ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ПЕТРОЗАВОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

На правах рукописи

Марченков Сергей Александрович

АВТОМАТИЗИРОВАННАЯ РАЗРАБОТКА ИНТЕРОПЕРАБЕЛЬНОЙ ПРОГРАММНОЙ ИНФРАСТРУКТУРЫ ДЛЯ ОРГАНИЗАЦИИ СОВМЕСТНО ИСПОЛЬЗУЕМОГО ИНФОРМАЦИОННОГО ИНТЕРНЕТ-ОКРУЖЕНИЯ

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

Диссертация на соискание учёной степени
кандидата технических наук

Научный руководитель:
кандидат физико-математических наук, доцент
Корзун Дмитрий Жоржевич

Петрозаводск — 2019

Оглавление

Стр.

Стр.

Введение

Актуальность темы. Совместно используемое информационное интернет-окружение (СИИО) поддерживает обмен информацией в коллективе людей при решении ими общей задачи из некоторой предметной области. Человеку предоставляются цифровые сервисы, ускоряющие информационную поддержку (обмен и анализ) за счет использования доступных в СИИО ресурсов, в том числе и экспертных ресурсов самого человека. Развитие технологий СИИО и их применение формируют актуальное направление процесса цифровизации различных сфер жизни общества, включая экономику, науку, образование, культуру. В СИИО интегрируются информационные, технические и экспертные ресурсы, организуется совместный доступ к общей информации, обеспечивается контекстная осведомленность человека. Для интерактивной мультимедийной информационной поддержки и видеоконференцсвязи по обмену экспертными знаниями используются широкоформатные экраны, аудиосистемы помещения, персональные мобильные устройства пользователей. Для обработки данных и извлечения из них информации используется локальное вычислительное и сетевое оборудование в соответствии с подходами периферийных и туманных вычислений. Ресурсы являются неоднородными и динамичными, что затрудняет их интеграцию при построении сервиса и увеличивает трудозатраты на разработку программного обеспечения (ПО). Областью данного исследования выступает автоматизированная разработка ПО, выполняющего интеграцию ресурсов при построении сервиса в СИИО.

На методы разработки ПО для СИИО влияет концепция окружающего интеллекта и соответствующих технологий искусственного интеллекта (ИИ). Возрастает роль средств автоматизированной разработки ПО для использования технологий ИИ. Информационная поддержка пользователей усиливается за счет контекстных сервисов, учитывающих состояние пользователя и имеющиеся ресурсы. Сервисы приобретают свойства «интеллектуальности»: распознавание текущей ситуации, анализ поведения человека, предоставление информации в ненавязчивой форме. Для реализации этих свойств используют известные методы многоагентных систем и технологии Семантического веба, что приводит к следующим проблемам: а) нет достаточного (для имеющегося разнообразия ресурсов) метода разработки ПО с интеграцией динамичных и неоднородных ресурсов в

СИИО; б) нет достаточных (для имеющегося разнообразия предметных областей) моделей и шаблонов проектирования контекстных сервисов, где сервис строится как многоагентная система с информационно-управляемым взаимодействием на основе технологий Семантического веба; в) нет развитых (с существенной долей кодогенерации) средств программирования взаимодействия агентов, требуемого для построения сервиса.

В диссертационной работе предлагается использовать подход интеллектуальных пространств (ИП) для создания СИИО. Построение сервиса выполняется программными агентами, каждый отвечает за определенный ресурс в СИИО. В соответствии с подходом ИП агенты создают и поддерживают общее информационное содержимое для семантического связывания ресурсов. В нем представлены виртуальные цифровые образы (двойники) имеющихся ресурсов, пользователей и происходящих процессов совместного использования информации. Управление информационным содержимым основано на интероперабельной программной инфраструктуре, отвечающей за организацию информационно-управляемого взаимодействия агентов. Существующие методы разработки программной инфраструктуры приводят к значительным трудозатратам на создание и сопровождение ПО. Необходим метод автоматизированной разработки программной инфраструктуры, обеспечивающий семантическую интероперабельность агентов при построении ими сервисов на основе интеграции ресурсов в СИИО.

