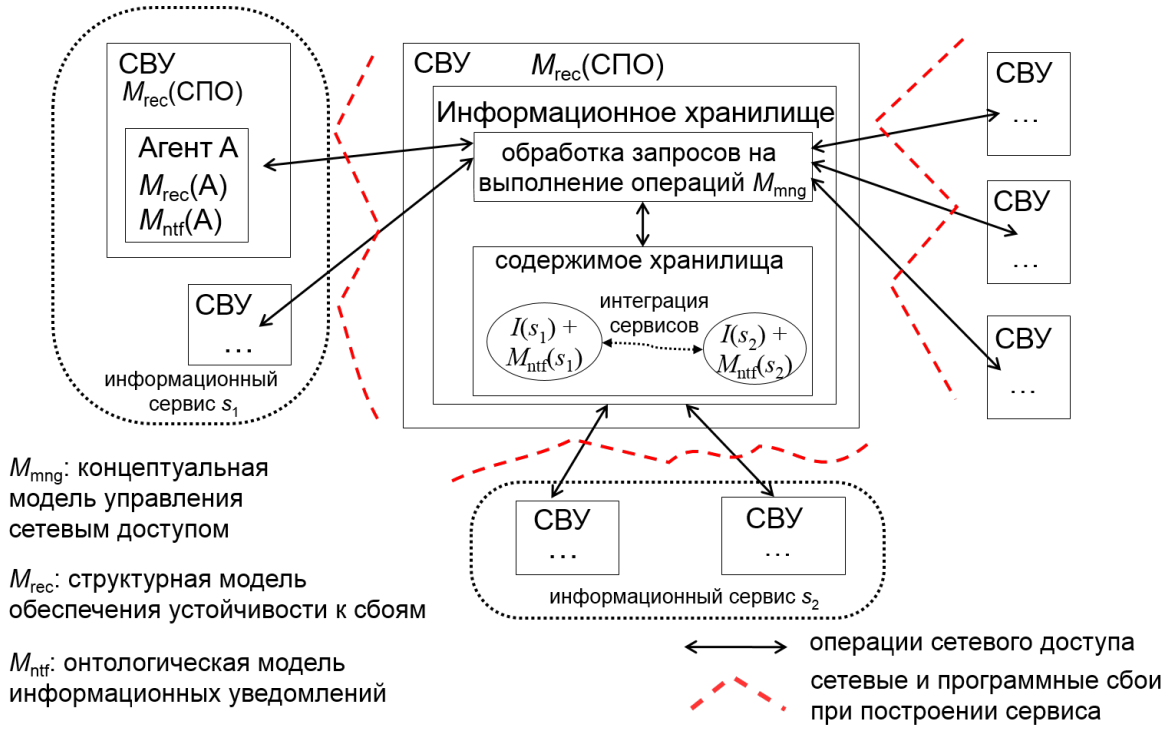


Рисунок 2.1. Применение моделей проектирования в методе разработки программной инфраструктуры

В таблице 2.1 сведены задачи по разработке программной инфраструктуры, решения которых достигаются с применением соответствующих моделей. Концептуальная модель управления сетевым доступом (см. раздел 2.2 далее)

описывает подход к разработке информационного хранилища, которое предоставляет возможность его настройки под различные СВУ ресурсно-ограниченной IoT-среды (включая запуск на слабопроизводительных устройствах). Структурная модель обеспечения устойчивости (см. раздел 2.3 далее) предлагает использование различных программных механизмов для обеспечения устойчивости к сбоям как в информационном хранилище, так и в инфраструктурных агентах при организации взаимодействия. Онтологическая модель информационных уведомлений (см. раздел 3.2 далее) предоставляет подход к описанию взаимодействия между агентами сервиса (в виде системы уведомлений) и использовании такого описания при реализации агентов.

Таким образом, описанный метод разработки программной инфраструктуры позволяет развернуть ИП в заданной ресурсно-ограниченной IoT-среде. Предоставляется возможность настройки информационного хранилища, разворачиваемого на одном из имеющихся в среде СВУ, для организации взаимодействия агентов с использованием доступного разнообразия слабопроизводительных СВУ, при низкой производительности локальных сетевых коммуникаций и без привлечения дополнительных СВУ. Для этого в методе применяются три модели проектирования программного обеспечения. В соответствии с этими моделями разрабатываются и настраиваются соответствующие компоненты программной инфраструктуры.

2.2. Модель управления сетевым доступом программных агентов к информационному хранилищу

Взаимодействие между агентами в ИП осуществляются за счет выполнения различных операций сетевого доступа над содержимым информационного хранилища. Таким образом, информационное хранилище является посредником, который обеспечивает взаимодействие агентов друг с другом. Концептуальная модель управления сетевым доступом программных агентов к инфор-

Таблица 2.1. Этапы метода разработки программной инфраструктуры

Этап	Задачи	Исполнитель
Разработка информационного хранилища (M_{mng})	Проектирование архитектуры и реализация модулей хранилища для обработки сетевых запросов, управления и обработки операций доступа к содержимому хранилища	Системный программист: при реализации инф. хранилища
Настройка информационного хранилища на СВУ (M_{mng})	Выбор способа обработки и набора операций доступа	Администратор ИП: при развертывании в заданной IoT-среде
Обеспечение взаимодействия агентов по заданной системе информационных уведомлений (M_{ntf})	Определение взаимодействий между агентами, описание взаимодействий в виде системы информационных уведомлений, реализация	Прикладной программист: при разработке агентов информационного сервиса
Обеспечение устойчивости к сбоям на СВУ (M_{rec})	Разработка механизмов восстановления после сбоев для информационного хранилища и инфраструктурных агентов	- Системный программист: при реализации инф. хранилища - Прикладной программист: при реализации агентов информационного сервиса - Администратор ИП: при развертывании инф. хранилища и агентов сервиса

мационному хранилищу описывает архитектуру информационного хранилища ИП, которое может быть развернуто в разнообразных IoT-средах. Особенностью модели управления сетевым доступом является возможность выбора поддерживаемых хранилищем операций доступа и способов их параллельного выполнения с учетом ограничений производительности в условиях разнообразия ресурсно-ограниченных IoT-сред.

Концептуальная модель управления сетевым доступом включает в себя описание архитектуры информационного хранилища (рисунок 2.2), а также

способ настройки определенных параметров информационного хранилища под заданную IoT-среду. Предлагаемая модель имеет вид $(D, C_D(R))$, где $D = (r, s, o_i, t)$ — последовательность модулей обработки операций для выбираемого набора $i = 1, 2, \ldots, n$ поддерживаемых операций, C_D — выбираемый способ выполнения вычислений для модулей в D в соответствии с множеством требований $R = (R_{\text{agn}}, R_{\text{env}}, R_{\text{dmn}})$ со стороны агентов, среды и предметной области, соответственно.

Рисунок 2.2. Концептуальная модель управления сетевым доступом программных агентов к информационному хранилищу

Сетевой доступ и обработка запросов определяются модульной структурой $D = (r, s, o_i, t)$, где r — обработчик сетевых запросов, s — планировщик, o_i — обработчик операции вида i, t — интерфейс взаимодействия с БД RDF-троек. Обработчик запросов принимает сетевые запросы на выполнение операций от множества агентов, осуществляет их разбор и преобразование из сетевого формата во внутренний и передает планировщику. Планировщик определяет, в какой момент операция должна быть выполнена, и передает ее на обработку. Для

каждого вида операции i определен собственный обработчик, в соответствии с выбранным набором операций. Возможны следующие виды операций: чтение (операция query), изменение (операции добавления insert, удаления remove, обновления update), SPARQL-запросы, подписка, постоянные операции изменения, автоматический вывод знаний. Обработчик операций реализует операцию через интерфейс взаимодействия с БД RDF-троек. Последняя осуществляет такие низкоуровневые операции, как добавление и удаление RDF-троек. Результат возвращается через интерфейс взаимодействия с БД RDF-троек и обработчик операций в планировщик, который передает его обработчику запросов для отправки агенту.

Основными особенностями концептуальной модели управления сетевым доступом, используемыми в информационном хранилище, являются:

- учет требований предметной области, среды выполнения, агентов;
- управление процессом обработки операций доступа для заданной вычислительной среды в зависимости от настройки:
  - последовательного/параллельного способа обработки;
  - набора операций.

Предложенная модульная структура позволяет управлять сетевым доступом агентов за счет настройки информационного хранилища в соответствии с тремя группами требований R: аппаратно-сетевые возможности среды выполнения R_{env}, требования программных агентов R_{agn}, требования предметной области R_{dmn}. Аппаратно-сетевые возможности отражают ограничения на имеющиеся ресурсы среды (напр., мощность процессора, оперативная память), влияющие на возможность запуска и настройку информационного хранилища. Требования программных агентов отражают индивидуально требуемые агентом возможности информационного хранилища (напр., поддерживаемые сетевые протоколы, операции доступа). Требования предметной области определяют тре-

бования информационного сервиса в целом (напр., ограничения на время выполнения). Объектами настройки являются набор поддерживаемых операций и способ управления процессом обработки операций доступа.

Управление процессом обработки операций сетевого доступа определяет будут ли обрабатываться операции соответствующим модулем информационного хранилища последовательно или параллельно. Возможность обрабатывать операции последовательно или параллельно зависит от мощности устройства, на котором запускается информационное хранилище. Известно, что в некоторых случаях более эффективным является параллельная обработка, а в некоторых — последовательная [25]. Распараллеливание заключается в создании отдельного потока для выполнения определенной задачи и возможно только в случае, когда ОС, в которой запускается информационное хранилище, позволяет создавать множество потоков для приложения. При этом следует учитывать ограничения ОС на количество возможных потоков.

В информационном хранилище выделяется несколько основных модулей, в которых возможна настройка последовательного или параллельного способа обработки операций:

- обработчик сетевых запросов;
- обработчики операций;
- интерфейс взаимодействия с БД RDF-троек.

Влияние требований на необходимость использования параллельности в определенном модуле приведено в таблице 2.2.

Обработчик сетевых запросов может обрабатывать входящие запросы параллельно или последовательно. При параллельном способе обработки запросы от нескольких агентов получаются и обрабатываются одновременно. Параллельная обработка сетевых запросов позволяет повысить эффективность обработки запросов от агентов, однако при этом увеличивается использование оперативной

Таблица 2.2. Влияние требований на необходимость использования параллельности

Модуль	Требования			
	Предметная область	Агенты	Среда	
Обработчик запросов	обработка запросов от множества агентов одновременно	поддержка сетевого соединения с агентами при большой нагрузке	повышение скорости обработки сетевых запросов при увеличении объема используемой оперативной памяти	- поддержка распараллеливания со стороны ОС - ограничения на количество потоков
Планировщик	-	поддержка сетевого соединения с агентами при большой нагрузке	-	
Обработчик операции	-	уменьшение времени ответа на операцию, (например, для уведомлений подписки)	использование нескольких ядер процессора для повышения скорости обработки операций при увеличении используемой оперативной памяти (зависит от хранилища)	
Интерфейс взаимодействия с БД RDF-троек	при поддержке хранилища: - повышение скорости обработки операций - увеличение макс. объема RDF-троек	повышение скорости обработки операций	зависит от типа хранилища, поддерживаемого средой (например, чем производительнее хранилище, тем требуется больше оперативной памяти)	

памяти. Такой вид параллельности необходим в случае, когда предметная область информационного сервиса подразумевает наличие большого количества агентов. При последовательной обработке запросов некоторые агенты могут не дождаться ответа на сетевой запрос в силу истечения времени ожидания, так как запрос еще не будет обработан. Параллельная обработка предоставляет возможность более «справедливого» доступа агентов к информационному хранилищу: время ожидания ответа на запрос у всех агентов становится приблизительно одинаковым. При большой нагрузке на остальные модули информационного хранилища в качестве ответа на сетевой запрос для поддержания связи с агентами параллельно может высылаться промежуточный ответ о необходимости ожидания. Также параллельным образом может происходить разбор запросов и преобразование операций во внутреннее представление хранилища.

Для обработки каждой операции существует собственный обработчик. Отдельный обработчик необходим для возможности включать/отключать определенный вид операций в информационном хранилище. При параллельном варианте обработки каждый вид операции может обрабатываться параллельно. Также параллельно может обрабатывать каждая операция одного вида. Параллельная обработка операций позволяет уменьшить время обслуживания агентов (время ответа на запрос о выполнении операции). Особенно это необходимо при использовании операции подписки для своевременной отправки оповещений подписки множеству агентов. Параллельная обработка операций возможна только при поддержке параллельной работы с БД, содержащей RDF-тройки.

Параллельное взаимодействие с БД RDF-троек возможно только при поддержке самой БД параллельного доступа. Таким образом, одним из параметров настройки информационного хранилища является тип БД для хранения RDF-троек. Более функциональные варианты БД предоставляют возможность параллельной работы с RDF-тройками, что увеличивает скорость обработки операций, однако такие БД требуют большие объемы оперативной памяти. Также тип БД влияет на максимальный объем хранимой в нем информации.

Дополнительно настройка параллельности может выполняться в планировщике. При последовательной обработке возможно создание очереди операций и учет в этой очереди приоритетов. Приоритет может распространяться на определенный вид операции или на все операции от определенного агента. Параллельная обработка позволяет осуществлять параллельную работу с несколькими обработчиками операций. В этом случае планировщик может более эффективно осуществлять свою функцию посредника между обработчиком сетевых запросов и обработчиками операций за счет создания отдельного вычислительного потока для каждого агента. При использовании параллельности планировщик также предоставляет возможность параллельной отправки уведомления операции подписки.

Рассмотрим последовательный и параллельный варианты настройки моду-

лей обработки операций в информационном хранилище. На рисунке 2.3 представлен последовательный вариант обработки операций. При последовательной обработке запросы от агентов в обработчике запросов помещаются в общую очередь из которой передаются в планировщик, а затем в соответствующий обработчик операций. Результат выполнения операции возвращается в общую очередь планировщика, который затем передает ее в очередь запросов и ответов обработчика запросов для дальнейшей отправки соответствующему агенту. При этом все запросы обрабатываются последовательно и существует единственный поток в котором обрабатывается операция. Отправка оповещений подписки выполняется во время обработки операции, что увеличивает время обработки. Положительным аспектом последовательной обработки является низкая требовательность к ресурсам СВУ, так как количество потоков минимально. Однако время обработки операций становится больше с увеличением количества одновременно работающих агентов и числа подписок.

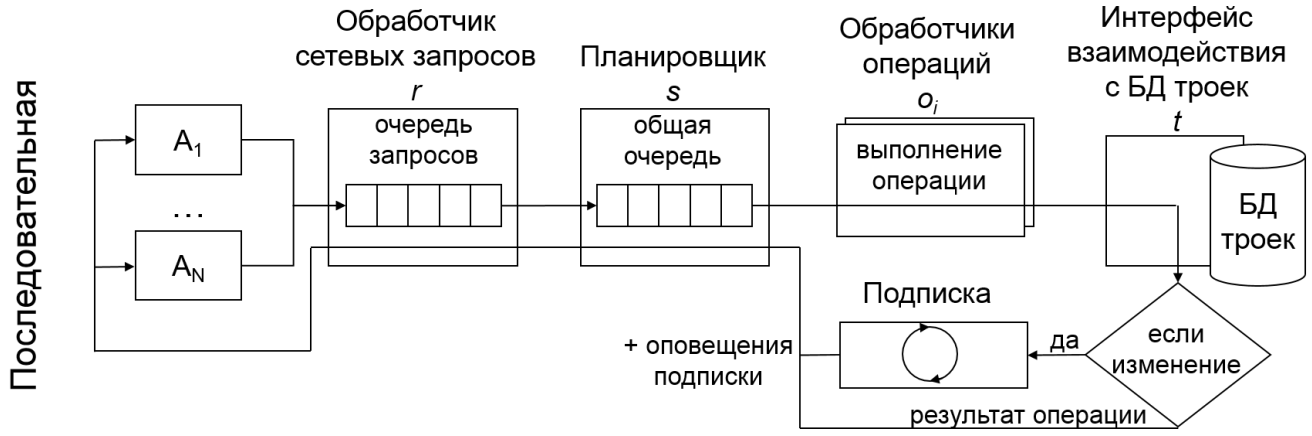


Рисунок 2.3. Последовательная обработка операций в информационном хранилище

Параллельный вариант обработки операций предполагает возможность настройки параллельности в различных модулях информационного хранилища. На рисунке 2.4 представлен вариант с параллельной обработкой в обработчике запросов и обработчике операций. При получении запросов от агентов для каждого агента создается собственный поток в обработчике запросов с очередью запросов агента. Затем запросы передаются в поток планировщика с общей

очередью. Параллельность в обработчике запросов позволяет помещать в общую очередь планировщика запросы в соответствии с различными для каждого агента приоритетами $p_1, ..., p_n$. Из планировщика запрос передается на обработку в соответствующий обработчик операций, где для него создается отдельный поток на обработку. Результат обработки помещается в общую очередь планировщика. Оповещения подписки также помещаются в общую очередь планировщика в отдельном потоке, при этом обработка операций не прерывается. Результат выполнения из планировщика затем отправляется в очередь ответов в обработчике запросов, находящуюся в отдельном потоке.

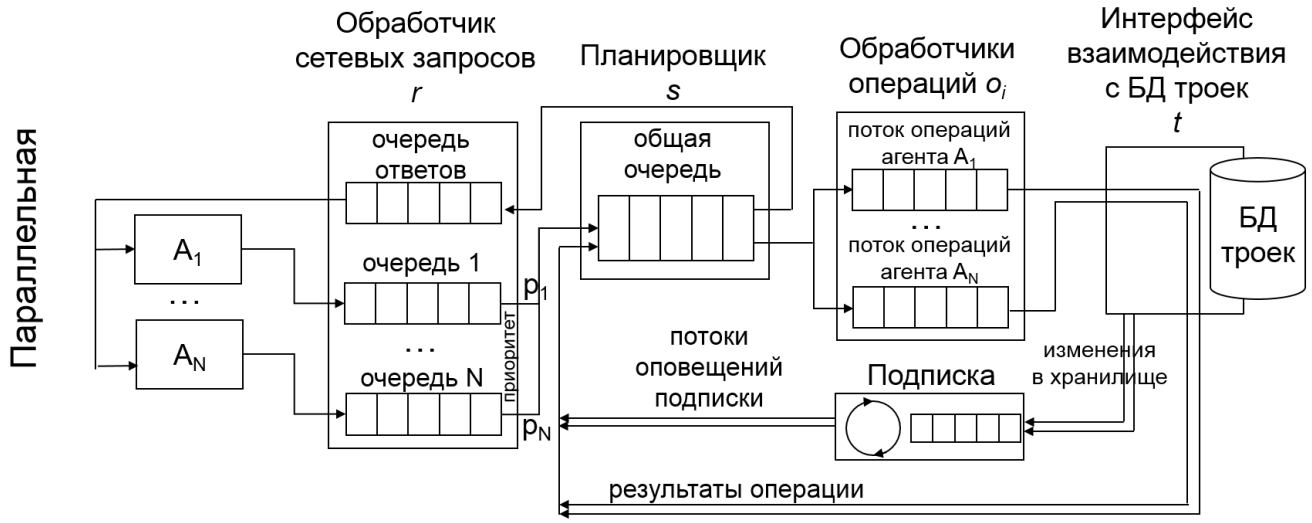


Рисунок 2.4. Параллельная обработка операций в информационном хранилище

Управление сетевым доступом агентов при помощи настройки последовательного или параллельного вариантов обработки операций в каждом модуле хранилища позволяет повысить эффективность обработки операций (при параллельности) за счет повышения требований к аппаратно-сетевым возможностям СВУ. Параллельная обработка сетевых запросов, поступающих от множества взаимодействующих агентов, позволяет контролировать отдельные сетевые соединения с агентами, а также снизить время отправки оповещений при выполнении операции подписки. Параллельная обработка операций (по видам) позволяет повысить производительность обработки операций и уменьшить время обслуживания агентов (время ответа на запрос о выполнении операции) в

случае, если БД RDF-троек поддерживает параллельный доступ.

Другим настраиваемым параметром информационного хранилища является возможность включения/отключения поддержки определенных операций. В архитектуре хранилища эта возможность реализуется с помощью подключения/отключения соответствующего модуля-обработчика операции. Подключение дополнительных операций позволяет расширить функциональность информационного сервиса, использующего данное информационное хранилище, за счет добавления более сложных операций. Однако, добавление операций увеличивает требования к ресурсам, необходимых хранилищу для выполнения таких операций.

Существует несколько основных видов операций: запросы (T-шаблоны, SPARQL), изменение (insert/update/remove), постоянные операции (подписка, постоянное изменение), дополнительные операции (вывод знаний). Влияние требований на необходимость включения/отключения определенной операции приведено в таблице 2.3.

Запросы предназначены для извлечения информации из БД RDF-троек. Наиболее распространенными являются два вида: T-шаблон и SPARQL-запрос. T-шаблон позволяет задать шаблон для получения RDF-троек, соответствующих этому шаблону. Запросы SPARQL позволяют создавать более сложные запросы с различными условиями и выполняются быстрее T-шаблонов, но требуют дополнительных ресурсов на их выполнение.

Операции изменения позволяют добавлять и удалять тройки из хранилища. Они являются основными для осуществления взаимодействия агентов, поэтому данный вид операций нельзя отключить. Дополнительным вариантом является операция обновления, которая осуществляет последовательное удаление и добавление троек в одной транзакции.

Подписка является постоянной операцией и позволяет реализовать проактивность в информационных сервисах. При выполнении подписки осуществляется оповещение агента-подписчика об изменениях в интересующей его ин-

Таблица 2.3. Влияние требований на необходимость включения/отключения операции

<table>
<tr><th rowspan="2">Тип операции</th><th rowspan="2">Операция</th><th colspan="4">Требования</th></tr>
<tr><th>Предметная область</th><th>Агенты</th><th colspan="2">Среда</th></tr>
<tr><td>Запросы</td><td>T-шаблон (обз.)</td><td>-</td><td>-</td><td>-</td><td rowspan="7">при возрастании мощности центрального процессора, возрастает скорость обработки операций</td></tr>
<tr><td></td><td>SPARQL</td><td>возрастает скорость обработки запросов</td><td>необходимость использования</td><td>увеличивается нагрузка на центральный процессор (CPU)</td></tr>
<tr><td rowspan="3">Изменение</td><td>insert (обз.)</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td>remove (обз.)</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td>update</td><td>-</td><td>необходимость использования</td><td>-</td></tr>
<tr><td rowspan="2">Постоянные</td><td>подписка</td><td>проактивность</td><td rowspan="2">необходимость использования</td><td rowspan="2">чем больше клиентов, тем больше требуется оперативной памяти (RAM)</td></tr>
<tr><td>изменение</td><td>изменение информации со стороны инф. хранилища по правилам</td></tr>
<tr><td>Доп. операции</td><td>логический вывод (reasoning)</td><td>необходимость использования</td><td>необходимость использования</td><td>требуются дополнительные ресурсы RAM, CPU и доп. ПО</td></tr>
</table>

формации. Данная операция требует дополнительных накладных расходов на оперативную память (RAM) и процессорное время, поэтому в предметных областях в которых она не требуется, данную операцию целесообразно отключить.

Постоянные операции изменения являются расширенным вариантом подписки, где вместо оповещения об изменениях совершаются указанные агентом изменения. Эта операция может быть подключена в случае, когда необходимо проактивное изменение информации со стороны информационного хранилища по указанным правилам.

Дополнительные операции предоставляют более расширенный функционал для агентов, в том числе за счет использования внешних утилит для обработки информации. Примером такой операции является вывод знаний, который выполняет изменения в хранящейся информации, на основе внутренних правил информационного хранилища [79, 99]. Такой подход может осуществлять бо-

лее сложную обработку информации, чем постоянные операции изменения, но требует больше ресурсов системы, а также наличия дополнительного ПО для выполнения обработки.

Таким образом, необходимость включения/отключения операций зависит от необходимости использования данных операций в агентах информационного сервиса, при необходимости реализации таких требований предметной области, как проактивность и дополнительная обработка информации на стороне информационного хранилища, а также необходимости выполнять обработку операции за определенный промежуток времени. Скорость обработки операций зависит от мощности процессора. При этом следует учитывать ограничения среды по объему оперативной памяти для постоянных и дополнительных операций.

Выбор набора операций сетевого доступа реализуется за счет подключения нужных обработчиков операций. Управление сетевым доступом агентов при помощи расширения видов операций приводит к программной инфраструктуре, обеспечивающей развитые варианты косвенного взаимодействия агентов. При этом, усиливаются требования к вычислительным ресурсам СВУ, на котором запущено информационное хранилище.

Кроме конфигурации набора операций и параллельной обработки при развертывании информационного хранилища возможен выбор поддерживаемых им протоколов и типа БД для хранения троек. На выбор поддерживаемого протокола влияет требование программных агентов к поддержке определенного протокола (например, использование бинарного протокола). При выборе БД для хранения троек возможно три варианта размещения хранилища: а) в оперативной памяти, б) в файле, в) во внешней СУБД (таблица 2.4).

Концептуальная модель управления сетевым доступом программных агентов к информационному хранилищу описывает архитектуру информационного хранилища для организации косвенного взаимодействия агентов. Разработанное в соответствии с моделью информационное хранилище обладает возможностью настройки для запуска на выбранном СВУ ресурсно-ограниченной IoT-

Таблица 2.4. Типы БД для хранения троек

Тип хранилища	Надежность	Особенности
В оперативной памяти	Данные не сохраняются	Высокая скорость обработки операций, объем троек зависит от количества оперативной памяти
В файле	Возможность сохранения данных, но есть риск повреждения файла	Низкая скорость обработки операций, с ростом количества троек — скорость обработки снижается
Во внешней БД	Высокая надежность	Средняя скорость обработки операций, большие объемы троек

среды за счет определения набора поддерживаемых операций сетевого доступа и способов их параллельной обработки в информационном хранилище.

2.3. Модель обеспечения устойчивости компонентов программной инфраструктуры к сбоям вычислительной среды

Структурная модель обеспечения устойчивости компонентов программной инфраструктуры используется для восстановления после сбоев при построении информационных сервисов программными агентами, взаимодействующими через информационное хранилище. Работоспособность компонентов подвержена частым сбоям как в IoT-среде (включая ее сетевую компоненту), так и в работе самих инфраструктурных компонентов ИП (информационного хранилища и программных агентов). Как правило, устойчивость к сбоям заключается в возможности приложения предоставлять сервисы в условиях возникновения сбоев [101]. Наиболее чувствительной к сбоям является операция подписки. Такие проблемы, как разрыв сетевого соединения или потеря оповещений подписки приводят к сбоям, нарушающим работоспособность приложения.