Степень разработанности темы. Исследование вопросов создания СИИО ведется в России и за рубежом. Область является междисциплинарной – изучаются свойства информационного обмена и оперативного анализа информации человеком с целью создания и применения цифровых технологий информационной поддержки деятельности людей в различных предметных областях. Развитию методологических основ разработки ПО для технологий информационной сервис-ориентированной поддержки способствовали труды российских и зарубежных ученых: С.И. Баландина, С. Болдырева, В.А. Васенина, В.И. Городецкого, А.М. Кашевника, Ю. Кильяндера, Д.Ж. Корзуна, Д. Кук, Я. Оливера, А.Л. Ронжина, А.В. Смирнова, А.Н. Терехова, Р.М. Юсупова, Т. Чинотти и др.

Целью диссертационной работы является повышение эффективности разработки программной инфраструктуры совместно используемого информационного интернет-окружения за счет: а) унифицированного моделирования сервиса как системы взаимодействующих агентов и б) автоматизированного программирования взаимодействия на основе кодогенерации.

Для достижения этой цели исследуются и решаются следующие **задачи**:

1. Выполнить анализ результатов современных исследований в области информационного обмена и анализа. Провести обзор существующих решений для совместно используемых информационных интернет-окружений, а также моделей проектирования сервисов, алгоритмов автоматизации программирования агентов и методов разработки программных инфраструктур. Сформулировать основные требования к разработке ПО, выполняющего интеграцию ресурсов.
2. Сформировать метод разработки интероперабельной программной инфраструктуры СИИО, обеспечивающей информационно-управляемое взаимодействие агентов для интеграции динамических и неоднородных ресурсов при построении сервиса.
3. Предложить концептуальную модель сервиса СИИО, позволяющую описывать с помощью онтологии варианты информационно-управляемого взаимодействия агентов для построения контекстных сервисов и их композиции на основе технологий Семантического веба.
4. Разработать алгоритм автоматизации программирования взаимодействия агентов СИИО, обеспечивающий кодогенерацию программных механизмов информационно-управляемого взаимодействия по заданной спецификации проектирования сервиса.
5. Разработать набор предметно-ориентированных моделей проектирования сервисов СИИО в качестве шаблонных решений для востребованных приложений.
6. Реализовать комплекс программных средств в составе: а) генератор программного кода взаимодействия агентов на основе полученных алгоритма автоматизации программирования и концептуальной модели сервиса; б) экспериментальные образцы предметно-ориентированных сервисов для вычислительных сред интеллектуального зала, умного музея и промышленного предприятия. Выполнить экспериментальное исследование на основе комплекса программных средств с целью оценки эффективности сформированного метода разработки программной инфраструктуры СИИО.

Объектом исследования является интероперабельная программная инфраструктура, обеспечивающая организацию информационно-управляемого взаимо-

действия агентов при построении ими сервисов на основе интеграции ресурсов вычислительной среды.

Предметом исследования являются модели проектирования сервисов как многоагентных систем, использующих технологии Семантического веба для информационного обмена, и алгоритмы автоматизации программирования информационно-управляемого взаимодействия, обеспечивающие кодогенерацию для реализации программ агентов.