Можно выделить следующие виды сбоев ИП в IoT-средах: разрыв соединения, обрыв подписки, потеря оповещения подписки, прекращение работы информационного хранилища или агента. Разрыв соединения приводит к невоз-

можности конкретного агента участвовать в построении информационного сервиса. Такому агенту требуется как можно быстрее восстановить соединение с информационным хранилищем. Разрыв соединения также приводит к другой проблеме, которая может происходить и независимо от разрыва соединения — обрыву подписки.

Обрыв подписки является частным случаем разрыва соединения. При этом теряется только соединение подписки и получение оповещений подписки становится невозможным. Для решения такой проблемы агент должен уметь проверять является ли подписка активной и работающей.

Другой проблемой, связанной с подпиской, является потеря оповещения подписки. Механизм подписки не гарантирует стопроцентной вероятности доставки оповещения подписки и в связи со сбоями сетевой пакет может потеряться. В этой ситуации агенты должны разрабатываться так, чтобы потеря оповещения подписки не являлась критичным событием. Также агент может косвенным образом определять было ли пропущено оповещение подписки по изменениям в информационном хранилище.

Другим видом сбоев является незапланированное завершение работы как самого информационного хранилища,так и агентов. Такое завершение может быть связано с ошибками при написании программной составляющей этих компонентов инфраструктуры или при сбоях в ОС.

Повысить работоспособность информационного сервиса возможно на уровне его инфраструктуры [56, 70]. Структурная модель обеспечения устойчивости компонентов программной инфраструктуры к сбоям описывает возможные уровни для программной инфраструктуры с целью размещения программных механизмов восстановления после сбоев с привязкой к участвующим СВУ и взаимодействующим агентам при построении ими информационных сервисов (рис. 2.5).

Предлагается выполнять восстановление после сбоев с использованием соответствующих программных механизмов на следующих уровнях: а) на уровне операции подписки (контроль подписки); б) на уровне сетевого соединения (пе-

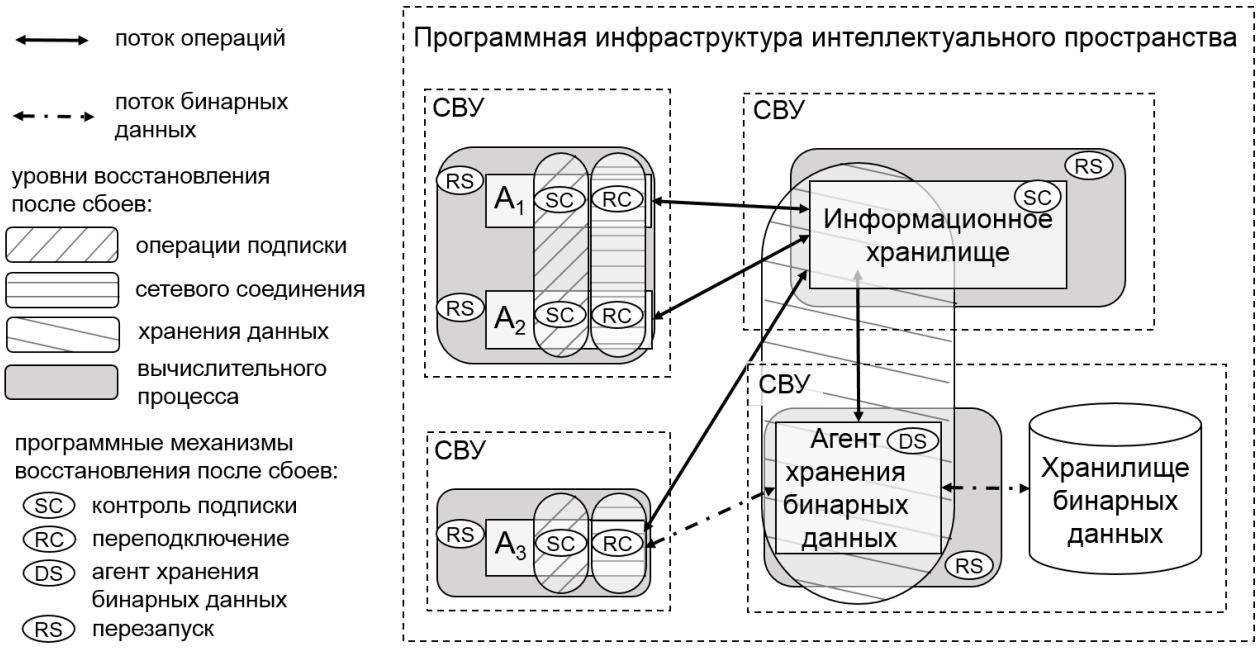


Рисунок 2.5. Структурная модель обеспечения устойчивости компонентов программной инфраструктуры к сбоям

реподключение); в) на уровне вычислительного процесса агента, запущенного на конкретном СВУ (перезапуск); и г) на уровне хранения данных (агент хранения бинарных данных). Контроль подписки и ее восстановление после сбоя требуют реализации на уровне каждого инфраструктурного агента. Данное решение позволяет реализовать инфраструктуру приложения в соответствии с рисунком 2.5.

На уровне операции подписки обеспечивается устойчивость к потерям оповещений подписки об изменениях в информационном хранилище. Такие сбои, например, возникают из-за нехватки вычислительных ресурсов на стороне информационного хранилища и из-за нестабильности беспроводных коммуникаций. Обнаружение сбоев выполняется на стороне агента, когда агент может использовать программный механизм контроля подписки SC. С его помощью агент сам производит проверку на наличие изменений в информационном хранилище (например, через регулярные промежутки времени). Обнаружение агентом изменений является и восстановлением, т.к. агент при проверке получает последние внесенные изменения. При этом, часть произошедших изменений мо-

жет быть потеряна для агента, если было несколько последовательных изменений до момента восстановления.

Контроль подписки предназначен для обнаружения сбоев в подписке в двух ситуациях: 1) при потере оповещения подписки и 2) при разрыве сетевого соединения с информационным хранилищем. В первой ситуации инфраструктурные агенты производят проверку данных подписки (напр., по таймеру) на наличие изменений. Сбой произошел, если агент обнаружил изменение данных. Таким образом, добавляется активная проверка подписки со стороны агента. Во второй ситуации агент обнаруживает разрыв соединения и должен перейти к процедуре его восстановления.

Для проверки пропущенных по подписке изменений может использоваться онтологическая модель информационных уведомлений, описанная в п. 3.2. Все сервисы приложения подписываются на собственные уведомления (специальные RDF-тройки). Уведомление представляет собой дополнительные данные в информационном хранилище, которые описывают факт изменения других данных. Если уведомление было опубликовано в информационном хранилище, но оповещение подписки не пришло, то агент может самостоятельно (автоматически или по команде пользователя) проверить наличие необработанных троек уведомления и продолжить работу. При обнаружении сбоя подписки для ее восстановления далее предлагаются механизмы перезапуска и переподключения.

На уровне сетевого соединения обеспечивается устойчивость к сбоям при разрыве сетевого соединения между агентом и информационным хранилищем. Такие сбои возникают из-за нестабильности беспроводных коммуникаций, например, СВУ переключается с одной локальной беспроводной сети на другую или теряет сигнал сети. Обнаружение сбоев выполняется на стороне СВУ агента, когда агент может использовать программный механизм переподключения (RC). При переподключении устанавливается новое сетевое соединение с информационным хранилищем, и агент может продолжить совместное построение сервиса с другими агентами. После переподключения также необходимо

возобновить установленные подписки (если они были): агент отправляет новый запрос к информационному хранилищу для установки подписки на нужную информацию.

На уровне вычислительного процесса агента обеспечивается устойчивость к сбоям, связанных с аварийным прекращением работы агента. Такие сбои возникают, например, из-за перезагрузки СВУ или временного отказа в ресурсах в силу их необходимости для выполнения основных функций СВУ. Один или более агентов, запущенных на одном СВУ (отвечающих за совместное построение одного сервиса), могут использовать программный механизм перезапуска RS. С его помощью агенты запускаются на заново (средствами программной среды СВУ, например, средствами ОС) и восстанавливают предыдущее состояние построения сервиса (до возникновения сбоя). При этом, сетевые соединения с информационным хранилищем устанавливаются заново.

В механизме переподключения агент пытается возобновить сетевое соединение с информационным хранилищем, без прекращения работы самого агента. Переустановка соединения производится средствами агентов, реализующих сервис. В механизме перезапуска работа сервиса завершается, реализующие его агенты запускаются заново с установкой сетевых соединений с информационным хранилищем. Перезапуск осуществляется средствами внешней программной среды, где запущен сервис, например, средствами ОС.

На уровне хранения данных в программную инфраструктуру вводится агент хранения бинарных данных. Другие агенты при работе с объемными бинарными данными могут использовать программный механизм DS для раздельного хранения бинарных данных и их семантики. С его помощью обеспечивается дополнительная защита от сбоев при передаче объемных данных по сети в силу использования специализированного хранилища (например, для мультимедийной информации). За сохранение и организацию доступа отвечает агент хранения бинарных данных, который получает сами данные от других агентов и публикует в информационном хранилище семантическую информацию в виде

ссылок, определяющих способ доступа к этим бинарным данным.

В таблице 2.5 представлены учитываемые моделью исключительные ситуации, основные виды сбоев и варианты применения механизмов восстановления после сбоев. Тем не менее модель не учитывает такие нештатные ситуации, как исчерпание заряда батареи, поломка идентифицируемого аппаратного элемента СВУ, выход из строя СВУ по неопределенной причине и другие. Таким образом, многоуровневая система восстановления компонентов программной инфраструктуры от сбоев позволяет обеспечить устойчивость к сбоям в IoT-средах именно программной, а не аппаратной составляющей инфраструктуры ИП.

Таблица 2.5. Виды сбоев в интеллектуальном пространстве

Исключительная ситуация	**Вид сбоя**	**Восстановление**
Потеря данных при передаче по сети	Потеря оповещения подписки	Активная проверка изменений информации по подписке, восстановление подписки
Отказ сетевого соединения, переключение СВУ на другую сеть	Обрыв подписки	Восстановление подписки на стороне агента
	Разрыв соединения	Переподключение, восстановление подписок
Утечка памяти, недоступность ресурса СВУ, программная ошибка в программном коде	Прекращение работы, «зависание» информационного хранилища	Перезапуск хранилища, восстановление информации из хранилища, восстановление подписок для агентов
	Прекращение работы/зависание агента	Перезапуск агента, восстановление локального и разделяемого состояний агента, восстановление подписок

Пример 2.1 («Система проведения мероприятий совместной деятельности: обеспечение устойчивости к сбоям»). Рассмотрим применение структурной модели обеспечения устойчивости компонентов программной инфраструктуры к сбоям на основе примера 1.1 «Система проведения мероприятий совместной деятельности» (с. 21) [71, 70]. Информационное хранилище запускается на серверной ЭВМ. Агенты для сервиса проведения конференции Conference-service и сервиса Content-service также запускаются на ЭВМ вместе с информационным хранилищем, чтобы уменьшить время сетевой передачи между агентами и ИП.

Агент отображения программы мероприятия Agenda-service и агент показа презентаций Presentation-service запускаются на компьютерах, соединенных с проекторами. В каждом из агентов реализованы механизмы контроля подписки и переподключения. На компьютерах где запущены агенты и информационное хранилище со стороны ОС ведется контроль за выполнением их процессов с помощью механизма перезапуска. При сбое в каком-нибудь агенте сначала будет применен механизм контроля подписки. Если проблема была связана не с подпиской, то будет выполнено переподключение к информационному хранилищу. В случае если агент завис и не реагирует на внешние сигналы будет выполнен перезапуск агента или информационного хранилища.

Структурная модель обеспечения устойчивости компонентов программной инфраструктуры к сбоям IoT-среды описывает многоуровневую систему восстановления после сбоев на уровнях: а) операции подписки, б) сетевых соединений между агентами и информационным хранилищем, в) вычислительных процессов агентов на СВУ, г) хранения данных с разделением семантической информации и объемных бинарных данных. Такая система позволяет повысить устойчивость к сбоям программных агентов и информационного хранилища при осуществлении их взаимодействия, тем самым повышая эффективность построения и доступность информационного сервиса в целом.

2.4. Выводы

Предложенный метод разработки программной инфраструктуры позволяет создать инфраструктуру ИП в разнообразных ресурсно-ограниченных IoT-средах, а также обеспечить устойчивость компонентов инфраструктуры к сбоям при построении информационного сервиса. На основе модели управления сетевым доступом программных агентов возможно создание настраиваемого информационного хранилища, которое может быть настроено в соответствии с различными аппаратно-сетевыми ограничениями IoT-среды и требованиями

предметной области информационного сервиса. Настройка позволяет запустить информационное хранилище на различных устройствах с учетом доступных для него ресурсов. С помощью модели обеспечения устойчивости к сбоям все компоненты программной инфраструктуры используют многоуровневую систему восстановления после сбоев, что позволяет этим элементам восстанавливать свою работоспособность при различных сбоях IoT-среды.

Глава 3

Построение сервисов в интеллектуальном пространстве

В главе рассматривается построение сервисов в интеллектуальном пространстве. Предлагаются шаблоны взаимодействия и модель информационных уведомлений для программирования косвенного взаимодействия программных агентов единообразным способом. Шаблоны взаимодействия описывают типичные сценарии взаимодействия агентов при построении сервиса и могут быть использованы разработчиком сервиса при проектировании приложения. Модель информационных уведомлений предоставляет способ описания взаимодействия агентов на основе семантического событийно-ориентированного представления взаимодействия агентов в виде онтологических классов и свойств и схему использования этого описания. На основе шаблонов и модели взаимодействия может быть сгенерирован программный код, реализующий взаимодействия между агентами. Результаты автора, представленные в данной главе, опубликованы в [5, 66, 6, 92, 95, 85, 89, 67, 69].

3.1. Разработка агентов для совместного решения задачи на основе шаблонного подхода

Задача организации взаимодействия агентов является ключевой при разработке информационного сервиса в ИП. В силу разнообразия возможных вариантов взаимодействия агентов в ИП и с учетом необходимости вовлечения большого количества участников и источников данных требуется создание новых методов для упрощения и автоматизации разработки информационных сервисов [107, 48].

Элементарным взаимодействием будем называть бинарное отношение на множестве агентов:

$$A_{\text{snd}} \rightarrow A_{\text{rcv}}, \tag{3.1}$$

где агент-получатель A_{rcv} читает (разовая операция или по подписке) данные из содержимого информационного хранилища I, опубликованные агентом-отправителем A_{snd}. Отношение (3.1) позволяет формализовать понятие программного приложения ИП (реализующего информационный сервис), как набора всех взаимодействующих агентов. Агенты A_1 и A_2 входят в состав одного приложения, если $A_1 \to A_2$ или $A_2 \to A_1$. Рефлексивно-транзитивное замыкание отношения состава определяет классы эквивалентности, каждый соответствует некоторому приложению. Отношение (3.1) обладает следующими полезными свойствами для организации взаимодействия:

- один-ко-многим: один агент A_{snd} может взаимодействовать с несколькими агентами A_{rcv} одновременно;
- асинхронность: отправка и получение данных не блокируют работу агентов A_{snd} и A_{rcv};
- анонимность: агентам A_{snd} и A_{rcv} не требуется непосредственно знать друг друга для выполнения взаимодействия.

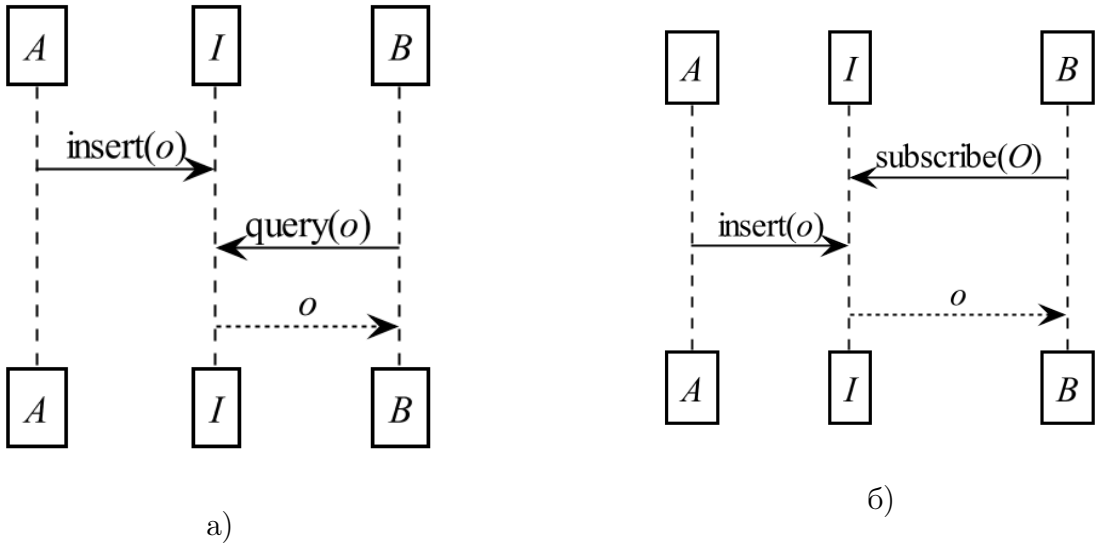
В диссертационном исследовании рассматривается один из подходов к разработке взаимодействия агентов в ИП — на основе шаблонов проектирования [11, 8, 50]. Частным случаем таких шаблонов являются шаблоны взаимодействия, описывающие организацию взаимодействия между агентами в ИП. Предлагается набор шаблонов взаимодействия, который является обобщением опыта разработки информационных сервисов ИП в соответствии с МЗ-архитектурой [94, 89, 112]. Применение прикладным разработчиком подходящего шаблона взаимодействия при проектировании информационного сервиса упрощает процесс разработки, т.к. используются уже готовые разработанные решения. Предложенные шаблоны взаимодействия также могут быть использованы как основа для генерации заготовок программного кода агентов, в котором реализуется часть логики, отвечающая за взаимодействие. Далее предлагается набор шаб-

лонов взаимодействия агентов в ИП, где каждый шаблон представлен в виде диаграммы последовательности. В качестве агентов рассматриваются агенты с ограниченной автономностью (процессоры знаний).

Шаблоны взаимодействия описывают ситуации обмена информацией между агентами в ИП. В таких ситуациях может быть задействовано два или более агентов. Взаимодействие между двумя агентами будем считать простым, если оно состоит из публикации/изменения агентом-отправителем определенных индивидов в содержимом I информационного хранилища и получении этих индивидов агентом-получателем, т.е. в результате передается информация от агента-отправителя агенту-получателю. Агент-отправитель может добавить информацию (операция insert), удалить (remove) или обновить (update — последовательное удаление и добавление). Агент-получатель может получить необходимую информацию при помощи операции запроса (query) или подписки (постоянного запроса, subscribe).

На рисунке 3.1 изображена публикация агентом A и получение агентом B информации из I по запросу в виде индивида онтологии (рисунок 3.1а) и подписке на класс (рисунок 3.1б). Запрос инициируется самим агентом-получателем разово, в определенный момент времени. Использование операции подписки ориентировано на постоянное упреждающее (проактивное) получение изменений интересующей агента информации в I. В зависимости от подписываемой информации возможны два варианта: а) подписка на класс из онтологии (оповещения подписки происходят только при создании/удалении индивидов заданного класса), б) подписка на изменение всех или конкретных свойств определенного индивида.

Подписка на класс подразумевает, что при появлении в I нового индивида определенного класса подписчик будет оповещен о появлении такого индивида, при этом он не будет оповещен об изменении свойств этого индивида. Для таких изменений используется подписка на свойства определенного индивида (рисунок 3.2). Таким образом, при публикации новой (или удалении существующей)



а) б)

Рисунок 3.1. Публикация и получение индивида o: а) по запросу, б) по подписке на класс индивида O

информации в I при помощи операции insert (remove) целесообразно использование подписки на класс, а при отслеживании изменений уже существующего индивида при помощи операции update — подписка на конкретные свойства индивида.

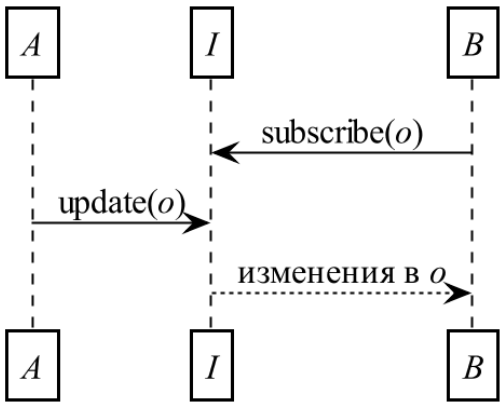


Рисунок 3.2. Получение изменений индивида o по подписке

Косвенное взаимодействие предполагает ситуацию, когда агенту-отправителю не известны агенты-получатели. Такое взаимодействие является частным случаем взаимодействия одного-ко-многим, когда агентами-получателями выступает множество одинаковых агентов $B_1, \ldots, B_n$ (рисунок 3.3). Такой подход чаще всего применяется в многопользовательских системах, где агенты-полу-

чатели являются клиентскими программами, работающими на устройствах конечного пользователя.

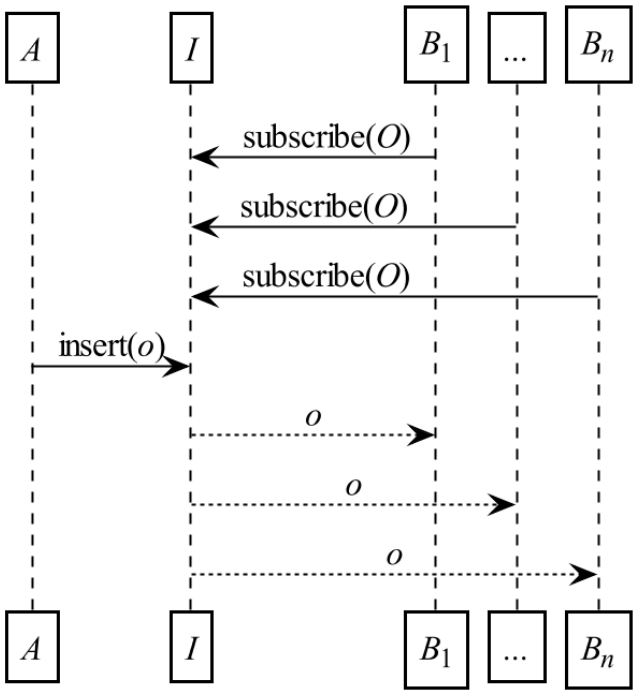
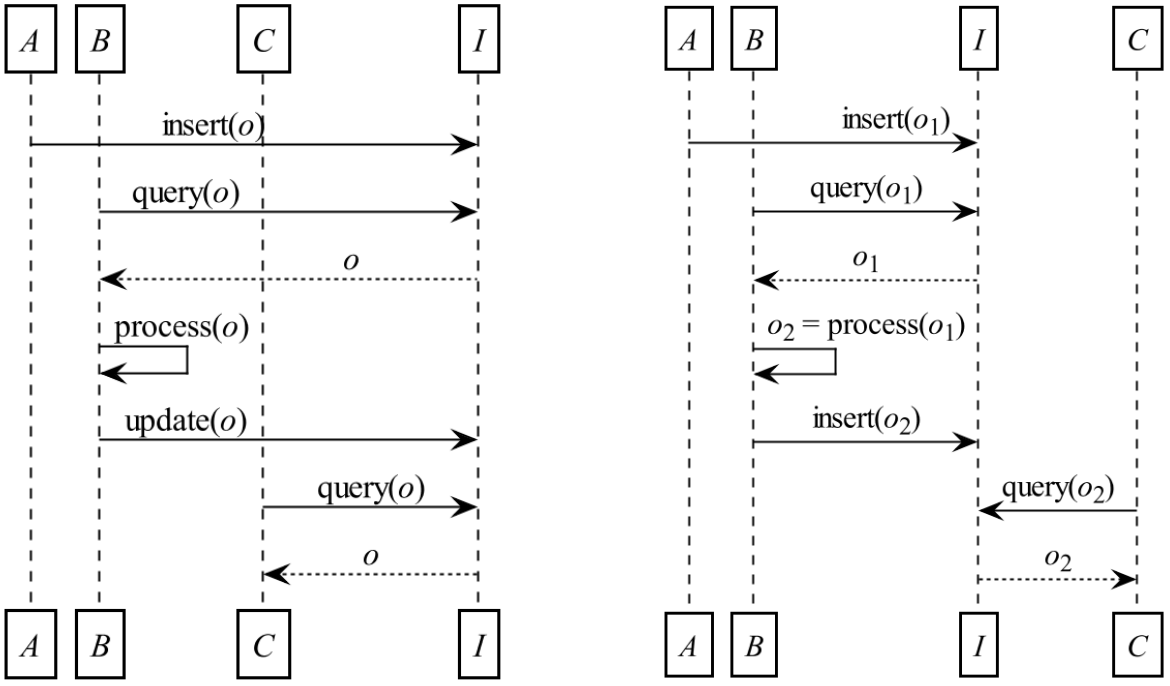


Рисунок 3.3. Взаимодействие один-ко-многим

На основе простого взаимодействия определим шаблоны взаимодействия для трех и более агентов. Шаблон последовательного взаимодействия для трех агентов заключается в следующем: A отправляет информацию B, B отправляет информацию C. Промежуточный агент B может выполнять одно из двух действий: (а) обновление существующего индивида o для последующей передачи агенту C (рисунок 3.4а), (б) публикацию нового индивида o_2 на основе полученного индивида o_1 (рисунок 3.4б). Вариант (а) заключается в дополнительной обработке (process) опубликованного индивида o агентом B перед передачей агенту C. Такая обработка может заключаться в проверке/корректировке свойств индивида или добавлении новых свойств. В варианте (б) при обработке происходит создание нового индивида o_2 согласно другой онтологии на основе существующего индивида o_1 . Агенты B и C могут получать индивидов как по запросу, так и по подписке. Данный подход может быть использован для преобразования информации (напр. конвертирование в другой формат, логический

вывод) в удобный для агента C вид (в соответствии с его онтологией). Следует отметить, что в таком случае агент C не располагает онтологией, описывающей информацию до преобразования.



а) б)

Рисунок 3.4. Последовательное взаимодействие

Вариант публикации новой информации может быть расширен за счет выполнения передачи информации в обратную сторону. В таком случае агент B выступает в роли агента-посредника (медиатора) [85, 67] (рисунок 3.5). Агент-посредник осуществляет преобразование индивидов, тем самым организуя опосредованное взаимодействие между агентами A и C. Этот подход может использоваться в случае, когда агенты A и C используют разные онтологии и агенту-отправителю не известна онтология агента-получателя. Агент B, выступающий в роли посредника, умеет работать с обеими онтологиями и осуществляет преобразование индивида o_1 из одной онтологии в индивида o_2 из другой онтологии и обратно.

Рассмотрим древовидные шаблоны взаимодействия, в которых происходит взаимодействие один-ко-многим и многие-к-одному [50]. На рисунке 3.6

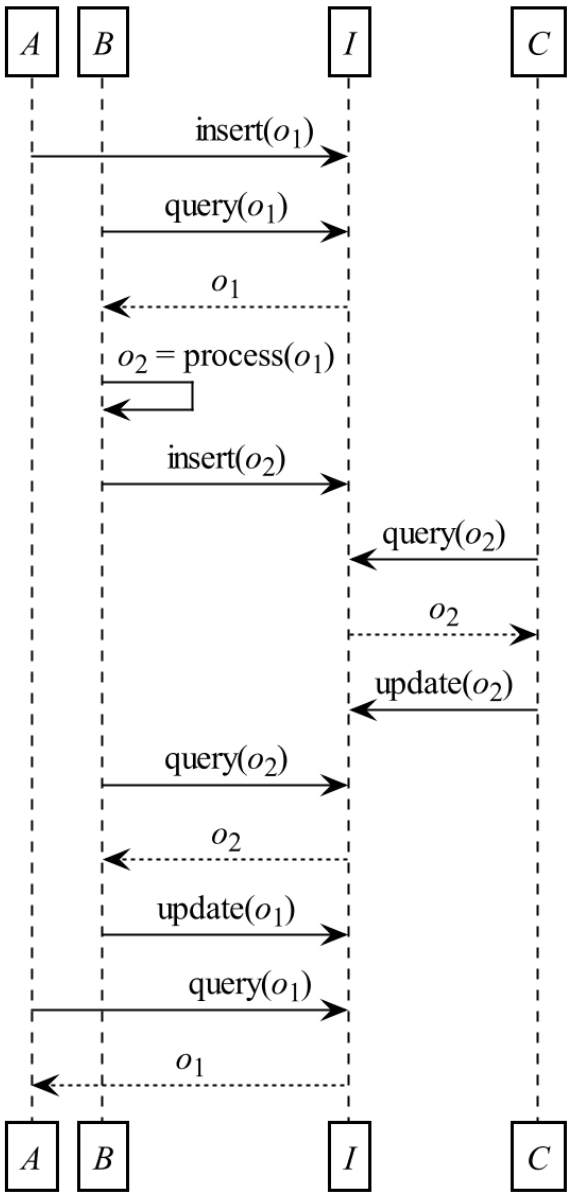


Рисунок 3.5. Агент-посредник

изображен шаблон взаимодействия «фасад». При таком варианте взаимодействия агент A должен передать информацию нескольким агентам. Используется агент-посредник, который преобразует опубликованного агентом A индивида o во множество новых различных индивидов $o_1, \ldots, o_n$, относящихся к соответствующему агенту-получателю (агенты $C_1, \ldots, C_n$). Данный вид взаимодействия отличается от простого взаимодействия один-ко-многим тем, что агенты-получатели являются разными агентами и могут использовать разные онтологии. Этот шаблон может применяться в ситуациях, когда необходимо преобразовать некоторую информацию и распространить среди множества функ-

циональноразличных агентов.

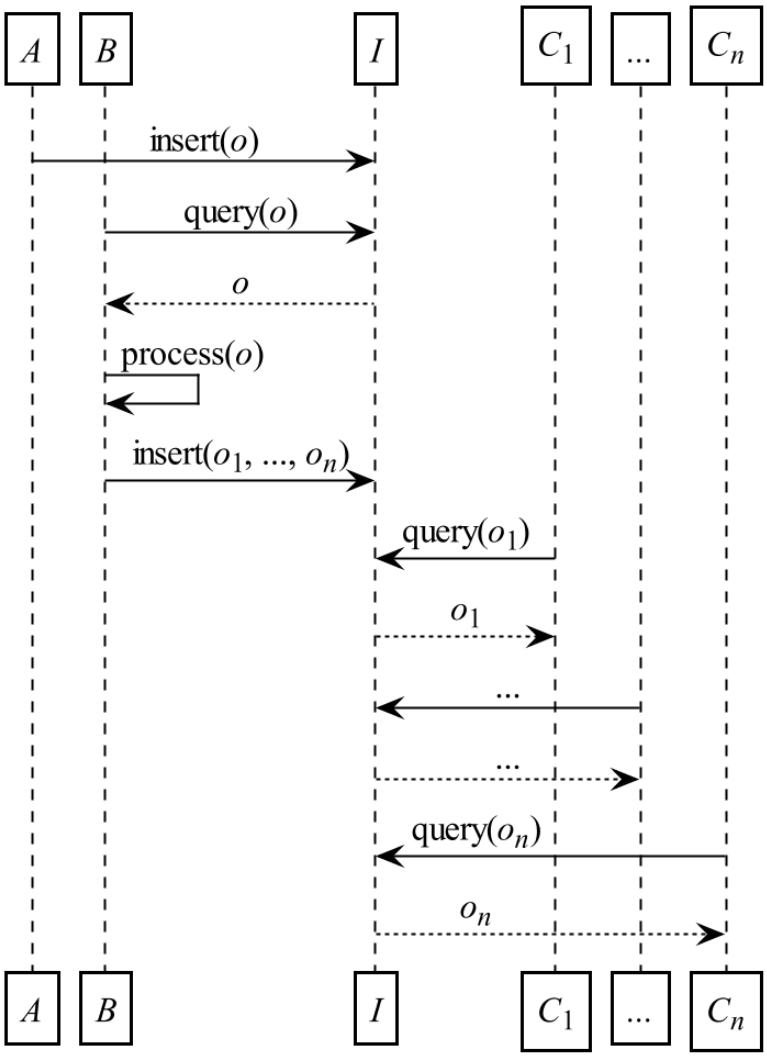


Рисунок 3.6. Шаблон взаимодействия «фасад»

Другим вариантом взаимодействия является взаимодействие «многие-к-одному», которое представлено в шаблоне «одиночка» (рисунок 3.7). При таком взаимодействии агент B получает множество различных индивидов $o_1, \ldots, o_n$ от множества агентов-отправителей $A_1, \ldots, A_n$ и преобразует их в одного индивида o, который передается агенту-получателю C. Агенты $A_1, \ldots, A_n$ могут публиковать информацию, используя как одинаковую, так и разные онтологии. В таком случае агент B является концентратором и на основе полученной информации он создает новую согласно онтологии агента C. Данный шаблон может применяться для сбора разнородной информации из различных источников, ее обработки и последующей доставки конечному пользователю. В частности,

данный шаблон может применяться для разработки системы отслеживания событий в ИП [69].

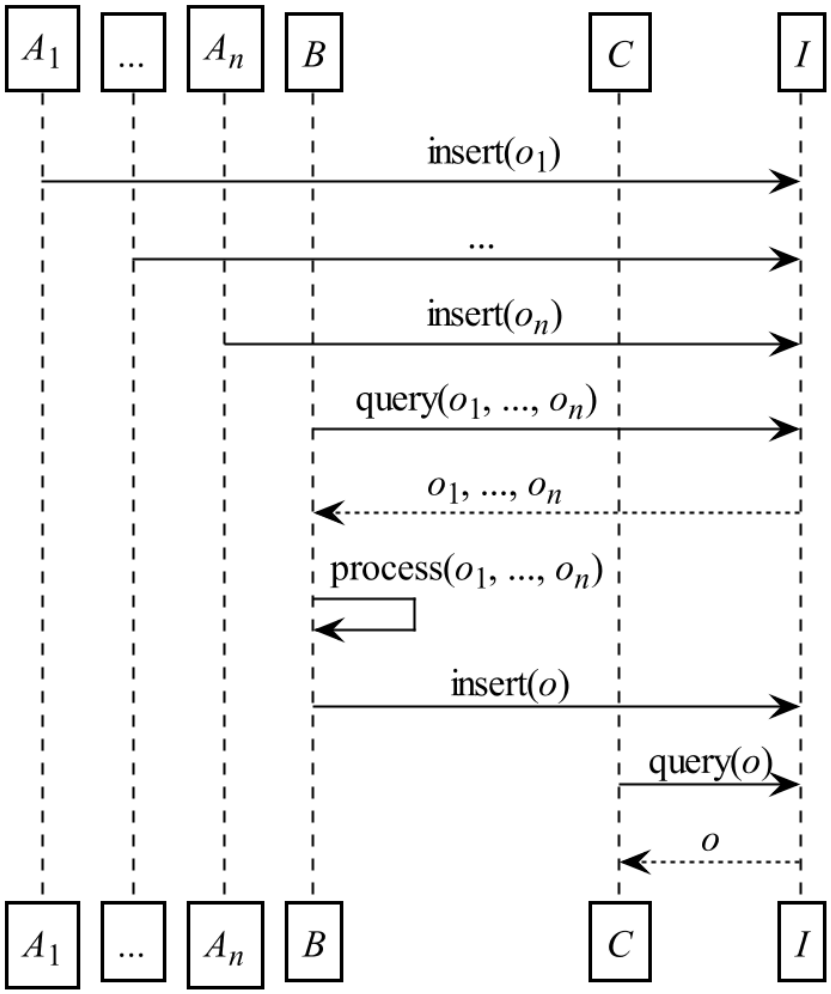


Рисунок 3.7. Шаблон взаимодействия «одиночка»

Следует отметить, что представленные шаблоны образуют иерархическую систему, где каждый последующий шаблон основывается на одном из предыдущих, как показано на рисунке 3.8.

При проектировании информационного сервиса представленные шаблоны могут комбинироваться, образуя более сложные цепочки взаимодействия. В итоге в ИП формируются фрагменты важной для пользователя в данный момент информации, допускающие визуализацию пользовательским агентом для восприятия конечным пользователем.

Шаблоны взаимодействия могут использоваться для автоматической генерации программного кода агентов, реализующих информационный сервис (ри-

Рисунок 3.8. Дерево шаблонов

сунок 3.9). При проектировании прикладной разработчик определяет между какими агентами какое требуется взаимодействие. Для каждого варианта взаимодействия выбирается подходящий шаблон. По шаблону взаимодействия создаются шаблоны программного кода для всех агентов. Шаблон кода состоит из двух частей: а) реализация собственно взаимодействия и б) локальное состояние. Реализация взаимодействия выполняется в терминах операций сетевого доступа к содержимому информационного хранилища. Локальное состояние определяется структурами данных для локального хранения агентом информации и возможными операциями по локальной обработке этой информации. Таким образом, прикладной разработчик концентрируется на реализации в агентах алгоритмов предметной области приложения, а не на реализации взаимодействия агентов в ИП.

Рассмотрим применение шаблона последовательного взаимодействия с созданием нового индивида на примере приложения M3-Weather [112]. Оно предназначено для отображения прогноза погоды относительно текущего местопо-

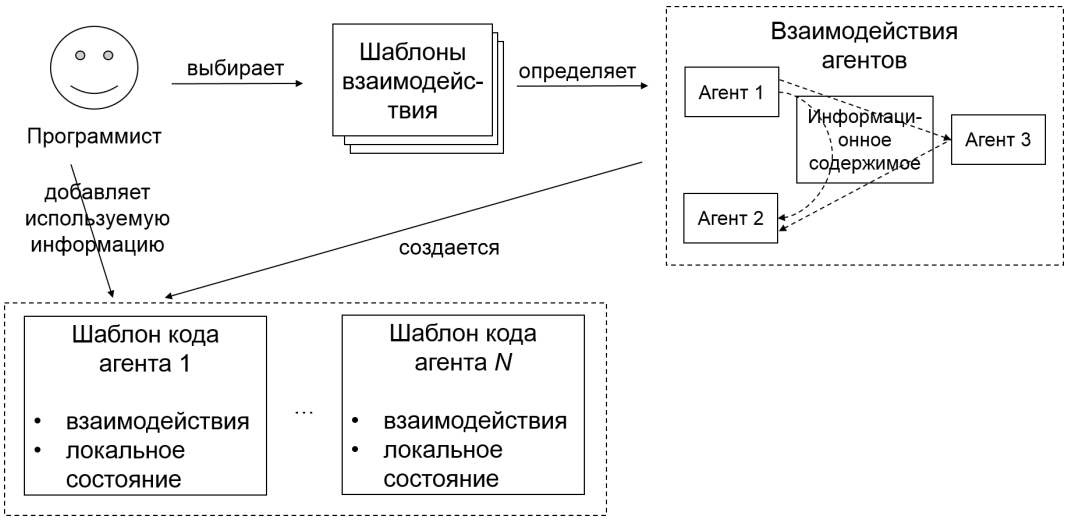


Рисунок 3.9. Применение шаблонов взаимодействия для генерации программного кода агентов

ложения пользователя. Выделяется три вида агентов: клиентский агент, агент местоположения и агент погоды (рисунок 3.10). Для получения прогноза погоды клиентский агент публикует текущие координаты устройства пользователя. На основе этих координат агент местоположения определяет ближайший населенный пункт. Затем, агент погоды по названию населенного пункта запрашивает и публикует прогноз погоды, который затем доставляется клиентскому агенту. Для реализации таких взаимодействий между тремя агентами подходит шаблон последовательного взаимодействия с созданием нового индивида. На рисунке. 3.10 представлены шаблоны взаимодействия и кода для приложения M3-Weather. В качестве локального состояния агентов используются координаты, название города и прогноз погоды.

Пример применения шаблона последовательного взаимодействия с обновлением индивида предоставляется программным приложением туристического информационного сервиса планирования путешествий [97]. Данный сервис позволяет спланировать маршрут путешествия по заданным пользователем точкам маршрута. Выделяется три вида агента: пользовательский агент, транспортный агент и агент для планирования расписания. На рисунке 3.11 представлены шаблоны взаимодействия и кода агентов сервиса для получения расписания

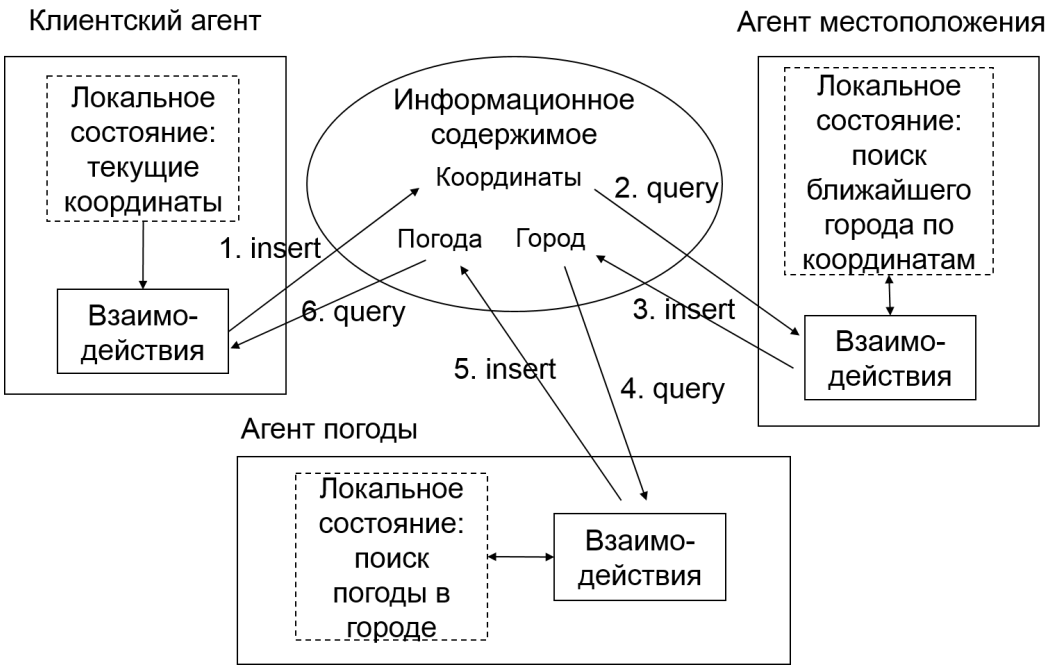


Рисунок 3.10. Использование шаблона последовательного взаимодействия на примере приложения M3-Weather

маршрута.

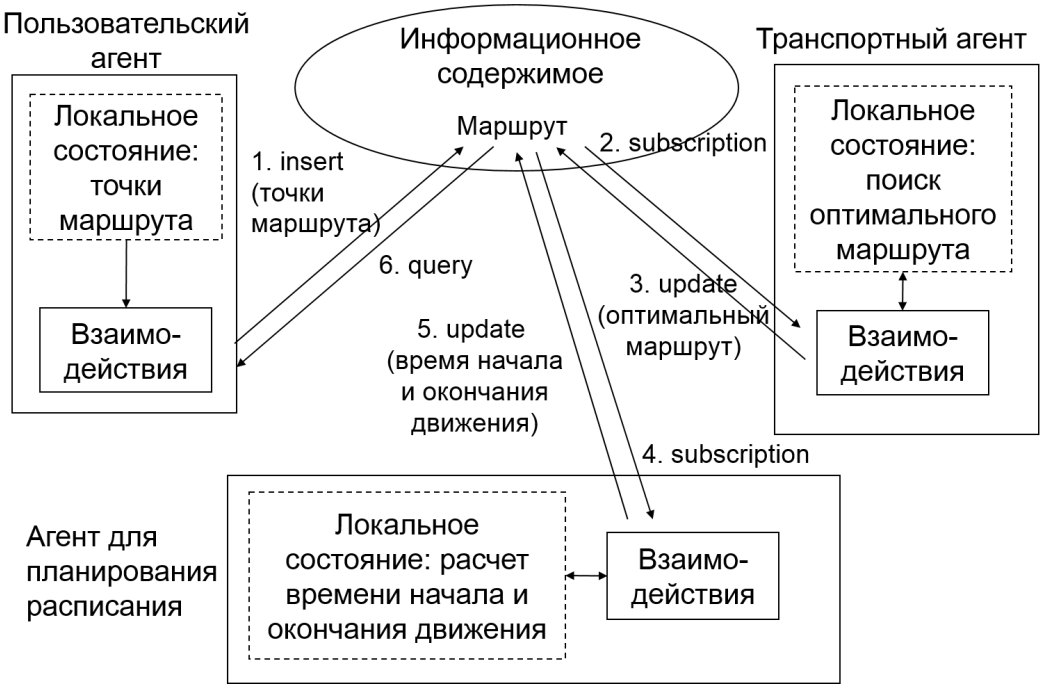


Рисунок 3.11. Использование шаблона последовательного взаимодействия на примере сервиса планирования путешествий

Пользователь с помощью пользовательского агента определяет целевые пункты путешествия (точки маршрута) и публикует эти данные в индивиде

онтологического класса, описывающего маршрут. Транспортный агент формирует оптимальный маршрут обхода целевых пунктов путешествия и обновляет данные индивида класса маршрут. Затем агент для планирования расписания на основании расписания транспорта рассчитывает время начала и окончания движения между точками маршрута. В результате, информационный сервис планирования поездки создает график, основанный на информации о маршруте, опубликованной пользователем. С помощью агента-посредника сервис планирования путешествий может быть интегрирован в систему проведения мероприятий совместной деятельности [22].

Шаблоны взаимодействия описывают типичные ситуации взаимодействия агентов при построении информационного сервиса. В процессе взаимодействия осуществляется передача информации между агентами. В следующем разделе представлена модель информационных уведомлений, описывающая каким образом возможно описать и организовать передачу информации между агентами в процессе взаимодействия.

3.2. Модель информационных уведомлений для программирования косвенного взаимодействия агентов

В условиях разнообразия платформ для развертывания ИП в IoT-среде существует множество различных способов программирования косвенного взаимодействия агентов. Онтологическая модель информационных уведомлений предлагает унифицированный подход для организации косвенного взаимодействия программных агентов для ИП с семантическим представлением содержимого информационного хранилища.

Рассмотрим организацию взаимодействия (3.1) на основе операции подписки. Результатом взаимодействия агентов A_{snd} и A_{rcv} является передача от агента A_{snd} факта о наступлении события для вызова определенной реакции на стороне агента A_{rcv}. Такой вариант может включать и передачу необходи-

мых данных для выполнения действий (напр., для построения сервиса). При реализации агентов A_{snd} и A_{rcv} разработчику необходимо выделить в онтологии приложения (информационного сервиса) свойства, отвечающие за каждый используемый вариант взаимодействия (3.1). На эти свойства подписывается агент A_{rcv}. При их изменении в I агент A_{rcv} получает обновленные значения и соответствующим образом реагирует. В онтологии предметной области подходящие свойства не всегда присутствуют явно, и требуется вводить дополнительные свойства, при изменении которых инициировалось бы взаимодействие.

В настоящее время нет общего подхода к выбору и добавлению таких свойств. Кроме того, при проектировании сервиса необходимо решать задачу о передаче обновленных значений свойств, которые определяют исходные данные для выполнения действий агентом A_{rcv}. Рассмотрим пример на рисунке 3.12. Пусть агент A_{snd} изменяет часть свойств (обновляет) некоторого индивида. В базовой операции подписки (например, в реализации информационного хранилища в платформе Smart-M3 [77, 103]) оповещение агенту A_{rcv} об изменениях (а соответственно и их обработка) происходит отдельно для каждого обновленного свойства. Однако, агенту A_{rcv} необходимо получить индивида целиком со всеми его обновленными свойствами. Определение момента завершения множественного изменения индивида в I является нетривиальной задачей. Дополнительные сложности появляются, когда требуется организовать взаимодействие с подпиской на свойства сразу нескольких индивидов. Для этого требуется заводить большее количество подписок, что повышает требования к ресурсам СВУ.

В итоге, требуемое решение задачи организации взаимодействия агентов на основе операции подписки должно обладать следующими свойствами: 1) инициирование выполнения действий для каждого требуемого в информационном сервисе варианта взаимодействия, 2) возможность передачи всех исходных данных для выполнения действий, 3) общий подход к описанию в онтологии предметной области сервиса свойств для организации взаимодействия, 4) расширяемость при добавлении новых вариантов взаимодействия в информационный

Рисунок 3.12. Пример подписки агента A_{rcv} на несколько онтологических свойств одного или более индивидов

сервис. Для организации косвенного взаимодействия агентов сервиса в работе предлагается онтологическая модель информационных уведомлений.

Онтологическая модель информационных уведомлений программных агентов в ИП определяет (рисунок 3.13): а) способ описания (представления) онтологии системы уведомлений — дополнительных онтологических классов и свойств для организации взаимодействия между двумя или более агентами на основе операции подписки и б) схему использования этого онтологического описания при программировании взаимодействия в агентах. За счет онтологического описания становится возможным проблемно-ориентированные проектирование и программирование взаимодействия на уровне каждого отдельного агента сервиса.

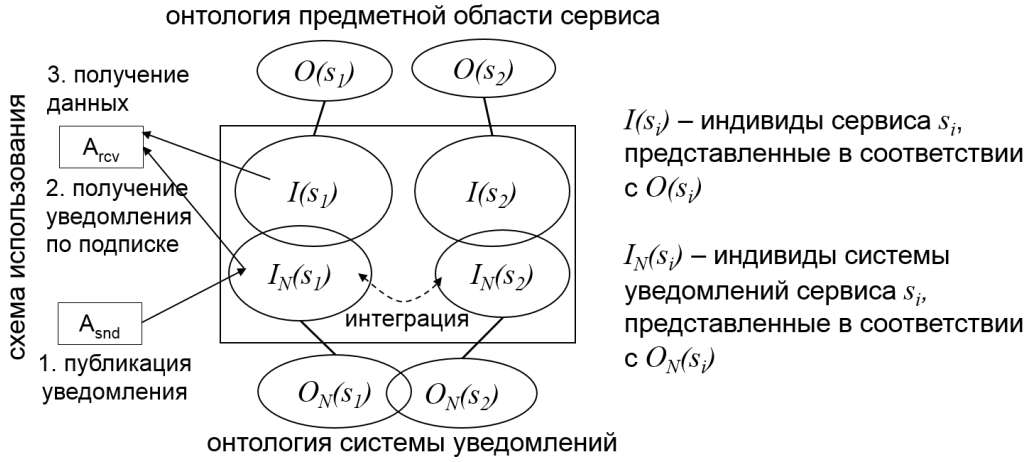


Рисунок 3.13. Онтологическая модель информационных уведомлений

Определим уведомление x как информационное сообщение для организа-

ции взаимодействия, отправляемое агентом A_{snd} при выполнении определенного условия e и получаемое агентом A_{rcv}. Уведомление содержит исходные данные — параметры уведомления. Каждое уведомление соответствует отдельному варианту взаимодействия в информационном сервисе. Прикладной разработчик при проектировании сервиса должен определить список уведомлений, отражающих все требуемые варианты взаимодействия агентов.

Пусть m — общее количество агентов в информационном сервисе, а M — общее количество вариантов взаимодействий между всеми агентами. Рассмотрим множество вариантов взаимодействия $N_k = \langle i, j, P_k \rangle \, k = \overline{1, M}$ между агентами i и j $(i, j = \overline{1, m})$ и множеством параметров взаимодействия $P_k = \{p_i\}_{i=1}^{n_k}$, где n_k — количество параметров взаимодействия — сколько индивидов классов или атрибутов передается при взаимодействии. При отсутствии взаимодействия между агентами $P_k = \varnothing$. Взаимодействием является передача информации, представленной параметрами взаимодействия P_k, от агента-отправителя i (A_{snd}) агенту-получателю j (A_{rcv}).

Онтологическая модель информационных уведомлений может быть представлена как $(\sigma(N_k), O_N, Q(N_k))$, где σ — условие, определяющее инициацию взаимодействия N_k, O_N — онтология системы уведомлений, $Q(N_k)$ — реакция агента на взаимодействие N_k. Инициация взаимодействия N_k может быть осуществлена: а) по запросу (действие пользователя), б) по событию (при изменении информации в I). Реакция агента $Q(N_k)$ включает множество действий, выполняемых агентом после получения уведомления.

Пусть $O_S =< C_S, A_S, D_S, R_S >$ — онтология информационного сервиса, где C_S — множество классов онтологии сервиса, A_S — множество атрибутов классов, D_S — множество доменов — областей допустимых значений атрибутов, R_S — множество отношений (ограничений), которыми связаны классы. Отношениями описываются: таксономия классов (отношение «быть экземпляром»), иерархия классов («быть частью»), наследование, свойства классов и т.д.

Онтология системы уведомлений $O_N =< C_N, A_N, D_N, R_N >$, где $C_N =$

$\{c_1, \ldots, c_M, c_{\text{ex}}\} \cup C'_S$; $c_1, \ldots, c_M$ — множество классов системы уведомлений, каждый из которых соответствует отдельному варианту взаимодействия, c_{ex} — дополнительный класс, используемый для хранения параметров уведомления при $|P_k| > 1$, $C'_S = \{b_1, \ldots, b_h\} \subset C_S$ — множество классов из онтологии сервиса, используемых для передачи информации в параметрах уведомления при взаимодействии, $A_N = a_1, \ldots, a_z$ — множество атрибутов, используемых для передачи текстовых параметров уведомления, D_N — множество доменов — областей допустимых значений атрибутов, R_N — множество отношений (ограничений), которыми связаны классы. Тогда уведомление — это индивид класса из C_N, представляющий отдельный акт взаимодействия между A_{snd} и A_{rcv}.