Научная новизна диссертационной работы состоит в следующем:

1. Предложен метод разработки интероперабельной программной инфраструктуры СИИО, отличающийся возможностью унифицированной и автоматизированной разработки сервиса как системы с информационно-управляемым взаимодействием агентов для интеграции динамических и неоднородных ресурсов при построении сервиса.
2. Предложена концептуальная модель информационного сервиса СИИО, отличающаяся возможностью онтологического описания информационно-управляемого взаимодействия агентов для построения контекстных сервисов и их композиции на основе технологий Семантического веба.
3. Предложен алгоритм автоматизации программирования взаимодействия агентов СИИО, отличающийся возможностью кодогенерации программных механизмов информационно-управляемого взаимодействия для построения сервиса, в дополнение к структурам данных предметной области.
4. Предложен набор предметно-ориентированных моделей проектирования сервисов СИИО, отличающихся предоставлением разработчику архитектурных и поведенческих абстракций информационно-управляемого взаимодействия агентов как шаблонных решений для востребованных приложений в составе: а) распознавание присутствия и анализ активности пользователей, б) сопровождение и визуализация плана деятельности людей, в) совместное пополнение информационного содержимого знаниями о предметной области, г) мониторинг объектов физической среды.
5. Разработан комплекс программных средств в соответствии с предложенными новым методом разработки программной инфраструктуры,

моделями проектирования сервисов и алгоритмом автоматизации программирования взаимодействия агентов.

Теоретическая и практическая значимость исследования. Предложенный в работе метод разработки развивает научные основы построения многоагентных информационных систем в условиях периферийных и туманных вычислений за счет введения новых моделей проектирования сервисов и алгоритма автоматизации программирования взаимодействия агентов. Достигается снижение трудозатрат на разработку и сопровождение сервисов СИИО. Реализован генератор программного кода участвующих в построении сервисов взаимодействующих агентов на основе онтологии предметной области и онтологии спецификации сервисов, который позволяет снизить трудозатраты на разработку и сопровождение СИИО. Получены экспериментальные образцы ПО и оценки их эффективности, допускающие практическое использование при разработке СИИО для вычислительных сред интеллектуального зала, умного музея и промышленного предприятия.

Полученные результаты также используются: а) в учебном процессе при проведении лабораторных работ для дисциплины «Интеллектуальные сетевые пространства»; б) для производства и развития высокотехнологической продукции в виде цифровых сервисов при разработке программного обеспечения информационно-управляемого взаимодействия в условиях Интернета вещей и больших данных; в) в исследованиях, направленных на разработку программно-аппаратного комплекса многопараметрического мониторинга роботизированного производственного оборудования. Практическая значимость подтверждается актами о внедрении и об использовании результатов диссертационной работы (см. Приложение А).

Методы исследования. Результаты выполненных прикладных исследований и разработок основаны на методах программирования распределенных и многоагентных систем, методах онтологического моделирования и Семантического веба, технологиях интеллектуальных пространств и интеллектуальных Интернет-технологиях.

Основные положения, выносимые на защиту:

1. Метод разработки интероперабельной программной инфраструктуры СИИО для унифицированной и автоматизированной разработки сервиса как многоагентной системы, интегрирующей динамические и неоднородные ресурсы при построении сервиса.

2. Концептуальная модель информационного сервиса СИИО для проектирования вычислительного процесса построения контекстных сервисов и их композиции на основе технологий Семантического веба с информационно-управляемым взаимодействием агентов.
3. Алгоритм автоматизации программирования взаимодействия агентов СИИО для кодогенерации программных механизмов информационно-управляемого взаимодействия по заданной онтологии информационного сервиса.
4. Предметно-ориентированные модели проектирования сервисов для разработки СИИО на основе шаблонных решений для востребованных приложений: а) распознавание присутствия и анализ активности пользователей, б) сопровождение и визуализация плана деятельности людей, в) совместное пополнение информационного содержимого знаниями о предметной области, г) мониторинг объектов физической среды.
5. Комплекс программных средств для метода разработки программной инфраструктуры СИИО в составе: а) генератор программного кода взаимодействия агентов по онтологической модели информационного сервиса; б) экспериментальные образцы ПО предметно-ориентированных сервисов для вычислительных сред интеллектуального зала, умного музея и промышленного предприятия.