Тогда уведомление с одним параметром (простое уведомление) $n_k = 1$ будет иметь вид:

$$c_i p_i b_i$$

$$c_i p_i a_i$$

Уведомление с несколькими параметрами $n_k > 1$ (сложное уведомление):

$$c_i p_i c_{\text{ex}} p_j b_j, \forall j : p_j \in P_k$$

$$c_i p_i c_{\text{ex}} p_j a_j$$

Онтологическая модель информационных уведомлений предназначена для организации следующих двух типов взаимодействия:

- запрос: агент A_{rcv} осуществляет требуемые действия, используя исходные данные от агента A_{snd};
- событие: агент A_{rcv} реагирует на определенное информационное событие, о котором информирует агент A_{snd}.

Они определяют две функциональные точки зрения на агента A_{rcv}: обработчик данных или реагирующий модуль. Первая точка зрения ближе к процедурному программированию, тогда как вторая соответствует событийно-ориентированному программированию.

Рассмотрим двунаправленное расширение взаимодействия (3.1):

$$A_{\text{snd}} \leftrightarrow A_{\text{rcv}}, \tag{3.2}$$

Во взаимодействии (3.2) могут участвовать несколько агентов A_{snd} и несколько агентов A_{rcv}. Схема использования уведомлений при организации взаимодействия (3.2) состоит из нижеперечисленных шагов (рисунок 3.14), программируемых в агентах A_{snd} и A_{rcv}. Перед выполнением этих шагов агенты должны подписаться на уведомления.

1. При выполнении определенного условия e агенту A_{snd} необходимо инициировать выполнение действий некоторым агентом A_{rcv}. Тогда A_{snd} формирует уведомление x с n параметрами и публикует его в I.

2. Агент A_{rcv} обнаруживает факт публикации уведомления по подписке и получает уведомление x из I.

3. Агент A_{rcv} обрабатывает уведомление x:

   - получает значения параметров уведомления;
   - выполняет действия, требуемые данным вариантом взаимодействия, используя исходные данные из полученного уведомления;
   - удаляет уведомление x (если тип взаимодействия — запрос).

4. Агент A_{rcv} формирует и публикует в I ответное уведомление y с результатами выполнения своих действий.

5. Агент A_{snd} получает ответное уведомление y с помощью операции подписки, реагирует и удаляет уведомление y.

Шаги 1–3 являются обязательными для взаимодействия (3.2). Шаги 4–5 реализуют обратную связь и могут быть пропущены, если это не требуется заданным вариантом взаимодействия в информационном сервисе.

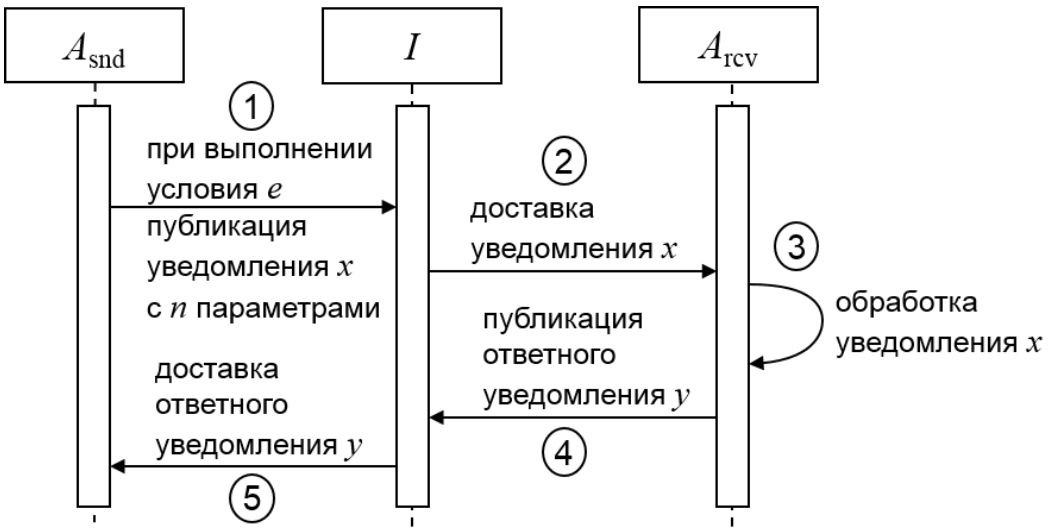


Рисунок 3.14. Схема использования уведомлений агентами A_{snd} и A_{rcv}

При разработке каждого агента разработчик должен учитывать следующие свойства уведомлений: представление (форма уведомления), момент инициации (отправки), тип взаимодействия (запрос или событие), наличие ответного уведомления и момент удаления. Отдельного внимания требует и вопрос производительности из-за использования ресурсоемкой операции подписки.

Представление: простое уведомление x (индивид уведомления) содержит только один параметр и состоит из двух RDF-троек:

$$\langle x_{\text{id}}, \text{rdf:type}, c(x) \rangle$$
$$\langle x_{\text{id}}, p, v \rangle$$

Здесь x_{id} — идентификатор индивида уведомления x, rdf:type — стандартный предикат RDF, описывающий тип индивида, $c(x) \in C_N$ — класс из онтологии, описывающий уведомление, p — свойство из онтологии, описывающее название уведомления (или его параметра), v — значение параметра (текстовые данные или идентификатор индивида из онтологии предметной области сервиса). Параметр содержит данные, которые необходимо передать получателю уведомления. Онтологический класс может содержать несколько свойств, т.е. описывать несколько различных уведомлений.

Составное уведомление состоит из набора RDF-троек для представления

передаваемых параметров:

$$\langle x_{\text{id}}, \text{rdf:type}, c(x) \rangle$$
$$\langle x_{\text{id}}, p, n_{\text{id}} \rangle$$
$$\langle n_{\text{id}}, \text{rdf:type}, c_{\text{ex}} \rangle$$
$$\langle n_{\text{id}}, p_1, v_1 \rangle$$
$$...$$
$$\langle n_{\text{id}}, p_n, v_n \rangle$$

Здесь p — название уведомления (вид запроса или события), n_{id} — идентификатор дополнительного индивида, хранящего все параметры уведомления, $c_{\text{ex}} \in C_N$ — онтологический класс, описывающий дополнительного индивида, содержащего параметры уведомления, p_i — название i-го параметра уведомления, v_i — значение i-го параметра.

При таком представлении онтология уведомлений $O_N(s)$ является расширением онтологии предметной области сервиса $O(s)$. Каждое уведомление имеет один или более параметров, значение каждого хранится как объект RDF-тройки уведомления. Значением параметра могут быть строковые данные (свойство-значение) или идентификатор индивида из I (объектное свойство). Индивиды уведомлений $I_N(s)$ могут быть связаны с индивидами из предметной области сервиса $I(s)$, как это показано на рисунке 3.15. Тем самым, взаимодействие между агентами осуществляется за счет публикации и изменения индивидов $I_N(s)$ онтологии системы уведомлений $O_N(s)$, а не индивидов $I(s)$ онтологии предметной области сервиса $O(s)$.

Для выполнения взаимодействия между агентами на основе операции подписки требуется изменение определенных данных в I. Без использования модели информационных уведомлений для передачи информации от A_{rcv} к A_{snd} потребовалось бы непосредственное изменение свойств индивидов из онтологии предметной области сервиса $O(s)$. Однако не во всех предметных областях заданы свойства, изменение которых инициирует взаимодействие между аген-

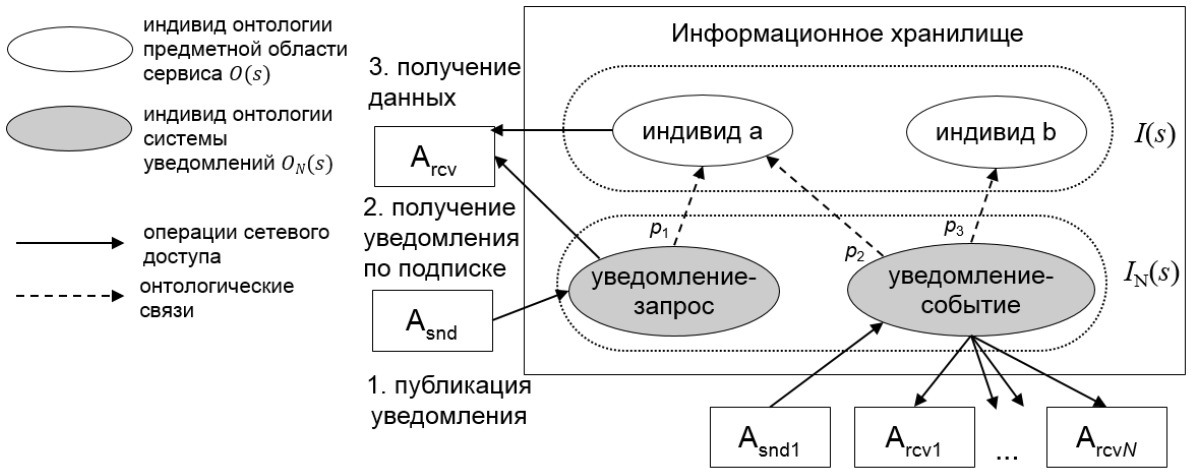


Рисунок 3.15. Связь между уведомлениями и индивидами предметной области сервиса

тами. Например, для передачи текстового сообщения от A_{rcv}к A_{snd} не требуется изменения данных из предметной области приложения, поэтому онтология приложения не содержит в себе классы и свойства для описания таких сообщений. В этом случае разработчику необходимо расширить онтологию $O(s)$. Онтологическая модель информационных уведомлений предлагает общий способ для описания всех таких свойств, которые отвечают за инициирование взаимодействия. Более того, предлагаемые уведомления описывают и формат обмена информацией между агентами в случае, если они используют разные онтологии (что позволяет осуществить интеграцию нескольких сервисов).

Инициация взаимодействия: в зависимости от момента отправки на стороне агента A_{snd} уведомления делятся на реактивные и проактивные (упреждающие). Момент отправки наступает при выполнении условия e, которое определяется разработчиком агента.

Реактивные уведомления соответствуют явной инициации пользователем (команда) на отправку уведомления от агента A_{snd} для выполнения требуемых действий агентом A_{rcv}. В этом случае агент A_{snd}, как правило, ожидает ответного уведомления с результатом выполненных действий.

Проактивное уведомление агент A_{snd} посылает без явной команды от пользователя и, как правило, не ожидает ответного уведомления. Контекст пользователя формирует неявную зависимость от активности пользователя: агент

A_{snd} в фоновом режиме анализирует контекст (параллельно с основной активностью пользователя) и затем осуществляет отправку уведомления. Проактивные уведомления обычно используются при событийно-ориентированном программировании взаимодействия между агентами.

Простейший вариант использования реактивного уведомления: пользователь запускает агента A_{snd} (клиентская программа) для получения информационного сервиса. Сервисный агент A_{rcv} отвечает за построение сервиса. По команде пользователя клиентский агент A_{snd} отправляет реактивное уведомление агенту A_{rcv} и ожидает ответа. Пользователь явно вовлечен в процесс инициации взаимодействия.

Простейший вариант использования проактивного уведомления: сервисный агент A_{rcv} выполняет рекомендательную функцию (напр., анализирует данные, с которыми работает пользователь, и предлагает сходные данные из других источников). На клиентском агенте A_{snd} пользователь просматривает информацию в рамках своей основной деятельности. Параллельно, агент A_{snd} автоматически посылает уведомления, описывающие контекст работы пользователя. Сервисный агент A_{rcv} реагирует и формирует рекомендации, которые уже явно получает пользователь. Отметим, что сервисный агент A_{rcv} может быть отключен, а приложение продолжит работу (с ограниченным набором сервисов).

Тип взаимодействия: по запросу или по событию. Уведомление-запрос используется для выполнения определенного действия другим агентом (один получатель). В предыдущих рассмотренных примерах использовалось уведомление-запрос: клиентский агент A_{snd} запрашивал (реактивно или проактивно) сервисный агент A_{rcv}. Уведомление-событие позволяет информировать других агентов (получателей может быть несколько) о наступлении события. Например, уведомление посылается, когда пользователь читает, а факт чтения может быть использован для приостановки формирования рекомендаций сервисным агентом A_{rcv} (т.е. реализуется, что чтение — это непрерываемая деятельность).

Ответное уведомление: в зависимости от цели взаимодействия, агент A_{snd} ожидает или нет ответного уведомления с результатом выполнения запрошенных действий.

Удаление: уведомление-запрос удаляется из I получателем A_{rcv}, когда он выполнит необходимые действия. Уведомление-событие удаляется отправителем A_{snd}, который сам определяет длительность нахождения уведомления в I.

Производительность: операция подписки является ресурсоемкой операцией, поэтому каждому агенту рекомендуется ограничивать число одновременно работающих подписок. Модель информационных уведомлений позволяет реализовать обработку всех уведомлений для заданного варианта взаимодействия с помощью одной подписки на следующие RDF тройки-шаблоны:

$$\langle x_{\text{id}}, *, * \rangle; \tag{3.3}$$

или

$$\langle *, \text{rdf:type}, c(x) \rangle; \tag{3.4}$$

где маска $*$ представляет любое значение, x_{id} — идентификатор уведомления x, $c(x)$ — класс онтологии, описывающий уведомление x. Подписка на идентификатор индивида (3.3) происходит, когда агенту соответствует уникальный идентификатор индивида уведомления. Подписка срабатывает при изменении свойств соответствующего индивида уведомления. Подписка на онтологический класс (3.4) срабатывает при появлении или удалении индивида уведомления в I, а не при изменении свойств этого индивида. Такой вариант не привязан к идентификатору определенного индивида уведомления.

Подписка (3.3) может использоваться в небольших сервисах, где достаточно завести один онтологический класс для уведомлений. Подписка (3.4) предназначена для сервисов, в которых каждый класс онтологии соответствует отдельному типу агента (со всеми соответствующими ему вариантами взаимодействия). Также, в (3.3), из-за подписки на фиксированный идентификатор уведомления (субъект тройки), невозможна отправка двух одинаковых уведом-

лений (т.к. предикат и объект тройки будут теми же самыми). Таким образом, требуется обработка и удаление из I уведомления, чтобы была возможность отправить следующее. В (3.4) такого ограничения нет, поскольку индивиды уведомления имеют различные идентификаторы (субъекты тройки).

После получения основной тройки уведомления при подписке агент должен получить все параметры уведомления. В разделе 4.3 рассматривается влияние количества параметров на производительность агента A_{rcv} при обработке уведомления.

Онтологическая модель информационных уведомлений предоставляет способ единообразного описания взаимодействия агентов при разработке информационного сервиса в ИП. Данный подход снижает трудоемкость реализации взаимодействия агентов, а также позволяет уменьшить количество используемых агентами подписок. Кроме этого модель позволяет осуществить интеграцию других сервисов в ИП (см. раздел 3.3 далее).

3.3. Программирование взаимодействия агентов на основе онтологического событийно-ориентированного описания взаимодействий

Предложенная модель информационных уведомлений предоставляет проблемно-ориентированный способ для организации взаимодействия агентов, применимый для широкого круга информационных сервисов ИП. В качестве примера использования модели информационных уведомлений рассмотрим систему SmartScribo [92]. Система SmartScribo реализует сервис мобильного мультиблоггинга и позволяет пользователям с персональных мобильных компьютеров взаимодействовать одновременно с несколькими блогами на внешних блог-сервисах. В I хранится персональная информация пользователя и данные о его блогах. Архитектура системы представлена на рисунке 3.16. В системе выделяется три вида агентов: клиенты, блог-процессоры и блог-медиаторы. Клиент

реализует интерфейс пользователя и позволяет ему работать с своими блогами. Блог-процессор отвечает за доступ к определенному блог-сервису. Для каждого подключаемого к системе блог-сервиса используется отдельный блог-процессор. Блог-медиатор выполняет дополнительный анализ и обработку текущего содержимого информационного хранилища (напр., персонализированная рекомендация блогов для просмотра).

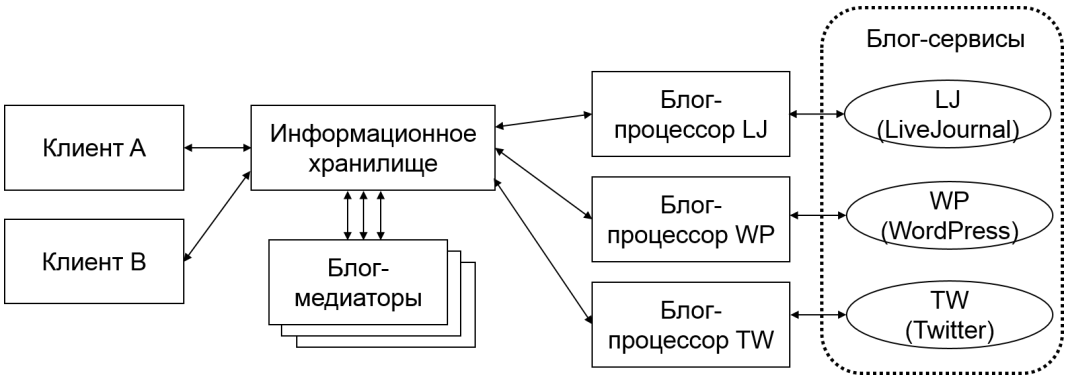


Рисунок 3.16. Архитектура системы SmartScribo

Взаимодействие пользователя с блогами организована следующим образом. С помощью клиентского агента пользователь выбирает блоги для подключения. Блог-процессоры загружают посты с выбранных блогов. Затем пользователь может читать/изменять существующие посты или создавать новые. При создании нового поста или изменении существующего, блог-процессоры отражают изменения на блог-сервисах.

В информационном хранилище может храниться информация о множестве постов и блогов различных пользователей. При использовании подписки на отдельные свойства индивидов постов, блог-процессор должен определить, какие посты были обновлены, а также когда обновленный индивид поста должен быть отправлен на блог-сервис. Например, у индивида поста с идентификатором p_{id} есть такие свойства, как заголовок (title), текст, ярлыки (теги). Тогда для получения изменений заголовка блог-процессор может подписаться на тройку-шаблон: $\langle p_{\text{id}}, \text{title}, * \rangle$ и, аналогично, для других свойств поста. Также есть возможность подписаться на тройку-шаблон с получением изменений всех

свойств (предикатов) поста: $\langle p_{\text{id}}, *, * \rangle$. Однако, в обоих случаях агент будет получать обновления по подписке отдельно для каждого измененного свойства и момент завершения изменения поста не определен явным образом.

Кроме того, посты динамически создаются и удаляются в I. Каждый раз устанавливать и снимать подписку на появившийся пост является неэффективным. Блог-процессор может подписаться на свойства, относящиеся к любому индивиду поста, например, $\langle *, \text{title}, * \rangle$, однако, подписка все равно будет информировать об изменении каждого свойства отдельно. В этом случае обновления будут относиться к различным индивидам поста. Для решения указанных проблем можно использовать модель информационных уведомлений. С помощью модели возможно организовать информирование блог-процессора о необходимости выполнить требуемое действие с определенным индивидом поста.

Список уведомлений, используемых для организации взаимодействия между клиентом и блог-процессором в системе SmartScribo, представлен в таблице 3.1. Все уведомления являются реактивными уведомлениями-запросами, т.к. отправляются после действий пользователя и требуют выполнения операций от блог-процессора. Каждое уведомление требует ответного уведомления, которое информирует пользователя о результате выполнения операции. В качестве параметров уведомлений используются идентификаторы индивидов: субъект тройки, описывающий определенную учетную запись, пост или комментарий.

Рассмотрим простое уведомление «refreshPosts». Оно имеет вид:

$$\langle \text{NTF-}s, \text{rdf:type}, \text{NTF}_{\text{class}} \rangle$$
$$\langle \text{NTF-}s, \text{refreshPosts}, a_{\text{id}} \rangle$$

Здесь NTF — постоянная часть идентификатора индивида уведомления, $\text{NTF}_{\text{class}}$ — класс индивида уведомления, s — тип блог сервиса (например «LJ» для LiveJournal), a_{id} — идентификатор индивида учетной записи. Различные идентификаторы сервисов используются для распределения уведомлений между различными блог-процессорами. Согласно варианту подписки (3.3) каждый блог-

Таблица 3.1. Список уведомлений в системе SmartScribo

Уведомление (вариант взаимодействия)	Параметры	Описание
refreshAccount	учетная запись	получить информацию об учетной записи блога
refreshPosts	учетная запись	получить посты заданной учетной записи
sendPost	учетная запись, пост	отправить пост
editPost	прошлый пост, новый пост	заменить прошлый пост на новый
delPost	учетная запись, пост	удалить пост
refreshComments	учетная запись	получить комментарии всех постов учетной записи
sendComment	учетная запись, комментарий, родительский объект	отправить комментарий на родительский объект (пост или комментарий)
delComment	учетная запись, комментарий, родительский объект	удалить комментарий у родительского объекта (поста или комментария)

процессор подписывается на шаблон тройки, субъект которой — идентификатор сервиса, а предикат и объект — любые значения. Для блог-процессора, работающего с сервисом LiveJournal, шаблон тройки имеет вид $\langle \text{NTF-LJ}, *, * \rangle$.

Пользователь SmartScribo указывает учетные записи для блогов с помощью клиента. Затем он может обновить список постов для каждой учетной записи: клиент публикует индивида учетной записи со всеми необходимыми для доступа к блог-сервису свойствами и публикует уведомление «refreshPosts». Блог-процессор получает уведомление по подписке, извлекает данные об учетной записи из I, получает список постов с блог-сервиса и удаляет уведомление. Затем блог-процессор публикует список постов в I и отправляет ответное уведомление для клиента об удачном завершении операции.

Рассмотрим сложное уведомление «sendPost». Оно имеет вид:

$$\langle \text{NTF-}s, \text{rdf:type}, \text{NTF}_{\text{class}} \rangle$$
$$\langle \text{NTF-}s, \text{sendPost}, n_{\text{id}} \rangle$$
$$\langle n_{\text{id}}, \text{rdf:type}, \text{NP}_{\text{class}} \rangle$$
$$\langle n_{\text{id}}, \text{postAcc}, a_{\text{id}} \rangle$$
$$\langle n_{\text{id}}, \text{postId}, p_{\text{id}} \rangle$$

Здесь s — тип блог-сервиса, n_{id} — идентификатор дополнительного индивида уведомления, хранящего параметры уведомления, NP_{class} — класс индивида с параметрами уведомления, a_{id} — идентификатор индивида учетной записи, p_{id} — идентификатор индивида поста. Пользователь (на клиенте) создает новый пост и публикует его и уведомление в I. Блог-процессор получает уведомление по подписке, извлекает идентификатор индивидов учетной записи и поста, получает свойства поста по идентификатору поста и отправляет пост на внешний блог-сервис. После этого блог-процессор удаляет обработанное уведомление и публикует ответное уведомление для клиента.

В качестве другого примера рассмотрим использование уведомлений в системе проведения мероприятий совместной деятельности (рисунок 3.17). Для на-

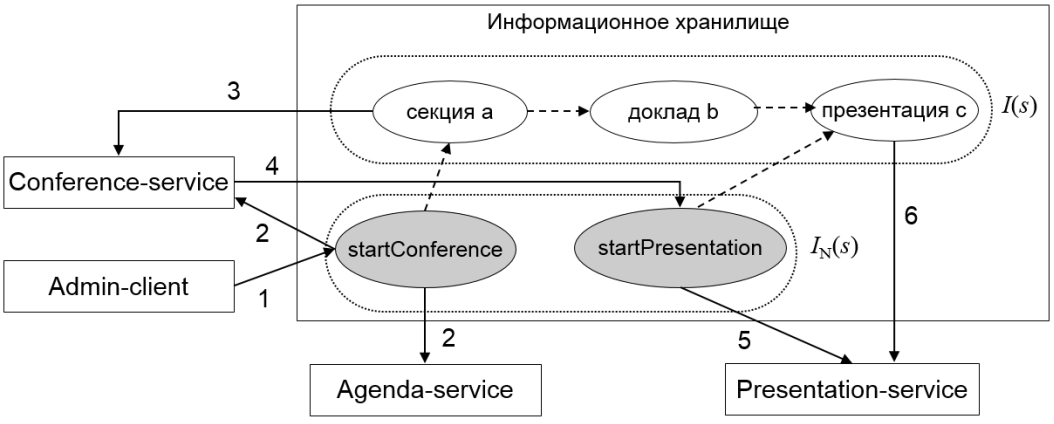


Рисунок 3.17. Использование уведомлений в системе проведения мероприятий совместной деятельности

чала мероприятия организатор с помощью мобильного клиента (Admin-client) публикует уведомление «startConference» о начале мероприятия. Данное уведомление получает агент сервиса проведения конференции (Conference-service),

из параметров уведомления получается информацию о предстоящем докладе и публикует уведомление «startPresentation» о необходимости начала показа презентации. Данное уведомление получается агентом Presentation-service, отвечающим за показ презентаций, который получает из параметров уведомления необходимую информацию о презентации и запускает ее показ. Таким образом с помощью уведомлений организуется взаимодействие в системе проведения мероприятий совместной деятельности.

Модель информационных уведомлений предоставляет возможность интеграции сервисов за счет того, что для интеграции одного сервиса в другой необходимо иметь представление только об онтологии уведомлений интегрируемого сервиса (и связанных с ней классов для передачи параметров уведомления). Уведомления, используемые сервисом, являются интерфейсом для взаимодействия с ним. Таким образом, зная интерфейс для взаимодействия с интегрируемым сервисом исходный сервис может использовать его функционал для собственных целей. Реализация интеграции сервисов может быть выполнена в каком-либо из существующих агентов исходного сервиса, либо в отдельном агенте-посреднике.

Рассмотрим применение шаблона агента-посредника для интеграции информационных сервисов на примере интеграции систем проведения мероприятий совместной деятельности и системы SmartScribo [85, 89]. Агент-посредник на основе шаблона посредник интегрирует две системы, расширяя возможности проведения конференции дискуссиями с использованием известных блог-сервисов социальных сетей. Для каждого выступления докладчика на конференции агент-посредник создает посты в блоге, в которых разворачивается дискуссия для данного выступления.

В приведенном примере агент-посредник преобразует информацию только в одном направлении. В то же время, возможно расширение для выполнения взаимодействия и в обратном направлении. Например, комментарии к посту на блоге, посвященному текущему докладу, могут быть переданы из системы

SmartScribo в систему проведения мероприятий совместной деятельности для отображения на общем экране с расписанием проводимой конференции.

3.4. Выводы

Модель информационных уведомлений позволяет организовать взаимодействие между программными агентами для построения информационного сервиса на основе шаблонов взаимодействия. Шаблоны взаимодействия описывают распространенные виды взаимодействия между агентами и применимы для широкого круга предметных областей. За счет комбинации различных шаблонов взаимодействия возможно проектирование информационного сервиса от его построения до доставки конечному пользователю. На основе шаблонов взаимодействия может быть сгенерирован программный код агентов, отвечающий за реализацию взаимодействия. Модель информационных уведомлений предоставляет универсальный подход для описания и реализации взаимодействия между агентами на основе событийно-ориентированного семантического описания взаимодействий. Модель позволяет единообразным способом организовать взаимодействие агентов и сокращает количество используемых подписок в информационном сервисе, что позволяет сократить нагрузку на информационное хранилище и трудоемкость прикладной программной разработки.

Глава 4

Экспериментальное исследование производительности

В главе представлена программная реализация и экспериментальные исследования полученных в работе результатов, проведена апробация и сделаны выводы об их эффективности при разработке информационных сервисов. Программная реализация информационного хранилища ориентирована на платформу Smart-M3, которая распространяется с открытым исходным кодом, поддерживает косвенное взаимодействие агентов и общее RDF-хранилище. Апробация проводится на основе, разрабатываемых при участии автора, информационного хранилища CuteSIB и системы проведения мероприятий совместной деятельности (интеллектуальный зал ПетрГУ SmartRoom). Исследуется вопрос повышения работоспособности и устойчивости к сбоям информационных сервисов для различных аппаратно-программных вычислительных платформ. Показана практическая эффективность на примере разработки агентов для системы проведения мероприятий совместной деятельности. Результаты, приведенные в данной главе, опубликованы в [68, 7, 6, 92, 95, 121] и подтверждаются свидетельством о государственной регистрации программы для ЭВМ (см. Приложение А) и актами внедрения результатов диссертационной работы (см. Приложение Б). Сводные характеристики разработанного комплекса программных средств представлены в Приложении В.

4.1. Трудоемкость обработки операций в программной реализации информационного хранилища CuteSIB

В ходе экспериментального исследования измерялась трудоемкость обработки операций в информационном хранилище CuteSIB. Данное исследование показывает за какое время какой объем RDF-троек способно обработать информационное хранилище в условиях ресурсно-ограниченной IoT-среды.

Для исследования производительности операций в модели управления сетевым доступом агентов к информационному хранилищу измерялось время выполнения T_o операций $o \in \{\text{insert}, \text{remove}, \text{query}\}$. Операция update является последовательным выполнением операций remove и insert. Операция subscribe основана на операции query. Следовательно, время выполнения этих операций соотносится со временем выполнения измеряемых базовых операций. Время T_o измерялось для трех типов СВУ: серверная ЭВМ (T_o^{srv}), персональная ЭВМ на примере ноутбука (T_o^{lpt}), и встроенная ЭВМ на примере одноплатного компьютера Raspberry Pi (T_o^{emb}). Измерялось время выполнения операции с количеством RDF-троек $10^3 \leq N_{\text{RDF}} \leq 35 \cdot 10^3$. Каждое измерение выполнялось по 50 раз. Для встроенной ЭВМ время выполнения операции при $N_{\text{RDF}} > 25 \cdot 10^3$ троек измерить не удалось из-за долгого времени обработки.

На рисунке 4.1 представлено время выполнения операции insert T_{insert}. Время выполнения на сервере и ноутбуке оказалось приблизительно одинаковым, а на встроенном устройстве в несколько раз больше. Это связано с тем, что встроенное устройство располагает гораздо меньшим количеством ресурсов (более слабый процессор и меньшее количество оперативной памяти). С объемами более $20 \cdot 10^3$ RDF-троек встроенное устройство не может работать, так как не хватает оперативной памяти и начинает использоваться файл подкачки.

На рисунке 4.2 представлено время выполнения операции query. Время выполнения на сервере на несколько сотен миллисекунд быстрее, чем на ноутбуке. Время выполнения операции на встроенном устройстве медленнее, но разница во времени с сервером и ноутбуком не такая большая, как в операции insert. Отсюда можно сделать вывод, что операция query меньше зависит от доступных ресурсов СВУ.

На рисунке 4.3 представлено время выполнения операции remove. Время выполнения на сервере и ноутбуке оказалось приблизительно одинаковым, а на встроенном устройстве в несколько раз больше. Это связано с тем, что встроенное устройство располагает гораздо меньшим количеством ресурсов (более сла-

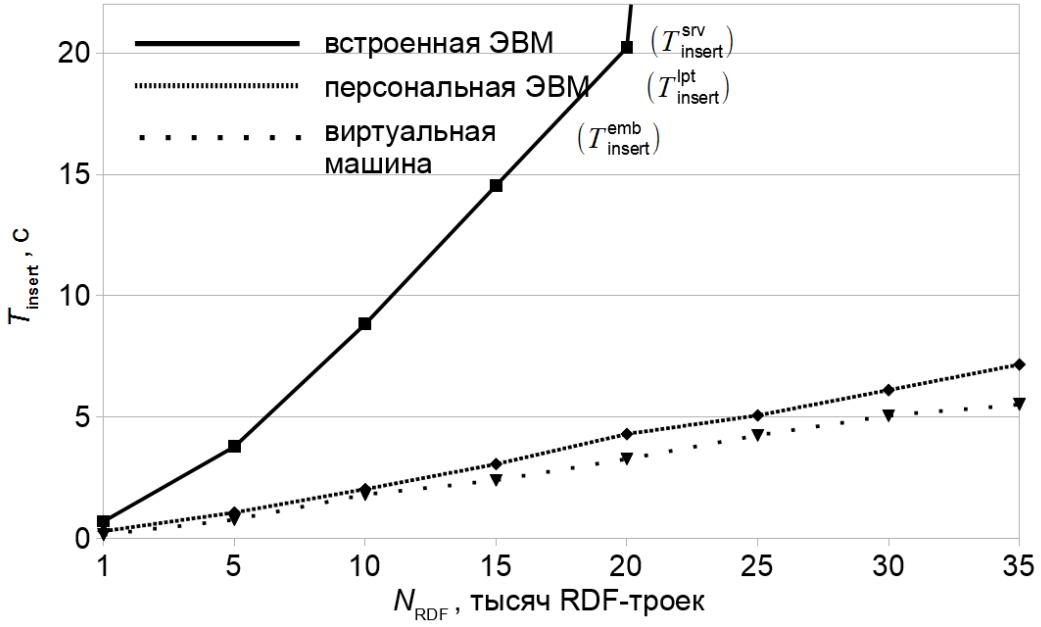


Рисунок 4.1. Время выполнения операции insert

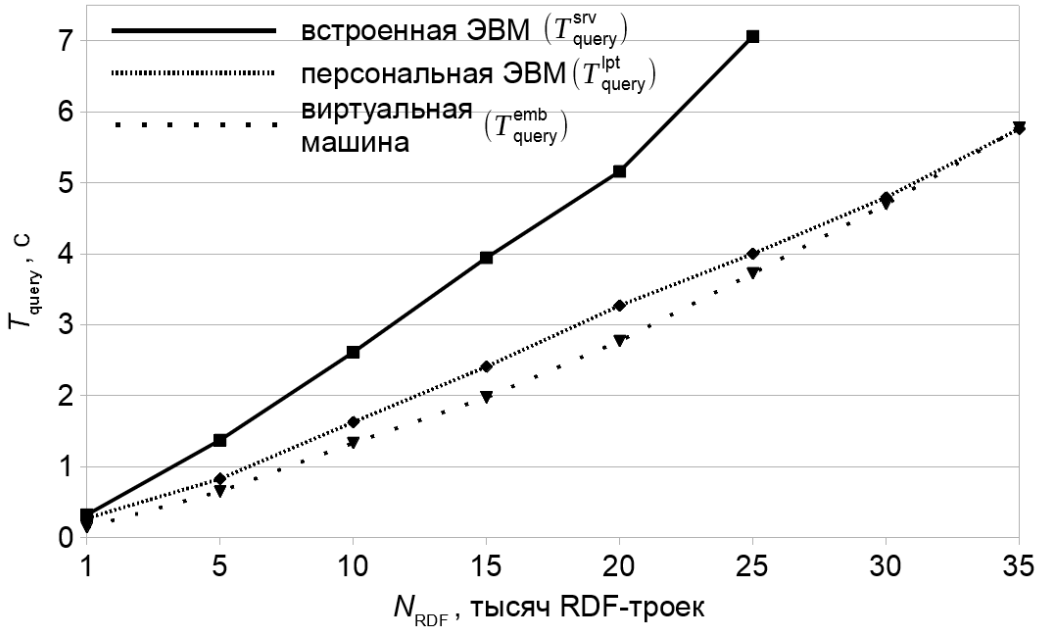


Рисунок 4.2. Время выполнения операции query

бый процессор и меньшее количество оперативной памяти). Операция remove выполнялась быстрее, чем операция insert.

Объем обрабатываемых RDF-троек N_{RDF} представляет в экспериментах ситуацию единоразовой обработки большого количества данных. Для системы проведения мероприятий совместной деятельности такая обработка может осуществляться над определенным количеством участников мероприятия. В соот-

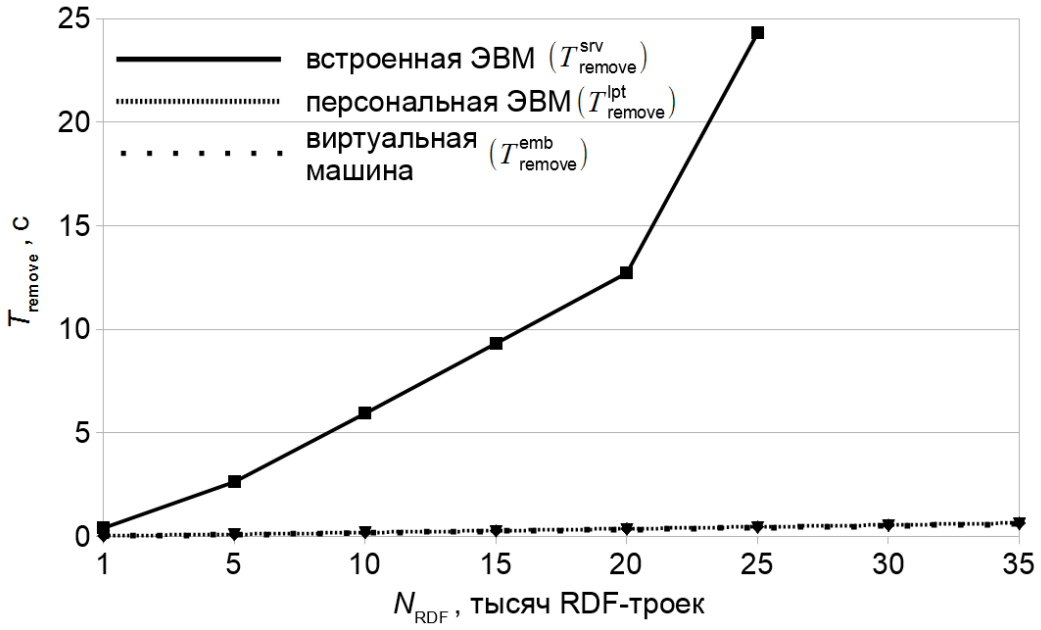


Рисунок 4.3. Время выполнения операции remove

ветствии с онтологией системы проведения мероприятий [88] для описания семантической информации одного участника может использоваться до 100 RDF-троек. Тогда, $N_{\text{RDF}}/100$ будет отражать обрабатываемое количество участников для заданного N_{RDF}.

В ресурсно-ограниченных средах Интернета вещей имеется в наличии множество разнообразных СВУ. Эксперименты показывают, что появляется техническая возможность запуска информационного хранилища как на встроенных ресурсно-ограниченных ЭВМ, так и на персональных и серверных ЭВМ. Согласно полученным данным можно сделать вывод, что время выполнения операций insert, remove, query при $10^3 \leq N_{\text{RDF}} \leq 35 \cdot 10^3$ троек не превышает 8 с., а встроенное СВУ Raspberry Pi позволяет работать с объемами до 10^4 троек за время, не превышающее 10 с. Для системы проведения мероприятий совместной деятельности такое время обработки является приемлемым при работе с числом участников до 350 человек в случае персональной и серверной ЭВМ и 100 человек в случае встроенной ЭВМ. Операция query является наиболее быстрой операцией, а операция insert наиболее долгой.

В информационном хранилище CuteSIB поддерживаются разные типы баз

данных для хранения RDF-троек. Наиболее распространенными типами баз данных является представление а) в базе Berkeley DB (в виде файла), б) в оперативной памяти. Было выполнено сравнение времени выполнения операций insert (рисунок 4.4), query (рисунок 4.5), remove (рисунок 4.6), при использовании этих двух типов базы данных для информационного хранилища CuteSIB.

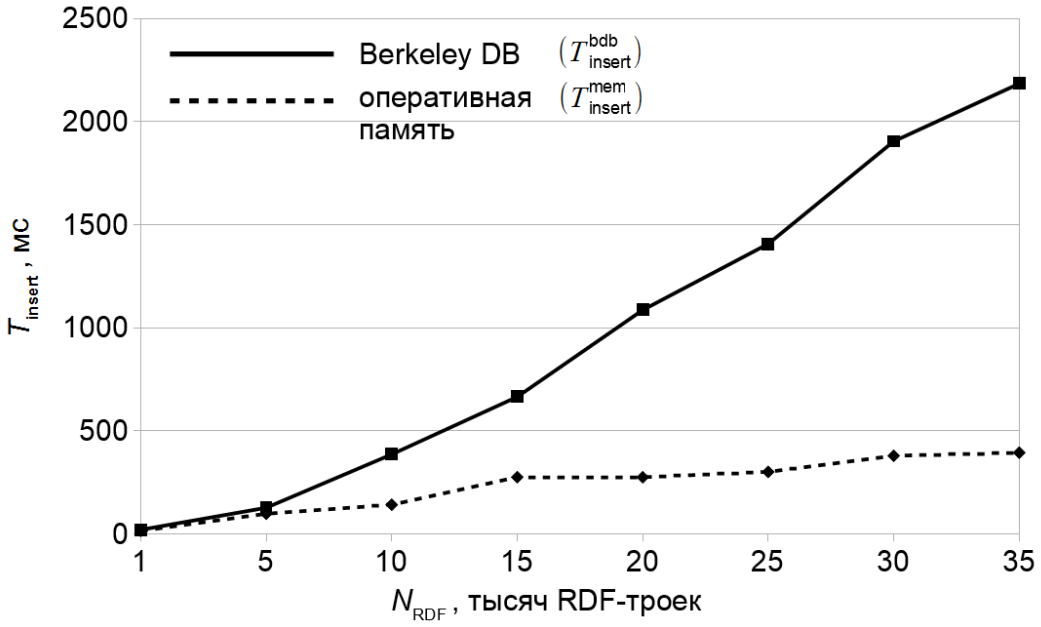


Рисунок 4.4. Соотношение времени выполнения операций insert с различными типами хранилища троек

Результаты показали, что время выполнения операций в оперативной памяти происходит быстрее, чем при использовании базы Berkeley DB, что объясняется отсутствием необходимости сохранять информацию в файл. При увеличении количества троек в операции время обработки растет значительно быстрее в случае с Berkeley DB и на больших объемах превышает 1 с., тогда как при работе с памятью время выполнения любой операции не превышает 500мс. Таким образом, для систем, в которых важна скорость обработки информации, но не критична возможная потеря информации из-за сбоев возможно использование БД-троек в оперативной памяти. Для системы проведения мероприятий совместной деятельности приемлемым является использование базы Berkeley DB, так как время обработки операций является достаточным, а возможность

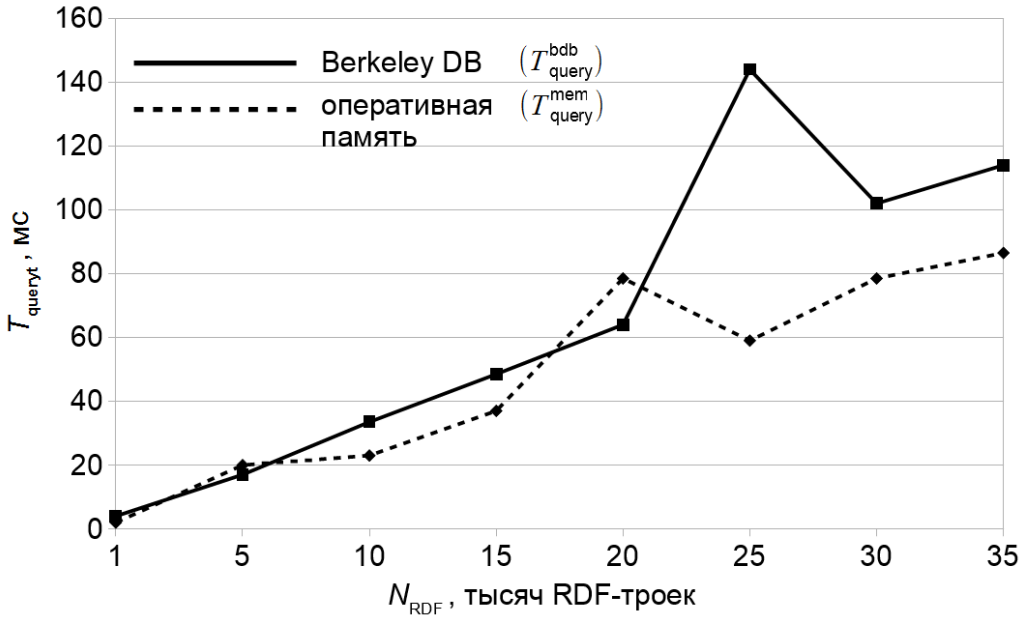


Рисунок 4.5. Соотношение времени выполнения операций query с различными типами хранилища троек

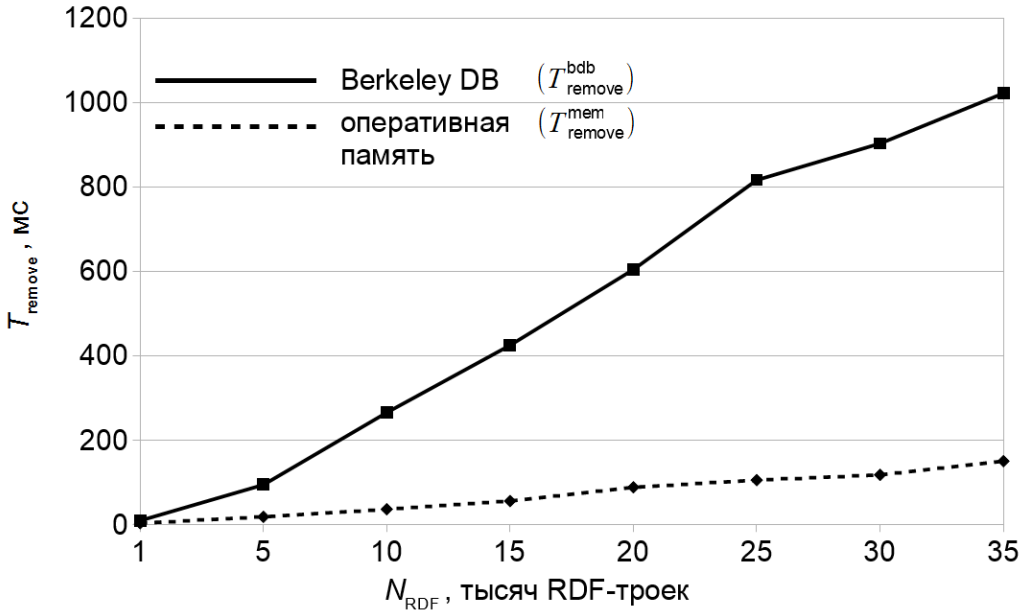


Рисунок 4.6. Соотношение времени выполнения операций remove с различными типами хранилища троек

хранения информации позволяет восстановить работоспособность сервиса после аварийного завершения информационного хранилища при сбое.

При получении запроса на выполнение операции в информационном хранилище CuteSIB сначала выполняется обработка сетевого запроса: разбор и преобразование сетевого запроса во внутреннюю структуру операции. Процесс раз-

бора запроса также требует вычислительных ресурсов ЭВМ. Измерялось время разбора запроса и время выполнения операции в хранилище RDF-троек (рисунок 4.7) для операции insert (как самой долгой операции). При $N_{\text{RDF}} \leq 5 \cdot 10^3$ разбор и обработка операции выполняются приблизительно равный промежуток времени. С увеличением количества RDF-троек при $N_{\text{RDF}} > 5 \cdot 10^3$ время обработки операции растет значительно быстрее времени разбора сетевого запроса.

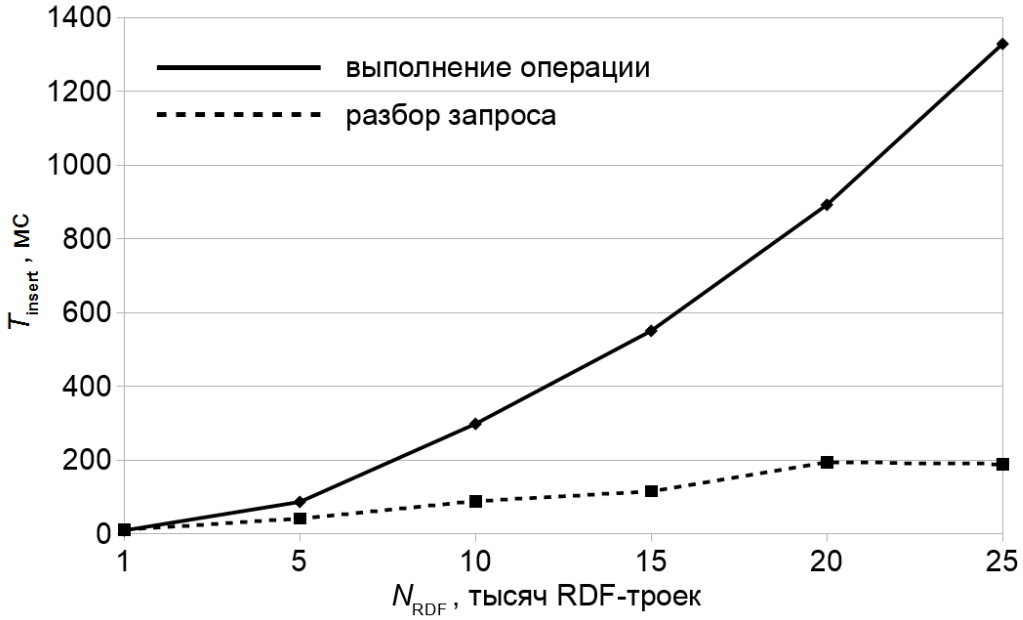


Рисунок 4.7. Соотношение времени обработки и выполнения для операции insert

Результаты эксперимента показывают, что время разбора слабо зависит от количества RDF-троек в запросе и не превышает 200 мс, следовательно, существует техническая возможность выполнять разбор больших запросов на слабопроизводительных СВУ. Время обработки операции возрастает при увеличении N_{RDF} и зависит от используемого типа базы данных для хранения RDF-троек. Для снижения времени обработки может использовать другой тип базы данных (например, в памяти), либо удаленная БД на другой более производительной ЭВМ.

Для исследования повышения производительности за счет использования модели управления сетевым доступом агентов проведено сравнение показателей

с другой реализацией информационного хранилища — RedSIB [64, 82]. Измерялось при хранении в информационном хранилище от 10^3 до $35 \cdot 10^3$ RDF-троек. Измерялось время выполнения T_o операций insert, remove, query в информационных хранилищах CuteSIB и RedSIB. Так как операция update является последовательным выполнением операций remove и insert, а операция subscribe основана на операции query, то время их выполнения соотносится с временем выполнения операций, на которых они основаны. Измерялось время выполнения операции T_o с определенным количеством RDF-троек $10^3 \leq N_{\text{RDF}} \leq 35 \cdot 10^3$ и при хранении в информационном хранилище от 10^3 до $35 \cdot 10^3$ RDF-троек. В ходе выполнения экспериментального исследования было установлено, что информационное хранилище RedSIB прекращает свое функционирование при $N_{\text{RDF}} > 15 \cdot 10^3$ RDF-троек.

В соответствии с рисунком 4.8 установлено, что время выполнения $T_{\text{query}}^{\text{RedSIB}} \gg T_{\text{query}}^{\text{CuteSIB}}$ в несколько раз. При увеличении количества RDF-троек разница во времени выполнения операции между информационными хранилищами CuteSIB и RedSIB возрастает. При $N_{\text{RDF}} > 5 \cdot 10^3$ время выполнения операции $T_{\text{query}}^{\text{RedSIB}} > 1$ минуты, что является неприемлемым для большинства приложений интеллектуальных пространств. Время выполнения операции $T_{\text{query}}^{\text{CuteSIB}}$ даже при $N_{\text{RDF}} = 35 \cdot 10^3$ не превышает 10 секунд.

В соответствии с рисунком 4.9 установлено, что время выполнения $T_{\text{remove}}^{\text{RedSIB}} \approx T_{\text{remove}}^{\text{CuteSIB}}$ при $N_{\text{RDF}} = 10^3$ RDF-троек. Однако, при $N_{\text{RDF}} > 10^3$ RDF-троек время выполнения операции $T_{\text{remove}}^{\text{RedSIB}} > T_{\text{remove}}^{\text{CuteSIB}}$ более чем в 2 раза.

По результатам измерений можно сделать вывод, что время обработки основных операций (insert, query, remove) в информационном хранилище CuteSIB меньше времени обработки этих операций в информационном хранилище RedSIB. Эффективность обработки операций по сравнению с другими реализациями информационного хранилища показана в [121].