Степень достоверности результатов проведенных исследований. Достоверность научных положений, результатов и выводов диссертационной работы обеспечивается за счет анализа состояния исследований в области организации совместно используемых информационных интернет-окружений, согласованности теоретических выводов с результатами проведенного экспериментального исследования полученного комплекса программных средств, а также апробацией основных положений диссертации на научных конференциях, в научных работах и приравненных к ним публикациях. По результатам исследования получены 7 свидетельств о государственной регистрации программ для ЭВМ и 1 свидетельство о государственной регистрации базы данных.

Апробация результатов исследования. Основные научные результаты диссертационного исследования представлялись на международных и российских научных мероприятиях, в том числе: международные конференции ассоциации открытых инноваций FRUCT (Россия, Финляндия, 2013-2019 гг.,), международная IEEE конференция «Intelligent Data Acquisition and

Advanced Computing Systems: Technology and Applications (IDAACS)» (Румыния, Бухарест, 2017), научный семинар «Проблемы современных информационно-вычислительных систем» (Россия, Москва, 2017), всероссийская конференция «Цифровые технологии в образовании, науке, обществе» (Россия, Петрозаводск, 2017-2018 гг.), конференция «Современные технологии в теории и практике программирования» (Россия, Санкт-Петербург, 2013-2016 гг.).

Реализация результатов работы. Работа выполнена по Заданию № 2.5124.2017/8.9 «Фундаментальные вопросы моделирования и программирования информационно-управляемого взаимодействия в социо-кибер-физических системах в условиях Интернета вещей и больших данных» Минобрнауки России в рамках базовой части государственного задания ПетрГУ на 2017-2019 гг. Междисциплинарное научное исследование и разработка поддержаны в рамках проектов: ОГОН РФФИ № 16-01-12033 «Создание программной инфраструктуры для коллективного семантического аннотирования, связывания информации и персонализированного доступа к корпусу источников по истории повседневности», 2016-2017 гг.; ФЦП по соглашению № 14.574.21.0060 «Разработка технологии интеллектуализации локализованных вычислительных сред Интернета физических устройств для персонализированного построения и упреждающей доставки сервисов», 2014-2016 гг.; Задания № 2014/154 «Методы онтолого-ориентированной разработки и интеллектуальные Интернет-технологии для реализации семантических сервисов следующего поколения в области историко-культурного туризма» Минобрнауки России в рамках базовой части государственного задания (НИР № 1481), 2014-2016 гг. Работа поддержана Программой развития опорного университета для ПетрГУ на 2017-2021 гг. Полученные результаты используются в учебном процессе ПетрГУ, для производства и развития цифровых сервисов в ООО «Опти-Софт» и при разработке программно-аппаратного комплекса многопараметрического мониторинга роботизированного производственного оборудования.

Научные публикации. По теме диссертационого исследования опубликовано 28 научных работ и приравненных к ним публикаций, среди которых 3 работы в журналах из списка ВАК [21; 23; 24] и 9 работ в международных изданиях, индексируемых в реферативных базах Web of Science и Scopus [23; 43; 44; 85; 91; 96; 109; 119; 120], 7 свидетельств о государственной регистрации программ для ЭВМ и 1 свидетельство о государственной регистрации базы данных (см. Приложение Б).

Объем и структура работы. Диссертация состоит из введения, 4 глав, заключения, списка литературы, 2 приложений. Текст диссертации объемом 155 страниц содержит 32 рисунка и 9 таблиц. Библиографический список литературы содержит 125 наименований.

Во введении обосновывается актуальность темы исследования, определяется цель и решаемые задачи, формулируются положения, выносимые на защиту, их научная новизна, теоретическая и практическая значимость исследования.