Проведено исследование на наличие утечек памяти в реализации информационного хранилища CuteSIB с измерением объема M_{start} и M_{end} занимаемой

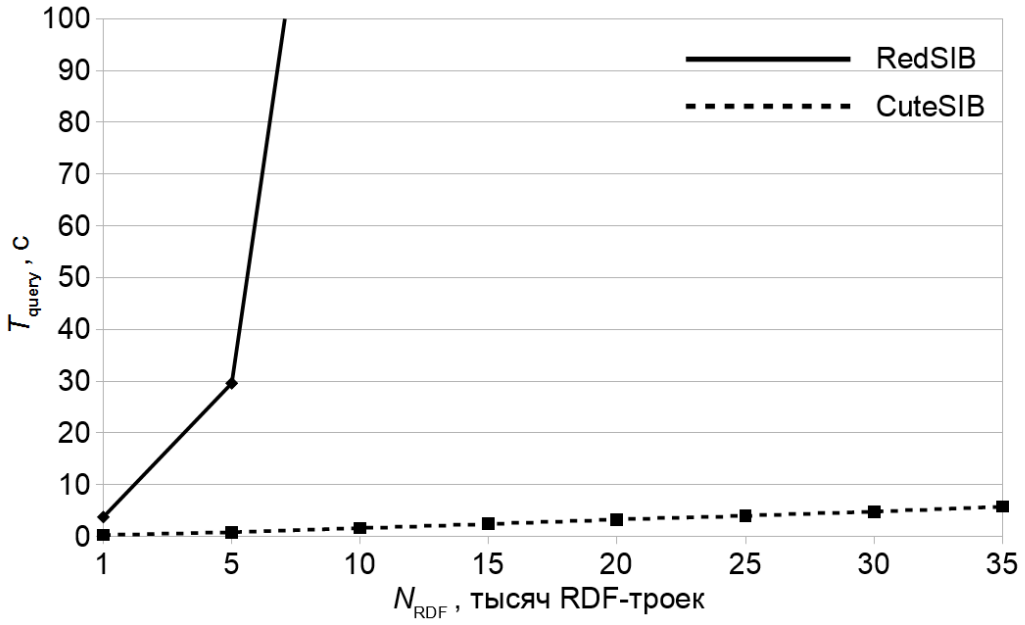


Рисунок 4.8. Время выполнения операции query в информационных хранилищах CuteSIB и RedSIB

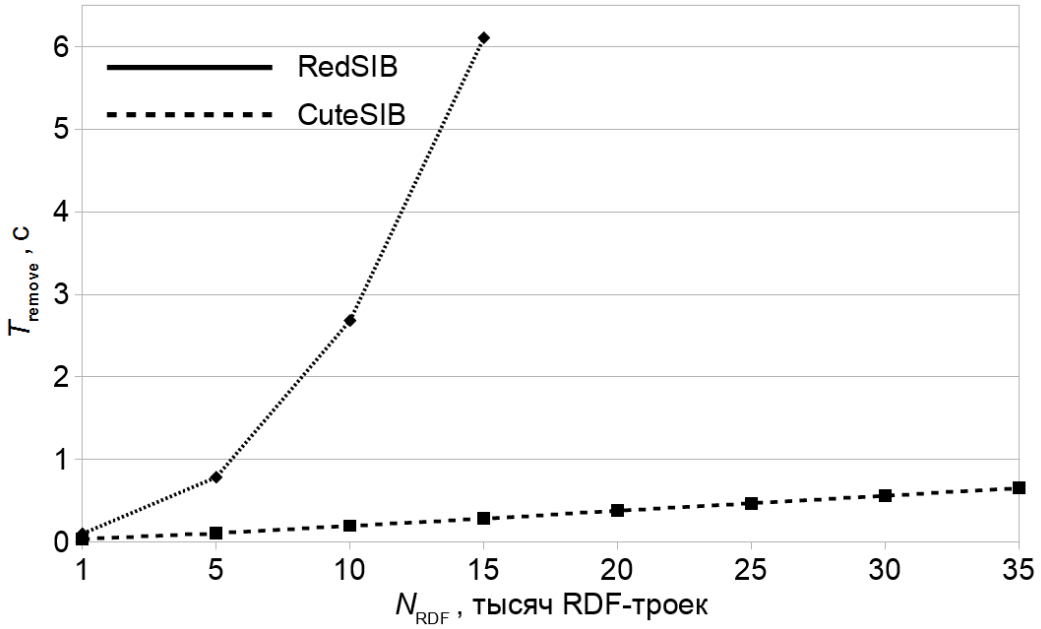


Рисунок 4.9. Время выполнения операции remove в информационных хранилищах CuteSIB и RedSIB

хранилищем оперативной памяти до и после периода активной работы с хранилищем. Экспериментальное исследование на наличие утечек памяти проводилось с использованием утилиты valgrind (с модулем-анализатором memcheck). Модуль-анализатор memcheck используется для обнаружения утечек памяти и

ошибок, связанных с неправильной работой с областями памяти. Проводимые тесты включали в себя набор циклов операций для различного количества троек. Цикл включал в себя три операции: добавить тройки, получить тройки, удалить тройки из информационного хранилища. В течение 12 часов с периодичностью 1 раз в минуту добавлялось $N = 5 \cdot 10^4$ троек, которые затем удалялись. Результаты измерений показали, что $M_{\text{start}}^{\text{CuteSIB}} \approx M_{\text{end}}^{\text{CuteSIB}}$, в отличие от реализации RedSIB.

Исследование способа параллельной обработки операций выполнялось на примере операции query. Одновременно запускалось $n = 10$ агентов, каждый из которых осуществлял запрос 5 тысяч троек. Измерялось время t получения этого запроса для каждого из агентов. На рисунке 4.10 изображены графики времени выполнения запроса при параллельной и последовательной обработке операции query. По результатам измерений видно, что при последовательной обработке для каждого следующего агента время обработки операции линейно возрастает. При параллельной обработке запросов время обслуживания первых трех агентов приблизительно одинаковое, затем наблюдается скачок и время обработки последующих агентов также является приблизительно одинаковым. Таким образом, можно сказать, что при параллельной обработке операции гарантируется одинаковое время обслуживания агентов, тогда как при последовательной обработки время обслуживания растет линейно числу параллельно работающих агентов.

В целом, предложенная модель управления сетевым доступом повышает эффективность совместного построения сервиса агентами за счет повышения производительности при выполнении операций на СВУ с ограниченными ресурсами, обеспечения долговременной работоспособности информационного хранилища и поддержки справедливости доступа агентов к информационному хранилищу.

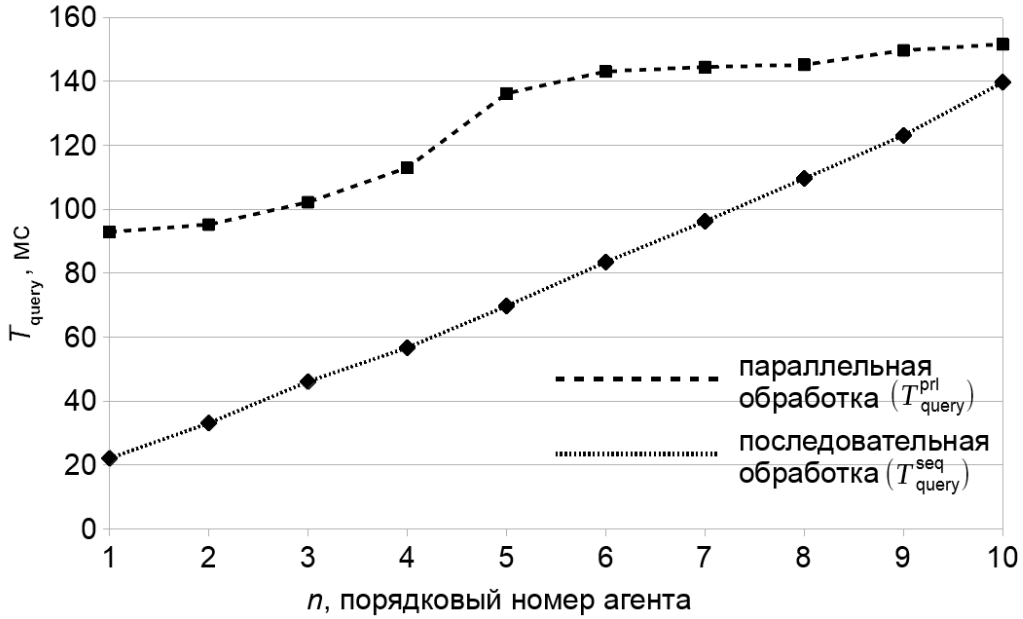


Рисунок 4.10. Время выполнения операции query при параллельной и последовательной обработке

4.2. Время восстановления после сбоев на примере системы проведения мероприятий совместной деятельности

Для исследования производительности восстановления после сбоев в модели обеспечения устойчивости программной инфраструктуры измерялось время восстановления работоспособности отдельного программного агента. Эксперименты проводились на примере системы проведения мероприятий совместной деятельности. Время восстановления работоспособности измерялось для механизмов перезапуска и переподключения.

Для запуска информационного хранилища (CuteSIB), сервиса управления конференцией (Conference-service) и агента хранения бинарных данных (Content-service) использовалась серверная ЭВМ (Ubuntu Linux, Intel Xeon 2.30GHz, 4GB RAM). Для запуска сервисов управления программой мероприятия (Agenda-service) и презентациями докладчиков (Presentation-service) использовалась персональная ЭВМ (Windows, Intel Core 2 Quad 2.40GHz, 8GB RAM).

При перезапуске измерялось время T_{rst} между возникшим сбоем и восстановлением работы агента. При переподключении измерялось время T_{rct} с

момента разрыва сетевого соединения агента с информационным хранилищем до полного восстановления соединения. В каждом эксперименте выполнено по 50 измерений для каждого агента (в системе проведения мероприятий совместной деятельности каждый сервис представлен одним отдельным агентом). Сбои имитировались искусственно за счет аварийного завершения вычислительного процесса агента и разрыва сетевого соединения.

Результаты сведены в таблицу 4.1. В результате получено, что $T_{\text{rst}} \approx 2$ с, $T_{\text{rct}} \approx 0,2$ с. Можно сделать вывод, что механизм перезапуска позволяет восстановить работоспособность сервиса за приемлемое время (в условиях системы проведения мероприятий совместной деятельности — не более 2 с). Переподключение происходит примерно в 10 раз быстрее, поскольку процесс агента продолжает работать.

Таблица 4.1. Время восстановления сервисов после сбоя

Время восстановления	Информационное хранилище	Conference-service	Agenda-service	Presentation-service	Content-service
T_{rst}, с	1,045	2,023	1,350	0,230	2,193
T_{rct}, с	–	–	0,134	0,075	–

При использовании программного механизма контроля подписки осуществляется проверка измененных данных подписки. Тем не менее, не всегда возможно корректное определение изменения данных (например, между двумя проверками данные могут быть изменены, а затем возвращены в исходное состояние). Кроме этого, не всегда возможно быстро обнаружить разрыв сетевого соединения для подписки. Указанные проблемы требуют разработки математических методов для оценки частоты таких проверок. Для платформы Smart-M3 в работе [119] предлагается модель для определения интервалов активной проверки подписки.

Показано, что такая производительность приемлема для работы мобильных клиентов в системах проведения мероприятий совместной деятельности,

где каждый клиент представляет человека-участника.

Для оценки эффективности агента хранения бинарных данных измерялось время T_{bds} загрузки и сохранения файла размером 10 Мб. Указанный объем определяет максимально разрешенный размер одной презентации в системе проведения мероприятий совместной деятельности и отражает наиболее часто загружаемый вид объемных фактических данных. Время загрузки измерялось для двух популярных веб-серверов: Apache и nginx. Они представляют требовательный и нетребовательный к ресурсам СВУ варианты веб-серверов, соответственно. Проведенное измерение позволяет оценить зависимость сервиса от используемого промежуточного ПО. Среднее время загрузки составляет $T_{\text{bds}}^{\text{Apache}} = 1,814$ с (стандартное отклонение 0,014) и $T_{\text{bds}}^{\text{nginx}} = 1,842$. Таким образом, агент Content-service имеет сходную эффективность вне зависимости от выбора веб-сервера. Загрузка презентации требует менее 2 с, что приемлемо для условий системы проведения мероприятий совместной деятельности.

Агент хранения бинарных данных предназначен для раздельного хранения бинарных данных и их семантики. Допустим, что объемные бинарные данные будут храниться в информационном хранилище. В этом случае, если произойдет сбой (с информационным хранилищем, сетью или агентом) во время сохранения или обработки такого файла, то он может быть поврежден или потерян. Содержимое I представлено набором RDF-троек. Преобразование объемных данных в тройки является сложной и затратной по времени операцией. Если сбой прервет эту операцию, то целостность данных будет нарушена. Предложенное решение использует проверенные веб-технологии для сетевого обмена файлами и файловую систему для их хранения. Сбой также может произойти, когда сервис обрабатывает файл, хранящийся на удаленном компьютере. Если возникают сбои в работе сети или на удаленном компьютере, то файл становится недоступным. Агент хранения бинарных данных заранее дублирует файл локально и, таким образом, обеспечивается доступность файла.

Агент хранения бинарных данных также в определенной мере автомати-

зирует процесс сбора файлов. В таких системах, как система проведения мероприятий совместной деятельности, участники должны предоставлять организаторам свои материалы для презентации. В случае ручного сбора файлов сбои часто возникают из-за влияния человеческого фактора. Организаторы могут ошибиться во время получения или сохранения файлов. Агент хранения бинарных данных автоматизирует и определяет общий подход к сбору файлов, тем самым снижая влияние человеческого фактора.

Предложенный агент хранения бинарных данных улучшает отдельные свойства работоспособности (безотказность, целостность), но он представляет централизованное решение. Отметим, что наличие распределенного хранилища может существенно усложнить систему и ухудшить ее работоспособность из-за увеличения числа элементов системы. Использование данного сервиса в системе проведения мероприятий совместной деятельности показывает, что централизованное решение является приемлемым для рассмотренного класса приложений.

Рассмотренные ранее решения в виде механизмов восстановления после сбоев и агента хранения бинарных данных предназначены для повышения работоспособности сервисов. Они улучшают такие характеристики, как безотказность и целостность данных. В целом, предложенная модель обеспечения устойчивости программной инфраструктуры предоставляет техническую возможность создания ИП для долговременной работы в условиях нестабильности IoT-среды.

4.3. Время обработки информационных уведомлений в зависимости от количества параметров

Применение модели информационных уведомлений повышает эффективность создания ИП за счет применения унифицированного способа программирования участия каждого агента в построении сервиса. Для исследования ухудшения производительности выполнения взаимодействия программных агентов

(из-за унификации), измерялось время $T_{\text{ntf}}(n)$, необходимое для получения агентом как самого уведомления (сигнал о необходимости участия во взаимодействии), так и всех n связанных с этим событием параметров (описание требуемого взаимодействия). Число параметров в уведомлении влияет на производительность приложения при обработке уведомления из-за особенностей операции подписки [20]. Рассмотрим экспериментальную оценку зависимости времени обработки уведомления от числа параметров.

Пусть $T_{\text{ntf}}(n)$ — это время, прошедшее с момента отправки уведомления агентом A_{snd} до получения результата агентом A_{rcv}. Было проведено два типа экспериментов: 1) все n параметров уведомления извлекаются одним запросом, 2) каждый параметр $i = 1, 2, \ldots, n$ получается отдельным запросом (запрос из n итераций) тройки-шаблона: $\langle \text{id}, p_i, * \rangle$.

С точки зрения производительности, первый тип соответствует наилучшему варианту реализации подписки агентом A_{rcv} на уведомление с n параметрами. Второй тип демонстрирует производительность наихудшего случая, когда агент A_{rcv} запускает запросы в цикле для получения всех n параметров. Пусть $T^{\text{bst}}_{\text{ntf}}(n)$ и $T^{\text{wst}}_{\text{ntf}}(n)$ — измеряемые оценки затрачиваемого времени. Таким образом, они будут показывать нижнюю и верхнюю границы производительности.

В экспериментах информационное хранилище запускалось на серверной ЭВМ. На персональном компьютере были запущены агенты A_{snd} и A_{rcv} (реализованы на языке Python). Выбранные серверная ЭВМ и персональный компьютер находятся в различных локальных сетях, в среднем RTT = 3 мс. В реальной ситуации агенты A_{snd} и A_{rcv} обычно запущены на разных компьютерах. Использование общего компьютера для агентов упростило измерение времени, исключив необходимость синхронизации времени. В то же время, каждый из этих агентов взаимодействует по сети непосредственно лишь с информационным хранилищем, а следовательно запуск агентов на одном компьютере достаточно близок к ситуации, когда эти агенты работают на двух одинаковых равноудаленных от серверной ЭВМ компьютерах.

Каждая итерация измерений состоит из шагов: 1) A_{snd} отправляет уведомление с n-параметрами в I, 2) A_{rcv} получает уведомление от информационного хранилища, получает значения всех n параметров и удаляет уведомление. Выполнено по 100 измерений для каждого значения n, варьируемого от 1 до 100, с вычислением средних значений для $T_{\text{ntf}}^{\text{bst}}(n)$ и $T_{\text{ntf}}^{\text{wst}}(n)$. Результат представлен на рисунке 4.11 для n усеченного до 50, чтобы растущая разность между $T_{\text{ntf}}^{\text{bst}}(n)$ и $T_{\text{ntf}}^{\text{wst}}(n)$ не скрыла детали.

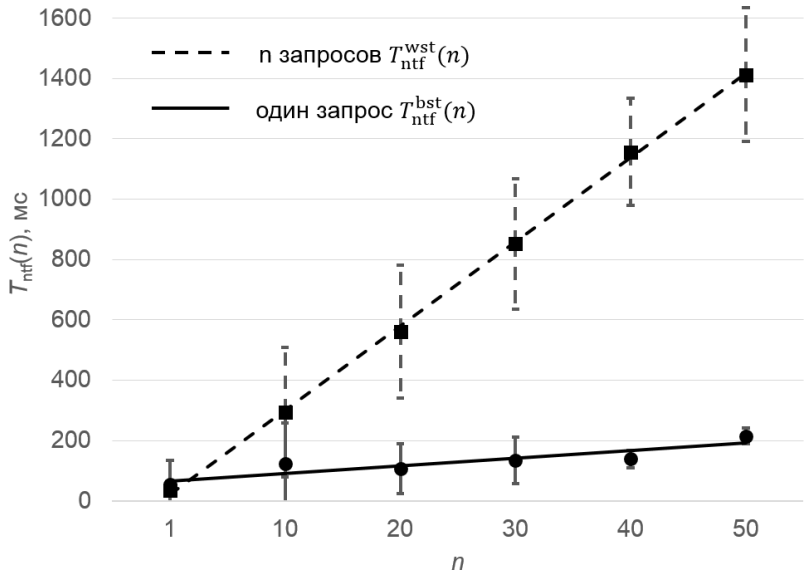


Рисунок 4.11. Экспериментальное поведение $T_{\text{ntf}}^{\text{bst}}(n)$ и $T_{\text{ntf}}^{\text{wst}}(n)$ при $n \leq 100$

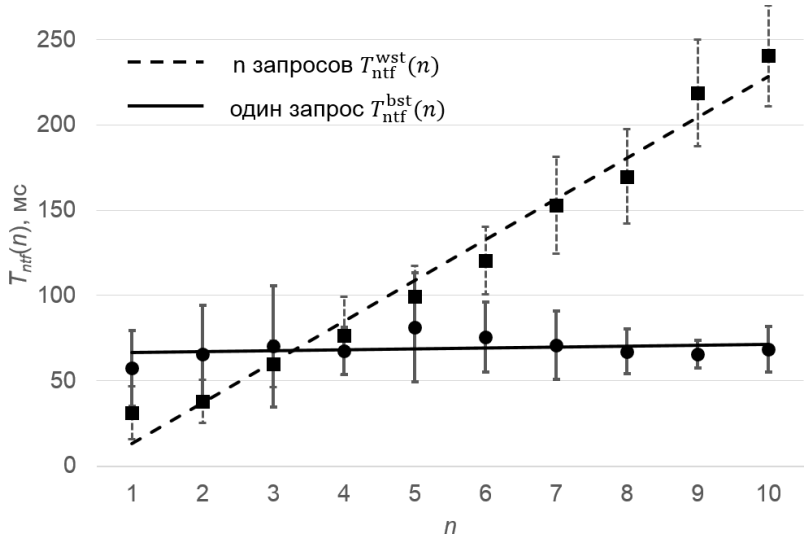


Рисунок 4.12. Экспериментальное поведение $T_{\text{ntf}}^{\text{bst}}(n)$ и $T_{\text{ntf}}^{\text{wst}}(n)$ при $n \leq 10$

Поведение $T_{\text{ntf}}^{\text{bst}}(n)$ и $T_{\text{ntf}}^{\text{wst}}(n)$ хорошо описывается линейной регрессией (построена для n от 1 до 100 с вычислением угловых коэффициентов k). Разница

угловых коэффициентов показывает более быстрый рост $T_{\text{ntf}}^{\text{wst}}(n)$ по сравнению с $T_{\text{ntf}}^{\text{bst}}(n)$. Этот факт является следствием организации запроса в виде цикла. Такой циклический вариант также приводит к большей вариабельности для $T_{\text{ntf}}^{\text{wst}}(n)$ (для каждого n на графике показано среднеквадратическое отклонение). Наблюдаемый линейный рост $T_{\text{ntf}}^{\text{bst}}(n)$ имеет малый угловой коэффициент k. Наличие такого роста вызвано увеличением времени обработки данных на стороне информационного хранилища и необходимостью передачи большего объема данных по сети при увеличении n. В целом, эксперименты позволяют сделать вывод, что предложенная модель уведомлений сохраняет приемлемую производительность даже для больших значений n.

Разумно предполагать, что для большинства информационных сервисов значение n ограничено, т.е. события, инициирующие взаимодействие, имеют компактное описание в силу особенностей предметной области. График на рисунке 4.12 показывает измерения для всех $n = 1, 2, \ldots, 10$ с построением линейной регрессии только для данного отрезка. Как и ранее, наблюдается линейное поведение, но угловые коэффициенты k меньше по сравнению со случаем для $n \leq 100$. При небольших n угловой коэффициент k линейной регрессии для $T_{\text{ntf}}^{\text{bst}}(n)$ практически равен нулю, т.е. временные затраты близки к константе.

Отметим, что изменение угловых коэффициентов линейной регрессии при смене интервалов для n свидетельствует о наличии нелинейного эффекта в производительности. Так, для $T_{\text{ntf}}^{\text{wst}}(n)$ коэффициент k равен 27.9 и 23.9 при $n \leq 100$ и $n \leq 10$, соответственно. Причиной такого эффекта является нелинейность по n затрат времени на выполнение а) поисковых запросов на стороне информационного хранилища и б) сетевой передачи данных между информационным хранилищем и агентом. В целом, предложенная модель информационных уведомлений сохраняет приемлемую для референтного ИП производительность. Даже для значений n порядка нескольких десятков (рост сложности варианта взаимодействия) время $T_{\text{ntf}}(n)$ не превышает нескольких секунд. Для референтного ИП выполняется ограничение $n < 10$, что приводит к $T_{\text{ntf}}^{\text{bst}}(n) < 1$ с.

4.4. Время обработки операций программной инфраструктурой в зависимости от числа агентов

Для исследования производительности в целом программной инфраструктуры, разрабатываемой по предложенному методу, проводились имитационные эксперименты на основе разработанного комплекса программных средств. Исследовалась зависимость от числа взаимодействующих с информационным хранилищем агентов трех типов (характерны для многих ИП): 1) сенсоры — обновляют конкретное значение в хранилище с частотой λ_{sns} с^{-1} (основная операция сетевого доступа— update); 2) обработчики данных — каждый подписывается на часть информации, регулярно поступающей от сенсоров, и выполняет их обработку с публикацией агрегированного значения (основная операция сетевого доступа— subscribe); 3) клиенты — ожидают агрегированных значений для доставки сервиса пользователю (основная операция сетевого доступа — query). Запускалось n_{sns} агентов-сенсоров, n_{rsn} агентов-обработчиков и n_{cln} агентов-клиентов в пропорции $10^3 : 1 : 10^2$, т.е. 1 обработчик на 10^3 сенсоров и 10^2 клиентов. При этом число обработчиков доходило до нескольких десятков. Такая пропорция отражает предположение: а) обработчик соответствует агенту, ответственному за построение основной части сервиса, б) для построения сервиса нужны данные от множества сенсоров, в) сервис необходим для нескольких клиентов. Измерялось время выполнения соответствующей операции для каждого типа агентов, варьируя число агентов в рамках выбранной пропорции и для различных значений λ_{sns}.

Каждому агенту-сенсору $1, \ldots, n_{\text{sns}}$ соответствует отдельная RDF-тройка, в которой он публикует число (начальное значение равно 1), а затем инкрементирует число с помощью операции обновления (update). Все множество агентов-сенсоров осуществляет публикацию с частотой λ_{sns}. Для того, чтобы выполнялась соответствующая интенсивность публикации каждый агент-сенсор осуществлял публикацию с задержкой времени, выбираемой случайным образом в

промежутке $[0, \frac{2n_{\text{sns}}}{\lambda_{\text{sns}}}]$ в соответствии с равномерным распределением. Каждый агент-обработчик $1, \ldots, n_{\text{rsn}}$ подписывается на тройки сенсоров в определенном промежутке равном $\frac{n_{\text{sns}}}{n_{\text{rsn}}}$, $n_{\text{sns}} > n_{\text{rsn}}$. Т.е. каждый обработчик отслеживает изменения в определенном подмножестве значений сенсоров, подмножества не пересекаются. После получения изменений по подписке обработчик публикует (операция update) число равное общему количеству полученных уведомлений подписки от сенсоров в виде RDF-тройки соответствующей данному обработчику. Агенты-клиенты запрашивают (операция query) все значения, опубликованные обработчиками, с частотой $\lambda_{\text{cln}} = \lambda_{\text{sns}}$.

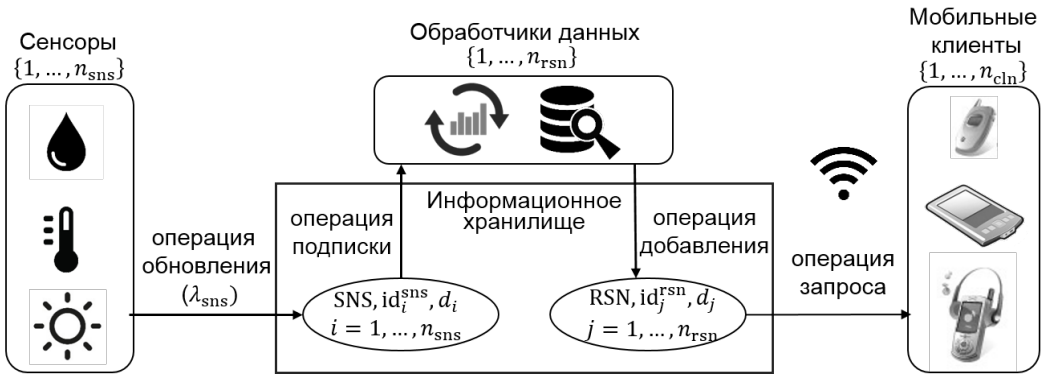


Рисунок 4.13. Схема эксперимента

Каждый тип агентов запускался на отдельной персональной ЭВМ со следующими характеристиками: CPU 2.3 MHz (2 ядра), RAM 4Gb. Для запуска агентов-сенсоров при $n_{\text{sns}} > 50000$ использовалось две ЭВМ вследствие ограничений ОС на количество сетевых соединений. Агенты обработчики и клиенты запускались каждый в отдельном потоке экспериментального приложения.

Измерялось время выполнения соответствующей операции для каждого вида агентов при различных пропорциях агентов и различной частотой публикации сенсоров. В таблице 4.2 приведены полученные значения. При увеличении количества взаимодействующих с информационным хранилищем агентов и при увеличении интенсивности λ_{sns} увеличивается время выполнения операций. На время выполнения операции существенно влияет количество подписок.

В целом, возможности программной инфраструктуры достаточны для ор-

Таблица 4.2. Среднее время выполнения операции при различной интенсивности λ_{sns}

Число агентов n_{sns}, n_{rsn}, n_{cln} (λ_{sns})	Сенсоры (update)	Обработчики (subscription + update)	Клиенты (query)
10^4, 10, 10^3 (2 с^{-1})	0,017	0,059	0,078
10^4, 10, 10^3 (5 с^{-1})	0,061	0,052	0,087
$25 \cdot 10^3$, 25, $25 \cdot 10^2$ (5 с^{-1})	0,043	0,037	0,101
$50 \cdot 10^3$, 50, $50 \cdot 10^2$ (2 с^{-1})	0,034	0,072	0,158
$50 \cdot 10^3$, 50, $50 \cdot 10^2$ (5 с^{-1})	0,129	0,101	0,187
$75 \cdot 10^3$, 75, $75 \cdot 10^2$ (5 с^{-1})	0,226	0,175	0,261
10^5, 10^2, 10^4 (2 с^{-1})	0,070	0,129	0,265
10^5, 10^2, 10^4 (5 с^{-1})	0,317	0,280	0,353

ганизации взаимодействия агентов в пределах до $10^4 \ldots 10^5$ одновременно участвующих СВУ с ограниченными ресурсами.

4.5. Выводы

Применение предложенного метода разработки позволяет обеспечить построение информационных сервисов для рассматриваемых условий ресурсно-ограниченных IoT-сред и с выполнением вычислений на имеющихся СВУ. Обеспечивается возможность запуска информационного хранилища на таких устройствах как одноплатные компьютеры, сетевые маршрутизаторы, мобильные телефоны. Обеспечивается возможность восстановления компонентов программной инфраструктуры после разнообразных сбоев. В ряде случаев время восстановления может занимать до нескольких секунд, что достаточно для ИП с участием мобильных конечных пользователей. Повышается эффективность программной разработки сервисов за счет применения унифицированного способа программирования участия агента во взаимодействии, а производительность построения сервисов при таком взаимодействии агентов повышается за счет снижения числа подписок до одной для отдельного агента.

Заключение

В диссертации предложено решение актуальной задачи по организации косвенного взаимодействия программных агентов интеллектуального пространства, позволяющей обеспечить техническую возможность и повысить эффективность построения информационных сервисов в условиях ресурсно-ограниченных IoT-сред. В процессе решения данной задачи были получены следующие результаты:

1. Предложен метод разработки программной инфраструктуры интеллектуального пространства, позволяющий настроить информационное хранилище в соответствии с аппаратно-сетевыми ограничениями IoT-среды и требованиями предметной области, автоматизировать восстановление компонентов программной инфраструктуры после возникающих сбоев, организовать взаимодействие программных агентов при построении сервисов унифицированным способом.
2. Предложена концептуальная модель управления сетевым доступом программных агентов к информационному хранилищу, позволяющая настраивать набор поддерживаемых хранилищем операций сетевого доступа и способов их параллельной реализации для эффективного использования имеющихся вычислительных ресурсов.
3. Предложена структурная модель обеспечения устойчивости компонентов программной инфраструктуры к сбоям IoT-среды, позволяющая восстанавливать работоспособность компонентов инфраструктуры после сбоев на уровнях: а) операции подписки, б) сетевых соединений между агентами и информационным хранилищем, в) вычислительных процессов агентов на СВУ, г) хранения данных.
4. Предложена онтологическая модель информационных уведомлений, позволяющая повысить эффективность процесса программной разработки приложения и упрощающая интеграцию ранее разработанных сервисов.

5. Создан комплекс программных средств в составе: а) реализация информационного хранилища CuteSIB для создания и настройки интеллектуальных пространств в ресурсно-ограниченных IoT-средах; б) реализация компонентов программной инфраструктуры на примере системы проведения мероприятий совместной деятельности, обеспечивающих построение опорных сервисов с использованием устройств с ограниченными вычислительными ресурсами, устойчивость к сбоям и возможность интеграции сервисов из других ИП; в) реализация ПО для проведения экспериментального исследования по оценке возможностей применения предложенных моделей.

Перспективными направлениями исследования являются внедрение в программную инфраструктуру компонентов для обеспечения безопасности, а также автоматизация процесса разработки, развертывания и настройки программной инфраструктуры и информационных сервисов.

Полученные результаты соответствуют п. 3 «Модели, методы, алгоритмы, языки и программные инструменты для организации взаимодействия программ и программных систем» и п. 8 «Модели и методы создания программ и программных систем для параллельной и распределенной обработки данных, языки и инструментальные средства параллельного программирования» паспорта специальности 05.13.11 — «Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей».

Список сокращений и условных обозначений

БД — база данных.

ИП — интеллектуальное пространство.

ОС — операционная система.

ПО — программное обеспечение.

Процессор знаний (KP от англ. Knowledge Processor) — программный агент для интеллектуального пространства на платформе Smart-M3.

Свойство-данные — отношения между экземплярами классов и RDF-литералами или типами данных, определяемых XML Schema.

Свойство-объект — отношения между экземплярами двух классов.

СВУ — сетевое вычислительное устройство.

Семантический веб — направление развития Всемирной паутины, целью которого является представление информации в виде, пригодном для эффективной машинной обработки.

СПО — системное программное обеспечение.

Т-шаблон — RDF-тройка, где компоненты заменены маской.

CPU (Central Processing Unit) — центральное процессорное устройство.

IoT (Internet of Things) — концепция “Интернет вещей”.

OSGi (Open Services Gateway Initiative) — спецификация динамической модульной архитектуры для Java-приложений.

OWL (Web Ontology Language) — язык описания онтологий для Семантического веб.

RAM (Random Access Memory) — запоминающее устройство с произвольным доступом.

RDF (Resource Description Framework) — модель Семантического веб для представления данных и метаданных.

RDF-тройка — утверждение о ресурсе, состоящее из трёх компонентов, имеет вид: “субъект - предикат - объект”.

SIB (Semantic Information Broker) — брокер семантической информации.

Smart-M3 — платформа с открытым исходным кодом, реализующая концепцию интеллектуальных пространств с косвенным взаимодействием агентов.

SPARQL — язык запросов к данным, представленным на основе модели RDF.

SSAP (Smart Space Access Protocol) — протокол доступа к интеллектуальному пространству на платформе Smart-M3.

URI (Uniform Resource Identifier) — унифицированный идентификатор ресурсов.

XML (eXtensible Markup Language) — расширяемый язык разметки.

Список литературы

1. Аксенова, М. А. От интернета людей — к интернету вещей: концепция XXI века [Текст] / М. А. Аксенова, Р. Я. Рахматулин // Информационные ресурсы России. — 2016. — № 5(153). — С. 37-39.
2. Александров, В. А. Разработка и анализ защищенности фрагмента информационно-телекоммуникационной системы, реализующей концепцию Интернета вещей [Текст] / В. А. Александров, В. А. Десницкий, Д. Ю. Чалый // Моделирование и анализ информационных систем. — 2016. — Т. 23. — № 6. — С. 767–776.
3. Александров, В. В. Технология программно-определяемых сред и импортозамещение [Текст] / В. В. Александров, С. В. Кулешов, Р. М. Юсупов // Информатизация и связь. — 2016. — № 3. — С. 82–85.
4. Гаврилов, А. В. Искусственный домовой [Текст] / А. В. Гаврилов // Искусственный интеллект и принятие решений. — 2012. — № 2. — С. 77–89.
5. Галов, И. В. Применение шаблонов проектирования программных приложений для реализации косвенного взаимодействия агентов в интеллектуальном пространстве [Текст] / И. В. Галов // Программная инженерия. — 2016. — № 8. — С. 351–359.
6. Галов, И. В. Модель уведомлений для разработки программных приложений интеллектуальных пространств [Текст] / И. В. Галов, Д. Ж. Корзун // Труды СПИИРАН. — 2014. — Т. 35. — С. 189—211.
7. Галов, И. В. Обеспечение устойчивости к сбоям Smart-M3 приложения на уровне программной инфраструктуры [Текст] / И. В. Галов, Д. Ж. Корзун // Труды СПИИРАН. — 2014. — Т. 37. — С. 188—207.
8. Гамма Э. Приемы объектно-ориентированного проектирования. Паттерны проектирования [Текст] / Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес. — [Б. м.] : СПб: «Питер», 2007. — 366 с.
9. Городецкий, В. И. Агенты и извлечение знаний из данных в интеллекту-

альном пространстве [Текст] / В. И. Городецкий // Сборник трудов первой международной конференции «Автоматизация управления и интеллектуальные системы и среды». — [Б. м. : б. и.], 2010. — С. 24–36.

10. Городецкий, В. И. Самоорганизация и многоагентные системы. I. Модели многоагентной самоорганизации [Текст] / В. И. Городецкий // Известия РАН. Теория и системы управления. — 2012. — № 2. — С. 92–120.
11. Грибова, В. В. Базовая технология разработки интеллектуальных сервисов на облачной платформе IACPaaS. Часть 2. Разработка агентов и шаблонов сообщений [Текст] / В. В. Грибова, А. С. Клещев, Д. А. Крылов, Ф. М. Москаленко, В. А. Тимченко, Е. А. Шалфеева // Программная инженерия. — 2016. – Т. 7. — № 1. — С. 14–20.
12. Давыдов Д. С. Разработка платформы планирования производства с использованием технологий «облачных вычислений» [Текст] / Д. С. Давыдов, А. М. Кашевник, Д. П. Косицын, А. И. Шабаев, И. М. Шабалина // Труды СПИИРАН. — 2012. — № 4 (23). — С. 416–430.
13. Ермолаев, В. И. Семантическая интеграция сервисов для интеллектуальной поддержки принятия решений в гибких сетях поставок [Текст] / В. И. Ермолаев, Т. В. Левашова, Н. Г. Шилов // Труды СПИИРАН. — 2014. — № Вып. 35. — С. 177–188.
14. Загорулько, Ю. А. Технология создания тематических интеллектуальных научных интернет-ресурсов, базирующаяся на онтологии [Текст] / Ю. А. Загорулько, Г. Б. Загорулько, О. И. Боровикова // Программная инженерия. — 2016. — Т. 7. — № 2. — С. 51–60.
15. Замула, Д. А. Автоматизированное проектирование мобильных приложений на основе онтологий и семантических данных [Текст] / Д. А. Замула, Д. И. Муромцев // Известия высших учебных заведений. Приборостроение. — 2015. — Т. 58. — № 11. — С. 939–944.
16. Корзун, Д. Ж. Формализм сервисов и архитектурные абстракции для программных приложений интеллектуальных пространств [Текст] / Д. Ж. Кор-

зун // Программная инженерия. — 2015. — № 2. — С. 3–12.

17. Корзун, Д. Ж. Автоматизированная модельно-ориентированная разработка программных агентов для интеллектуальных пространств на платформе smart-m3 [Текст] / Д. Ж. Корзун, А. А. Ломов, П. И. Ванаг // Программная инженерия. — 2012. — № 5. — С. 6–14.
18. Котенко, И. В. Архитектура системы параллельной обработки больших данных для мониторинга безопасности сетей Интернета вещей [Текст] / И. В. Котенко, И. Б Саенко, А. Г. Кушнеревич // Программная инженерия. — 2017 (принята к печати).
19. Левоневский, Д. К. Многомодальная информационно-навигационная облачная система МИНОС для корпоративного киберфизического интеллектуального пространства [Текст] / Д. К. Левоневский, И. В. Ватаманюк, А. И. Савельев // Программная инженерия. — 2017. — Т. 8. — № 3. — С. 120–128.
20. Ломов, А. А. Операция подписки для приложений в интеллектуальных пространствах платформы smart-m3 [Текст] / А. А. Ломов, Д. Ж. Корзун // Труды СПИИРАН. — 2012. — Т. 4, № 23. — С. 439–458.
21. Лушпа, И. В. Оценка надежности в концепции Интернета вещей [Текст] / И. В. Лушпа // Новые информационные технологии в автоматизированных системах. — 2016. — № 19. — С. 216–218.
22. Марченков, С. А. Расширение возможностей совместной деятельности в интеллектуальном зале на основе сервисов электронного туризма [Текст] / С. А. Марченков, А. С. Вдовенко, Д. Ж. Корзун // Труды СПИИРАН. — 2017. — Вып. 50. — С. 165–189.
23. Медведева, Л. В. Система оповещения, основанная на концепции интернета вещей и data mining [Текст] / Л. В. Медведева // VII международный молодежный форум «Образование, наука, производство». — 2015. — С. 385–388.
24. Михайлов, С. А. Организация интеллектуальных пространств на основе платформы Smart-M3 с использованием устройств на базе операционной системы DD-WRT [Текст] / С. А. Михайлов, А. М. Кашевник // Труды

СПИИРАН. — 2017. — Вып. 52. — С. 180–203.

25. Немнюгин, С. А. Параллельное программирование для многопроцессорных вычислительных систем [Текст] / С. А. Немнюгин, О. Л. Стесик — СПб: БХВ-Петербург, 2002. — 400 с.
26. Никифоров, О. Ю. Концепция и технологии «интернета вещей» [Текст] / О. Ю. Никифоров // Современные научные исследования и инновации. — 2014. — № 11-1 (43). — С. 151–153.
27. Орлов, С. А. Технологии разработки программного обеспечения [Текст] / С. А. Орлов, Б. Я. Цилькер. — СПб: Издательский дом «Питер», 2012. — 608 с.
28. Росляков А. В. Интернет вещей [Текст] / А. В. Росляков, С. В. Ваняшин, А. Ю. Гребешков, М. Ю. Самсонов. — Самара: ПГУТИ, 2015. — 200 с.
29. Смирнов А. В. Онтолого–ориентированный многоагентный подход к построению систем интеграции знаний из распределённых источников [Текст] / А. В. Смирнов, Т. В. Левашова, М. П. Пашкин, Н. Г. Шилов // Информационные технологии и вычислительные системы. — 2002. — № 1. — С. 62-82.
30. Соловьев В. Д. Онтологии и тезаурусы [Текст] / В. Д. Соловьев, Б. В. Добров, В. В. Иванов, Н. В. Лукашевич. — Казань, Москва, 2006.
31. Разумовский, А. Г. Использование онтологий в разработке программного обеспечения [Текст] / А. Г. Разумовский, М. Г. Пантелеев // Известия Санкт-Петербургского государственного электротехнического университета ЛЭТИ. — 2011. — № 8. — С. 46–51.
32. Ронжин, А. Л. Сравнительный анализ функциональности прототипов интеллектуальных пространств [Текст] / А. Л. Ронжин, А. А. Карпов // Труды СПИИРАН. — 2013. — Вып. 24. — С. 277–290.
33. Сталлингс, У. Интернет вещей: сетевая архитектура и архитектура безопасности [Электронный ресурс] / У. Сталлингс. — Режим доступа: http://internetinside.ru/internet-veshhey-setevaya-arkhitektura-i/ (дата обращения: 16.08.2017).

34. Терехов, А. Н. Конфигурируемая ОС РВ Embox для встроенных систем [Текст] / А. Н. Терехов // Системный администратор. — 2015. — № 7–8. — С. 104–105.

35. Толкачев, С. А. Интеллектуальное производство сквозь призму третьей промышленной революции [Текст] / С. А. Толкачев, К. Н. Андрианов, Н. В. Лапенкова // Мир новой экономики. — 2014. — № 4. — С. 48–54.

36. Трибушков, Д. Э. Система для реализации сценариев интернета вещей на основе облачных технологий [Текст] / Д. Э. Трибушков, А. А. Кочетков, А. Ю. Попов // Инженерный вестник. — 2015. — № 8. — С. 11.

37. Хорошевский, В. Ф. Пространства знаний в сети интернет и semantic web (часть 1) [Текст] / В. Ф. Хорошевский // Искусственный интеллект и принятие решений. — 2008. — № 1. — С. 80–97.

38. Шабаев, А.И. Использование технологии промышленного интернета в задачах планирования и управления производством в лесной промышленности [Текст] / А. И. Шабаев , Д. П. Косицын , И. М. Шабалина , Д. Ж. Корзун // Сб. трудов XI Международной научно-технической конференции «Автоматизация и управление в ЦБП, ЛПК и энергетике». — 2016. — С. 42-47.

39. Юсупов, Р. М. От умных приборов к интеллектуальному пространству [Текст] / Р. М. Юсупов, А. Л. Ронжин // Вестник РАН: научный и общественно-политический журнал. — 2010. — Т. 80, № 1. — С. 45–51.

40. Aggarwal, C. C. The internet of things: A survey from the data-centric perspective [Текст] / C. C Aggarwal, N. Ashish, A. Sheth // Managing and mining sensor data. — Springer, 2013. — P. 383–428.

41. Anantharam, P. Data processing and semantics for advanced internet of things (iot) applications: Modeling, annotation, integration, and perception [Текст] / P. Anantharam, P. Barnaghi, A. Sheth // Proceedings of the 3rd International Conference on Web Intelligence, Mining and Semantics. — ACM, 2013. — P. 1–5.

42. Andreou, P. Towards a network-aware middleware for wireless sensor networks [Текст] / P. Andreou, D. Zeinalipour-Yiazti, P.K. Chrysanthis, G. Samaras //

8th International Workshop on Data Management for Sensor Networks (DMSN 2011), The Westin Hotel, Seattle, WA, USA. — 2011.

43. Baader, F. The description logic handbook: theory, implementation, and applications [Текст] / Ed. by F. Baader, D. Calvanese, D. L. McGuinness [et al.]. — New York, NY, USA : Cambridge University Press, 2003.
44. Balandin, S. Key properties in the development of smart spaces [Текст] / S. Balandin, H. Waris // Proc. 5th Int'l Conf. Universal Access in Human-Computer Interaction (UAHCI '09). Part II: Intelligent and Ubiquitous Interaction Environments, LNCS 5615 — Springer-Verlag, 2009. — P. 3–12.
45. Baldauf, M. A survey on context-aware systems [Текст] / M. Baldauf, S. Dustdar, F. Rosenberg // Int. J. Ad Hoc Ubiquitous Comput. — 2007. — Vol. 2, no. 4. — P. 263–277.
46. Ball, M. Managing control, convenience and autonomy [Текст] / M. Ball, V. Callaghan // Ambient Intelligence and Smart Environments. — 2012. — P. 159–196.
47. Baresi, L. Achieving self-adaptation through dynamic group management [Текст] / L. Baresi, S. Guinea, P. Saeedi // Assurances for Self-Adaptive Systems. — Springer, 2013. — P. 214–239.
48. Bartolini, S. Reconfigurable natural interaction in smart environments: approach and prototype implementation [Текст] / S. Bartolini, B. Milosevic, A. D'Elia [et al.] // Personal and Ubiquitous Computing. — 2012. — Vol. 16, no. 7. — P. 943–956.
49. Bechhofer, S. Owl: Web ontology language [Текст] / Sean Bechhofer // Encyclopedia of Database Systems. — Springer, 2009. — P. 2008–2009.
50. Billard, E. A. Pattern of agent interaction scenarios as use case maps [Текст] / E. A. Billard // IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics. — 2004. — Vol. 34, no. 4. — P. 1933—-1939.
51. Bonomi, F. Fog computing and its role in the internet of things [Текст] / F. Bonomi, R. Milito, J. Zhu, S. Addepalli // Proceedings of the first edition of

the MCC workshop on Mobile cloud computing. — ACM, 2012. — P. 13–16.

52. Chen, H. Intelligent agents meet the semantic web in smart spaces [Текст] / H. Chen, T. Finin, A. Joshi [et al.] // IEEE Internet Computing. — 2004. — Vol. 8, no. 6. — P. 69–79.
53. Cook, D. J. Pervasive computing at scale: Transforming the state of the art [Текст] / D. J. Cook, S. K. Das // Pervasive Mob. Comput. — 2012. — Vol. 8, no. 1. — P. 22–35.
54. Corkill, D. D. Collaborating Software: Blackboard and Multi-Agent Systems & the Future [Текст] / D. D Corkill // Proc. the Int'l Lisp Conference. — 2003. — Invited paper.
55. Cyganiak, R. Semantic sitemaps: Efficient and flexible access to datasets on the semantic web [Текст] / R. Cyganiak, H. Stenzhorn, R. Delbru [et al.]. — Springer, 2008.
56. da Costa, C. A. Toward a general software infrastructure for ubiquitous computing [Текст] / C. A. da Costa, A. C. Yamin, Claudio Fernando Resin Geyer // IEEE Pervasive Computing. — 2008. — Vol. 7, no. 1. — P. 64–73.
57. Dastjerdi, A. V., Buyya, R. Fog computing: Helping the Internet of Things realize its potential [Текст] / A. V. Dastjerdi, R. Buyya // Computer. — 2016. — Vol. 49. — No. 8. — P. 112–116.
58. De, S. An internet of things platform for real-world and digital objects [Текст] / S. De, T. Elsaleh, P. Barnaghi, S. Meissner // Scalable Computing: Practice and Experience. — 2012. — Vol. 13, no. 1. — P. 45–57.
59. D'Elia A. Enabling interoperability in the internet of things: A OSGi semantic information broker implementation [Текст] / A. D'Elia, F. Viola, L. Roffia [et al.] // International Journal on Semantic Web and Information Systems (IJSWIS). — 2017. — Vol. 13, no. 1. — P. 147–167.
60. Domingue, J. Toward a service web: integrating the semantic web and service orientation [Текст] / J. Domingue, D. Fensel // IEEE Intelligent Systems. —

2009. — Vol. 23, no. 1. — P. 86–88.

61. Eisenhauer, M. Hydra: A development platform for integrating wireless devices and sensors into ambient intelligence systems [Текст] / M. Eisenhauer, P. Rosengren, P. Antolin // The Internet of Things. — Springer, 2010. — P. 367–373.
62. Eugster P. Th. The many faces of publish/subscribe [Текст] / P. Th. Eugster, P. A. Felber, R. Guerraoui, A. Kermarrec // ACM Comput. Surv. — 2003. — Vol. 35, no. 2. — P. 114–131.
63. Esposito, A. Coupling semantics with things in the internet of the future: a survey [Текст] / A. Esposito // International Journal of Metadata, Semantics and Ontologies. — 2012. — Vol. 7, no. 4. — P. 254–259.
64. Etelapera, M. Feasibility evaluation of m3 smart space broker implementations [Текст] / M. Etelapera, J. Kiljander, K. Keinanen // Applications and the Internet (SAINT), 2011 IEEE/IPSJ 11th International Symposium on. — 2011. — P. 292–296.
65. Forkan, A. Cocamaal: A cloud-oriented context-aware middleware in ambient assisted living [Текст] / A. Forkan, I. Khalil, Z. Tari // Future Generation Computer Systems. — 2014. — Vol. 35. — P. 114–127.
66. Galov, I. A notification model for Smart-M3 applications [Текст] / I. Galov, D. Korzun // Proc. 14th Int'l Conf. Next Generation Wired/Wireless Networking and 7th Conf. on Internet of Things and Smart Spaces (NEW2AN/ruSMART 2014), LNCS 8638 / Ed. by S. Balandin, S. Andreev, Yevgeni Koucheryavy. — Springer-Verlag, 2014. — P. 121–132.
67. Galov I. Compositions of Personal Smart Spaces in Multi-Blogging [Текст] / I. Galov, D. Korzun // Proc. 10th Conf. of Open Innovations Association FRUCT and 2nd Finnish–Russian Mobile Linux Summit. — SUAI, 2011. — P. 54–58.
68. Galov, I. Design of semantic information broker for localized computing environments in the Internet of Things [Текст] / I. Galov, A. Lomov,

D. Korzun // Proc. 17th Conf. of Open Innovations Association FRUCT. — ITMO Univeristy, IEEE, 2015. — P. 36–43.

69. Galov, I. Event Recording in Smart Room [Текст] / I. Galov, R. Kadirov, A. Vasilev, D. Korzun // Proc. 13th Conf. of Open Innovations Association FRUCT and 2nd Seminar on e-Tourism for Karelia and Oulu Region / Ed. by S. Balandin, Ulia Trifonova. — SUAI, 2013. — P. 20–28.
70. Galov, I. Fault tolerance support of smart-m3 application on the software infrastructure level [Текст] / I. Galov, D. Korzun // Proc. 16th Conf. of Open Innovations Association FRUCT. — 2014. — P. 16–23.
71. Galov, I. The smartroom infrastructure: Service runtime reliability [Текст] / I. Galov, D. Korzun // Proc. 14th Conf. of Open Innovations Association FRUCT / Ed. by S. Balandin, Ulia Trifonova. — SUAI, 2013. — P. 188–189.
72. Gao, M. Personalisation in web computing and informatics: Theories, techniques, applications, and future research [Текст] / Min Gao, Kecheng Liu, Zhongfu Wu // Information Systems Frontiers. — 2010. — Vol. 12, no. 5. — P. 607–629.
73. Giang, N. K., Leung, V., Lea, R. On Developing Smart Transportation Applications in Fog Computing Paradigm [Текст] / N. K. Giang, V. Leung, R. Lea // Proceedings of the 6th ACM Symposium on Development and Analysis of Intelligent Vehicular Networks and Applications. — ACM, 2016. — P. 91–98.
74. Guarino, N. Conceptual modeling: Foundations and applications [Текст] / N. Guarino. — Springer-Verlag, 2009. — P. 52–67.
75. Guessoum, Z. Towards reliable multi-agent systems: An adaptive replication mechanism [Текст] / Z. Guessoum, J.-P. Briot, N. Faci, O. Marin // Multiagent and Grid Systems. — 2010. — Vol. 6, no. 1. — P. 1–24.
76. Gurtov, A. Secure Communication and Data Processing Challenges in the Industrial Internet [Текст] / A. Gurtov, M. Liyanage, D. Korzun // Baltic Journal of Modern Computing. — 2016. — Vol. 4. — No. 4. — P. 1058–1073.
77. Honkola, J. Smart-M3 information sharing platform [Текст] / J. Honkola,

H. Laine, R. Brown, O. Tyrkkö // Proc. IEEE Symp. Computers and Communications (ISCC'10). — IEEE Computer Society, 2010. — P. 1041–1046.