В первой главе приводится концептуальная модель для содержательного описания структуры совместно-используемого информационного интернет-окружения. Раскрывается актуальность и востребованность СИИО в развитии процессов информатизации общества и связанных с этим сфер жизни общества (цифровая экономика, наука и образование, культура). Выполняется анализ результатов современных исследований в области информационного обмена и анализа на основе многоагентных систем и технологий Семантического веба. Приводится обзор существующих решений для организации СИИО, а также моделей проектирования сервисов, алгоритмов автоматизации программирования агентов и методов разработки интероперабельных программных инфраструктур для СИИО. Формулируются основные требования к разработке программной инфраструктуры СИИО, отвечающей за интеграцию ресурсов.

Во второй главе представлен новый метод разработки интероперабельной программной инфраструктуры СИИО для интеграции разнообразных ресурсов за счет унификации процесса разработки сервисов как системы взаимодействующих агентов с поддержкой автоматизации программирования взаимодействия агентов. Предлагается концептуальная модель информационного сервиса СИИО, предназначенная для проектирования сервиса как системы взаимодействующих агентов за счет создания унифицированного онтологического описания процессов построения сервиса с учетом контекста взаимодействующих агентов и задействованных ресурсов. Проектирование осуществляется за счет использования и расширения онтологического описания семантических веб-сервисов (онтология OWL-S) с использованием выделенных категорий сервисов и типовых моделей информационно-управляемого взаимодействия программных агентов при их построении. Предлагается алгоритм автоматизации программирования взаимодействия агентов на основе генерации программного кода с использованием онтологии предметной области и спецификации сервиса.

В третьей главе представлены предметно-ориентированные модели проектирования сервисов для выполнения прикладной разработки на основе шаблонных решений для следующих востребованных приложений СИИО: распознавание присутствия и анализ активности пользователей; сопровождение и визуализация плана деятельности людей; пополнение информационного содержимого знаниями о предметной области; мониторинг объектов физической среды. Представленные предметно-ориентированные модели проектирования сервисов используются разработчиком при выполнении первого этапа («разработка концепции необходимых сервисов») предложенного метода для снижения трудозатрат на проектирование за счет предоставления архитектурных и поведенческих абстракций взаимодействия агентов, а также частной онтологии сервиса.

В четвертой главе описывается программная реализация комплекса программных средств: а) реализация генератора программного кода взаимодействия агентов на основе полученного алгоритма автоматизации программирования и концептуальной модели сервиса; б) экспериментальные образцы предметно-ориентированных сервисов на основе метода разработки интероперабельной программной инфраструктуры для организации СИИО для вычислительных сред интеллектуального зала, умного музея и промышленного предприятия. На основе реализованного комплекса программных средств выполняются экспериментальные исследования с целью оценки эффективности метода разработки программной инфраструктуры, который позволяет снизить трудозатраты на создание сервисов как систем взаимодействующих агентов за счет использования унифицированной онтологии сервиса и генератора программного кода взаимодействия агентов.

Глава 1. Принципы разработки совместно используемых информационных интернет-окружений

В главе приводится концептуальная модель для содержательного описания структуры (множество понятий и связей между ними) совместно-используемого информационного интернет-окружения. Раскрывается актуальность и востребованность СИИО в развитии процессов информатизации общества и связанных с этим сфер жизни общества (цифровая экономика, наука и образование, культура). Выполняется анализ результатов современных исследований в области информационного обмена и анализа на основе многоагентных систем и технологий Семантического веба. Приводится обзор существующих решений для организации СИИО, а также моделей проектирования сервисов, алгоритмов автоматизации программирования агентов и методов разработки интероперабельных программных инфраструктур для СИИО. Формулируются основные требования к разработке программной инфраструктуры СИИО, отвечающей за интеграцию ресурсов. В целом, в главе показано, что требуется новый метод разработки программной инфраструктуры СИИО. В целом, в главе показано, что требуется новый метод разработки программной инфраструктуры СИИО.