78. Huda, M. T. An agent oriented proactive fault-tolerant framework for grid computing / M. T. Huda, H. W. Schmidt, I. D. Peake // First International Conference on e-Science and Grid Computing. — IEEE, 2005. — P. 3–11.
79. Janhunen, T. Meta programming with answer sets for smart spaces [Текст] / T. Janhunen, V. Luukkala // Proc. 6th Int'l Conf. Web Reasoning and Rule Systems (RR 2012). — Springer, 2012. — P. 106–121.
80. Kang, B. Internet of Everything: A large-scale autonomic IoT gateway [Текст] / B. Kang, D. Kim, H. Choo // IEEE Transactions on Multi-Scale Computing Systems. — 2017. — No. 99. — P. 1–10.
81. Kiljander, J. Knowledge sharing protocol for smart spaces [Текст] / J. Kiljander, F. Morandi, J.-P. Soininen // International Journal of Advanced Computer Science and Applications (IJACSA). — 2012. — Vol. 3, no. 9. — P. 100–110.
82. Kiljander, J. Semantic interoperability architecture for pervasive computing and Internet of Things [Текст] / J. Kiljander, A. D'Elia, F. Morandi [et al.] // IEEE Access. — 2014. — Vol. 2. — P. 856–874.
83. Kim, H. A Toolkit for Construction of Authorization Service Infrastructure for the Internet of Things [Текст] / H. Kim, E. Kang, E. A. Lee, D. Broman // Proceedings of the Second International Conference on Internet-of-Things Design and Implementation. — ACM, 2017. — P. 147–158.
84. Klyne, G. Resource description framework (rdf): Concepts and abstract syntax [Текст] / G. Klyne, J. J Carroll. — 2006.
85. Korolev, Y. Smart space applications integration: A mediation formalism and design for Smart-M3 [Текст] / Yuri Korolev, D. Korzun, I. Galov // Proc. 12th Int'l Conf. Next Generation Wired/Wireless Networking (NEW2AN 2012) and 5th Conf. Internet of Things and Smart Spaces (ruSMART 2012). — LNCS 7469. — Springer-Verlag, 2012. — P. 128–139.
86. Kortuem, G. Smart objects as building blocks for the Internet of Things

[Текст] / G. Kortuem, F. Kawsar, V. Sundramoorthy, D. Fitton // IEEE Internet Computing. — 2010. — Vol. 14, no. 1. — P. 44–51.

87. Korzun, D. Deployment of Smart Spaces in Internet of Things: Overview of the design challenges [Текст] / D. Korzun, S. Balandin, Andrei Gurtov // Proc. 13th Int'l Conf. Next Generation Wired/Wireless Networking and 6th Conf. on Internet of Things and Smart Spaces (NEW2AN/ruSMART 2013), LNCS 8121. — Springer-Verlag, 2013. — P. 48–59.
88. Korzun, D. Development of smart room services on top of Smart-M3 [Текст] / D. Korzun, I. Galov, S. Balandin // Proc. 14th Conf. of Open Innovations Association FRUCT / Ed. by S. Balandin, Ulia Trifonova. — SUAI, 2013. — P. 37–44.
89. Korzun D. G. Integration of Smart-M3 applications: Blogging in smart conference [Текст] / D. G. Korzun, I. V. Galov, A. M. Kashevnik [et al.] // Proc. 4th Conf. Smart Spaces (ruSMART'11) and 11th Int'l Conf. Next Generation Wired/Wireless Networking (NEW2AN'11). — Springer-Verlag, 2011. — P. 51–62.
90. Korzun, D. On the smart spaces approach to semantic-driven design of service-oriented information systems [Текст] / D. Korzun // International Baltic Conference on Databases and Information Systems. — Springer International Publishing, 2016. — P. 181–195.
91. Korzun, D. G. Overview of Smart-M3 principles for application development [Текст] / D. G. Korzun, S. I. Balandin, V. Luukkala [et al.] // Proc. Congress on Information Systems and Technologies (IS&IT'11), Conf. Artificial Intelligence and Systems (AIS'11). — Vol. 4. — Moscow: Physmathlit, 2011. — P. 64–71.
92. Korzun, D. G. Proactive personalized mobile mutli-blogging service on Smart-M3 [Текст] / D. G. Korzun, I. V. Galov, S. I. Balandin // Journal of Computing and Information Technology. — 2012. — Vol. 20, no. 3. — P. 175–182.
93. Korzun, D. G. Smart Space Deployment in Wireless and Mobile Settings of the Internet of Things [Текст] / D. G. Korzun, I. V. Galov, A. A. Lomov // IDAACS

2016. — 2016. — P. 86–91.

94. Korzun, D. The Smart-M3 platform: Experience of smart space application development for Internet of Things [Текст] / D. Korzun, Alexey Kashevnik, S. Balandin, A. Smirnov // Internet of Things, Smart Spaces, and Next Generation Networks and Systems. Proc. 15th Int'l Conf. Next Generation Wired/Wireless Networking and 8th Conf. on Internet of Things and Smart Spaces (NEW2AN/ruSMART 2015), LNCS 9247. — Springer, 2015. — P. 56–67.
95. Korzun, D. Virtual shared workspace for smart spaces and M3-based case study [Текст] / D. Korzun, I. Galov, A. Kashevnik, S. Balandin // Proc. 15th Conf. of Open Innovations Association FRUCT. — ITMO Univeristy, 2014. — P. 60–68.
96. Kubitza, T. Towards a toolkit for the rapid creation of smart environments [Текст] / T. Kubitza, A. Schmidt // International Symposium on End User Development / Springer. — 2015. — P. 230–235.
97. Kulakov, K. Ontological model for storage historical and trip planning information in smart space [Текст] / K. Kulakov, O. Petrina // Proc. 17th Conf. Open Innovations Framework Program FRUCT. — ITMO Univeristy, 2015. — P. 96–103.
98. Latchoumy, P. Survey on fault tolerance in grid computing [Текст] / P. Latchoumy, P. S. A. Khader // International Journal of Computer Science and Engineering Survey. — 2011. — Vol. 2, no. 4. — P. 97.
99. Luukkala, V. Enhancing a smart space with answer set programming [Текст] / V. Luukkala, I. Niemelä // Proc. 2010 Int'l Conf. Semantic web rules (RuleML'10). — Springer-Verlag, 2010. — P. 89–103.
100. Lynch, D. A proactive approach to semantically oriented service discovery [Текст] / D. Lynch, J. Keeney, D. Lewis, D. O'Sullivan. — 2006.
101. Lyu, M. R. Handbook of software reliability engineering [Текст] / M. R. Lyu [et al.]. — IEEE computer society press CA, 1996. — Vol. 222.
102. Mitrović D. Agent-based approaches to managing fault-tolerant networks of distributed multi-agent systems [Текст] / D. Mitrović, Z. Budimac, M. Ivanović,

M. Vidaković // Multiagent and Grid Systems. — 2011. — Vol. 7, no. 6. — P. 203–218.

103. Morandi, F. RedSib: a Smart-M3 semantic information broker implementation [Текст] / F. Morandi, L. Roffia, A. D'Elia [et al.] // Proc. 12th Conf. of Open Innovations Association FRUCT and Seminar on e-Tourism. — SUAI, 2012. — P. 86–98.

104. Moustafa, H. Remote monitoring and medical devices control in ehealth [Текст] / H. Moustafa, E. M Schooler, G. Shen, S. Kamath // 12th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob). — IEEE, 2016. — P. 1–8.

105. Murth, M. Knowledge-based interaction patterns for semantic spaces [Текст] / Martin Murth, Eva Kühn // Proc. the 2010 Int'l Conf. on Complex, Intelligent and Software Intensive Systems (CISIS '10). — IEEE Computer Society, 2010. — P. 1036–1043.

106. Nixon, L. J. B. Tuplespace-based computing for the semantic web: A survey of the state-of-the-art [Текст] / L. J. B. Nixon, E. Simperl, R. Krummenacher, F. M.-Recuerda // Knowl. Eng. Rev. — 2008. — Vol. 23, no. 2. — P. 181–212.

107. Ovaska, E. The design principles and practices of interoperable smart spaces [Текст] / E.Ovaska, T. S. Cinotti, A. Toninelli // Advanced Design Approaches to Emerging Software Systems: Principles, Methodology and Tools. — IGI Global, 2012. — P. 18–47.

108. Pellegrino L. A distributed publish/subscribe system for RDF data [Текст] / L. Pellegrino, F. Huet, F. Baude, A. Alshabani // Data Management in Cloud, Grid and P2P Systems, LNCS 8059. — Springer-Verlag, 2013. — P. 39–50.

109. Perera, C., Zaslavsky, A., Christen, P., Georgakopoulos, D. Context aware computing for the internet of things: A survey [Текст] / C. Perera, A. Zaslavsky, P. Christen, D. Georgakopoulos // IEEE Communications Surveys & Tutorials. — 2014. — Vol. 16. — No. 1. — P. 414-454.

110. Qian, J. Jtangcsps: A composite and semantic publish/subscribe system over

structured p2p networks [Текст] / J. Qian, J. Yin, J. Dong, D. Shi // Engineering Applications of Artificial Intelligence. — 2011. — Vol. 24, no. 8. — P. 1487–1498.

111. Razzaque, M. A. Middleware for internet of things: a survey [Текст] / M. A. Razzaque, M. M.-Jevric, A. Palade, S. Clarke // IEEE Internet of Things Journal. — 2016. — Vol. 3, no. 1. — P. 70–95.

112. Samoryadova A. M3-weather: A Smart-M3 world-weather application for mobile users [Текст] / A. Samoryadova, I. Galov, P. Borovinskiy [et al.] // Proc. 8th Conf. of Open Innovations Framework Program FRUCT — SPb: SUAI, 2010. — P. 160–166.

113. Sathish, S. Supporting smart space infrastructures: a dynamic context-model composition framework [Текст] / S. Sathish, C. di Flora // Proc. 3rd Int'l Conf. on Mobile Multimedia Communications / ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). — 2007. — P. 67.

114. Shi, D. An rdf-based publish/subscribe system [Текст] / D. Shi, J. Yin, Y. Li [et al.] // Semantics, Knowledge and Grid, Third International Conference on / IEEE. — 2007. — P. 342–345.

115. Smirnov A. Anonymous agent coordination in smart spaces: State-of-the-art [Текст] / A. Smirnov, A. Kashnevik, N. Shilov [et al.] // Smart Spaces and Next Generation Wired/Wireless Networking. Proc. 9th Int'l Conf. NEW2AN'09 and 2nd Conf. on Smart Spaces ruSMART 2009. LNCS 5764. — Berlin, Heidelberg : Springer-Verlag, 2009. — P. 42–51.

116. Tapia D. I. Agents and ambient intelligence: case studies [Текст] / D. I. Tapia, A. Abraham, J. M. Corchado, R. S. Alonso // J. Ambient Intelligence and Humanized Computing. — 2010. — Vol. 1, no. 2. — P. 85–93.

117. Teixeira, T. Service oriented middleware for the internet of things: a perspective / T. Teixeira, S. Hachem, V. Issarny, N. Georgantas // Towards a Service-Based Internet. — 2011. — P. 220–229.

118. Vasilev, A. Mechanism for context-aware substitution of Smart-M3 agents

based on dataflow network model [Текст] / A. Vasilev, I. Paramonov, S. Balandin [et al.] // Proc. Int'l Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT) / IEEE. — 2013. — P. 113–117.

119. Vdovenko, A., Korzun, D. Active control by a mobile client of subscription notifications in smart space [Текст] / A. Vdovenko, D. Korzun // Proceedings of the 16th Conference of Open Innovations Association FRUCT. — SPb.: ITMO University. — 2014. — P. 123–128.

120. Viola, F. A modular lightweight implementation of the smart-m3 semantic information broker [Текст] / F. Viola, A. D'Elia, L. Roffia, T. S. Cinotti // Open Innovations Association and Seminar on Information Security and Protection of Information Technology (FRUCT-ISPIT), 2016 18th Conference of / IEEE. — 2016. — P. 370–377.

121. Viola, F. The M3 Architecture for Smart Spaces: Overview of Semantic Information Broker Implementations [Текст] / F. Viola, A. D'Elia, D. Korzun, I. Galov, A. Kashevnik, S. Balandin // Proc. 19th Conf. of Open Innovations Association FRUCT. — 2016. — P. 264–272.

122. Wang, J. An agent-based hybrid service delivery for coordinating internet of things and 3rd party service providers [Текст] / J. Wang, Q. Zhu, Y. Ma // Journal of Network and Computer Applications. — 2013. — Vol. 36, no. 6. — P. 1684–1695.

123. Wang, J. A semantic-aware publish/subscribe system with rdf patterns [Текст] / J. Wang, B. Jin, J. Li, D. Shao // Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International / IEEE. — 2004. — P. 141–146.

124. Wang, X. Semantic space: An infrastructure for smart spaces [Текст] / X. Wang, J. S. Dong, C. Chin [et al.] // Computing. — 2002. — Vol. 1, no. 2. — P. 67–74.

125. Wu, X. Data mining with big data [Текст] / X. Wu, X. Zhu, G. Wu, Wei Ding // IEEE Transactions on Knowledge and Data Engineering. — 2014. — Vol. 26,

no. 1. — P. 97–107.

126. Xie, W. Smart platform — a software infrastructure for smart space (SISS) [Текст] / W. Xie, Y. Shi, G. Xu, Y. Mao // Proc. 4th IEEE Int'l Conf. on Multimodal Interfaces (ICMI '02). — IEEE Computer Society, 2002. — P. 429–434.

127. Yi, S. A survey of fog computing: concepts, applications and issues [Текст] / S. Yi, C. Li, Q. Li // Proceedings of the 2015 Workshop on Mobile Big Data. — ACM, 2015. — P. 37–42.

128. Yu, L. A Developer's Guide to the Semantic Web. [Текст] / Liyang Yu. — Springer, 2011.

129. Yusupov, R. M. Models and hardware-software solutions for automatic control of intelligent hall [Текст] / R. M. Yusupov, A. L. Ronzhin, M. V. Prishchepa, Al. L. Ronzhin // Autom. Remote Control. — 2011. — Vol. 72, no. 7. — P. 1389–1397.

Приложение А

Регистрация программ для ЭВМ

Свидетельства о государственной регистрации программы для ЭВМ.

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2015615091

Программный комплекс опорных сервисов управления программой и материалами докладчиков для проведения мероприятий коллаборативной деятельности типа «конференция» в интеллектуальном зале SmartRoom в составе: Conference-service и Content-service

Правообладатель: ***Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Петрозаводский государственный университет» (RU)***

Авторы: ***Галов Иван Викторович (RU), Марченков Сергей Александрович (RU), Корзун Дмитрий Жоржевич (RU)***

Заявка № **2015610185**

Дата поступления **12 января 2015 г.**

Дата государственной регистрации в Реестре программ для ЭВМ ***07 мая 2015 г.***

Врио руководителя Федеральной службы по интеллектуальной собственности

Л.Л. Кирий

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2017615801

Программа, реализующая информационное хранилище CuteSIB, для организации косвенного взаимодействия агентов интеллектуального пространства в ресурсно-ограниченных вычислительных средах.

Правообладатель: ***Федеральное государственное бюджетное образовательное учреждение высшего образования «Петрозаводский государственный университет» (RU)***

Авторы: ***Галов Иван Викторович (RU), Ломов Александр Андреевич (RU), Корзун Дмитрий Жоржевич (RU)***

Заявка № **2017610901**
Дата поступления **02 февраля 2017 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***24 мая 2017 г.***

Руководитель Федеральной службы по интеллектуальной собственности

Г.П. Ивлиев

Приложение Б

Акты внедрения

1) Акт об использовании результатов кандидатской диссертационной работы для разработки программных агентов экспериментальных образцов сервисов, используемых ООО “Опти-Софт”.

ООО "Опти-Софт"

Пр. Ленина, д. 31, г. Петрозаводск, Республика Карелия, 185910
тел. (814 2) 71-32-10, 71-32-22, 71-32-32, факс: (814 2) 71-32-16
e-mail: ashabaev@opti-soft.ru

ОКПО 6568922, ОГРН 1101001003641, ИНН/КПП 1001232729\100101001

_____№ ОС-1

На № _____ от _____

А К Т

об использовании результатов
кандидатской диссертационной работы

Галова Ивана Викторовича

Директор Шабаев Антон Игоревич составил настоящий акт о том, что результаты диссертационной работы «**Модели проектирования программной инфраструктуры интеллектуального пространства для ресурсно-ограниченных вычислительных сред**» использованы в рамках внедрения результатов выполнения проекта прикладных научных исследований по федеральной целевой программе «Исследования и разработки по приоритетным направлениям развития научно-технологического комплекса России на 2014-2020 годы». Научные исследования проводились в 2014-2016 гг. по Соглашению № 14.574.21.0060 (RFMEFI57414X0060) от 30.06.2014 по теме "Разработка технологии интеллектуализации локализованных вычислительных сред Интернета физических устройств для персонализированного построения и упреждающей доставки сервисов".

Полученный И. В. Галовым *метод разработки программной инфраструктуры интеллектуального пространства для построения информационных сервисов в условиях разнообразия, динамики и нестабильности ресурсно-ограниченных IoT-сред* был использован при разработке и экспериментальном исследовании следующих экспериментальных образцов сервисов: 1) окружения для коллаборативной деятельности для проведения мероприятий вида «конференция» и «совещание», 2) мобильного здравоохранения для удаленного персонализированного предоставления услуг медицинской информационной системы 3) электронного туризма для персонализированного планирования, отслеживания и адаптации туристических поездок, 4) индустриального интернет для предоставления рекомендаций при планировании и управлении технологическими процессами.

Использование указанного метода позволило:
а) настроить информационное хранилище для развертывания экспериментальных образцов сервисов в заданных аппаратно-сетевых средах, соответствующих требуемым проблемным областям;
б) программно реализовать требуемую организацию взаимодействия в программном коде агентов с целью построения информационных сервисов при ограниченных ресурсах заданных аппаратно-сетевых сред;
в) запустить экспериментальные образцы сервисов на предоставленном экспериментальном стенде и обеспечить работоспособность программных агентов в условиях частых сбоев аппаратно-сетевой среды, достаточную для практического использования сервисов.

Директор ООО «Опти-Софт», к.т.н. А. И. Шабаев

"29" апреля 2017 г.

М.П.

2) Акт об использовании результатов кандидатской диссертационной работы для проведения секций конференций Ассоциации открытых инноваций FRUCT

Open Innovations Association FRUCT

CERTIFICATE OF DEPLOYMENT

for Ph.D. research application of

Galov Ivan Viktorovich

By this certificate of deployment, we confirm that results of the Ph.D. thesis research on "**Design models of smart space software infrastructure for resource-constrained computing environments**" by Galov I. V. have been used in the software infrastructure development of portable version of the SmartRoom system (http://oss.fruct.org/wiki/SmartRoom). Portable version of the system was used for sections management at the following International conferences of the FRUCT Association:

- The 15th International Conference of Open Innovations Association FRUCT (21-25 April 2014, Saint-Petersburg, Russia), Section "Smart-M3 Applications";
- The 16th International Conference of Open Innovations Association FRUCT (27-31 October 2014, Oulu, Finland), Section "Smart Spaces and Internet of Things I";
- The 17th International Conference of Open Innovations Association FRUCT (20-24 April 2015, Yaroslavl, Russia), Section "Smart Spaces and Internet of Things";
- The 18th International Conference of Open Innovations Association FRUCT (18-22 April 2016, Saint-Petersburg, Russia), Section "Smart Spaces and Internet of Things";
- The 19th International Conference of Open Innovations Association FRUCT (21-25 April 2014, Jyväskylä, Finland), Section "Smart Spaces and Internet of Things I".

The software infrastructure implementation of the portable SmartRoom system is essentially based on the following results of the Ph.D. research of Galov I. V.

1. The CuteSIB implementation of information storage was used for creation of the SmartRoom system smart space in a resource-constrained computing environment of a conference hall.
2. Application of the information storage access management model for software agents allowed setting up information storage for specific resource-constrained computing environment of the conference hall.
3. Application of the fault tolerance model for software infrastructure components allowed restoring information storage and software agents after various failures without interruption of conference sections and participants.
4. Application of the information notification system model allowed integrating additional services in portable version of the SmartRoom system including conference speeches discussion and culture program construction.

These results provided fault tolerance to portable version of the SmartRoom system in a resource-constrained computing environment of a conference hall in comparison with previous implementations of the SmartRoom system. The SmartRoom system did not need additional resources and equipment for running in comparison with typical equipment of conference hall.

22 June 2017

/ Sergey I. Balandin /
Ph.D., Adjunct Professor
President of FRUCT Association

Перевод акта об использовании результатов кандидатской диссертационной работы для проведения секций конференций Ассоциации открытых инноваций FRUCT.

Open Innovations Association FRUCT

АКТ

об использовании результатов
кандидатской диссертационной работы
Галова Ивана Викторовича

Данным актом подтверждается, что результаты кандидатской диссертационной работы Галова И. В. **«Модели проектирования программной инфраструктуры интеллектуального пространства для ресурсно-ограниченных вычислительных сред»** были использованы для разработки программной инфраструктуры с целью создания и применения переносного варианта системы SmartRoom (http://oss.fruct.org/wiki/SmartRoom). Переносной вариант системы был применен в рамках экспериментальной апробации при проведении секционных докладов на следующих международных конференциях Ассоциации открытых инноваций FRUCT:

- 15-я международная конференция открытых инноваций Ассоциации FRUCT (21-25 апреля 2014, г. Санкт-Петербург, Россия), секция "Smart-M3 Applications";
- 16-я международная конференция открытых инноваций Ассоциации FRUCT (27-31 октября 2014, г. Оулу, Финляндия), секция "Smart Spaces and Internet of Things I";
- 17-я международная конференция открытых инноваций Ассоциации FRUCT (20-24 апреля 2015, г. Ярославль, Россия), секция "Smart Spaces and Internet of Things";
- 18-я международная конференция открытых инноваций Ассоциации FRUCT (18-22 апреля 2016, г. Санкт-Петербург, Россия), секция "Smart Spaces and Internet of Things";
- 19-я международная конференция открытых инноваций Ассоциации FRUCT (9-11 ноября 2017, г. Йювяскюля, Финляндия), секция "Smart Spaces and Internet of Things I".

Разработанная Галовым И.В. программная инфраструктура для переносного варианта системы SmartRoom существенно основана на следующих научно-технических решениях, полученных в ходе диссертационного исследования Галова И. В. и представленных в его диссертационной работе.

1. Программная реализация CuteSIB для информационного хранилища использовалась при создании интеллектуального пространства системы SmartRoom в ресурсно-ограниченной вычислительной среде конференц-зала.
2. Использование модели управления сетевым доступом программных агентов к информационному хранилищу позволило настроить информационное хранилище под возможности аппаратно-сетевого комплекса в заданной ресурсно-ограниченной вычислительной среде конференц-зала.
3. Использование модели обеспечения устойчивости компонентов программной инфраструктуры к сбоям позволило восстанавливать работу информационного хранилища и программных агентов после разнообразных сбоев, без прерывания секционных докладов и работы участников конференции.
4. Использование модели системы информационных уведомлений позволило интегрировать в переносной вариант системы SmartRoom дополнительные информационные сервисы, включая обсуждение участниками программы докладов и построение участниками культурной программы.

В целом, эти результаты обеспечили возможность устойчивой работы переносного варианта системы SmartRoom в ресурсно-ограниченной вычислительной среде конференц-зала в сравнении с предыдущими версиями реализации системы SmartRoom. Для запуска и работы переносного варианта системы SmartRoom не потребовались дополнительные затраты для усиления аппаратно-сетевого комплекса по сравнению с типовым оборудованием конференц-зала.

22 июня 2017 г.

С. И. Баландин
к.т.н., адъюнкт-профессор
Президент Ассоциации FRUCT

Перевод верен
23.06.

Отдел сопровождения академической мобильности Сектор протокола

3) Акт об использовании результатов кандидатской диссертационной работы в рамках курса “Интеллектуальные сетевые пространства”.

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Петрозаводский государственный университет» (ПетрГУ)

"УТВЕРЖДАЮ"
директор института математики
и информационных технологий,
Светова Нина Юрьевна
“23” июня 2017 г.

А К Т
об использовании результатов
кандидатской диссертационной работы
Галова Ивана Викторовича

Комиссия в составе: председателя Семеновой Е. Е., заведующего кафедрой информатики и математического обеспечения Богоявленского Ю. А. и доцента Корзуна Д. Ж. составили настоящий акт о том, что результаты диссертационной работы «**Модели проектирования программной инфраструктуры интеллектуального пространства для ресурсно-ограниченных вычислительных сред»** используются при проведении лабораторных работ в рамках учебной дисциплины «Интеллектуальные сетевые пространства» для магистратуры по направлениям «прикладная математика и информатика» и «информационные системы и технологии», начиная с 2014/2015 уч. года.

Результаты работы используются при решении обучающимися следующих задач.
1. Запуск и настройка информационного хранилища при развертывании программной инфраструктуры интеллектуального пространства в заданной вычислительной среде.
2. Разработка программных агентов в части реализации модулей, отвечающих за организацию косвенного взаимодействии при совместном построении информационных сервисов в интеллектуальном пространстве.

Такой вариант использования научно-технических результатов позволяет обеспечить обучающихся знаниями и навыками разработки интеллектуальных пространств для различных проблемных областей и для различных сетевых вычислительных сред.

Председатель методической
комиссии института:
доцент каф. прикладной математики
и кибернетики, к.ф.-м.н. Семенова Е. Е.

Члены комиссии:
зав.каф. информатики
и математического обеспечения,
доцент, к.т.н. Богоявленский Ю. А.

доцент каф. информатики
и математического обеспечения,
вед. науч. сотр., к.ф.-м.н. Корзун Д. Ж.

Приложение В

Сводные характеристики комплекса программных средств

1) Информационное хранилище CuteSIB (свидетельство о регистрации программы для ЭВМ № 2017615801 от 24.05.2017):

- Строк кода/комм.: 7687 / 5651;
- Вклад автора: 50% (обработчики операций, взаимодействие с БД троек);
- Технологии: язык C++, фреймворк Qt, библиотека librdf.

2) Инфраструктурные агенты системы проведения мероприятий совместной деятельности (свидетельство о регистрации программы для ЭВМ № 2015615091 от 07.05.2015):

Сервис	Строк кода/ комм.	Вклад автора	Технологии
Conference-service	3555/2354	70%	Qt / C++, smartslog
Presentation-service	3049/2119	100%	Qt / C++, smartslog
Content-service	1126/845	100%	Python, webpy, libpoppler

3) ПО для проведения экспериментального исследования:

- Реальные и имитируемые агенты: встроенные сенсоры на ресурсно-ограниченных СВУ, обработчики (агрегаторы) на персональных ЭВМ, клиенты на персональных мобильных СВУ;
- Технологии: Python, Qt / C++, valgrind.