1.1 Понятие и характеристики совместно используемого информационного интернет-окружения

СИИО поддерживает обмен информацией в группе людей при решении ими общей задачи из некоторой предметной области. Такое окружение используется группой или командой людей, которые должны быть вовлечены в процесс достижения общей цели (решение задач) и при этом взаимодействовать в некотором общем рабочем пространстве. Для достижения общего понимания и высокой степени осведомленности члены группы или команды должны предоставлять различную информацию и знания (например, знания о навыках, компетенциях, ролях и др.) для общего доступа и обмениваться ими. Также коммуникационное взаимодействие и деятельность внутри группы должны быть скоординированы. Таким образом, хоть СИИО использует результаты, полученные в области

распределенных вычислений, основным свойством является явное обеспечение осведомленности между участниками и их деятельностью [78], а не их разделение.

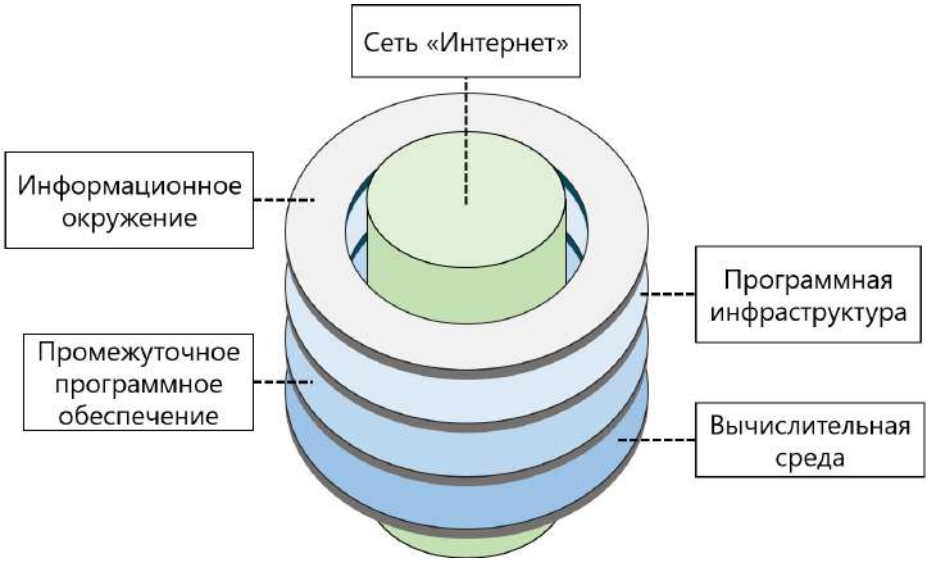


Рисунок 1.1 — Многоуровневая совокупность архитектурных структурных элементов для определения СИИО

В соответствии с рисунком 1.1 понятие СИИО выводится из многоуровневой совокупности архитектурных структурных элементов, характеризующих собой программно-аппаратный комплекс для совместного решения пользователями задач в рамках заданной предметной области. Нижележащим архитектурным слоем в СИИО выступает вычислительная среда, которая предоставляет совокупность программных и аппаратных средств, а также сетевых коммуникаций для реализации определенной распределенной концепции вычислений (см. раздел 1.4). Организация СИИО происходит в некоторой пространственно-ограниченной области, которое определяется для пользователей как общее рабочее пространство, которое объединяет внутри себя имеющиеся доступные ресурсы, а также предоставляет доступ к внешним ресурсам посредством сети Интернет. При этом пользователи имеют как локальный, так и удаленный доступ к СИИО. Следующий вышележащий слой, промежуточное ПО, направлен на реализацию определенных требований для организации СИИО (см. раздел 1.3), обеспечивая интеграцию разнообразных устройств и программных компонентов и организацию их сетевого взаимодействия на программном уровне (реализует общее рабочее пространство). Промежуточное ПО определяет программную инфраструктуру, которая обеспечивает функционирование СИИО, предоставляя

инструментальные средства программирования для реализации основных программных компонентов. Верхним слоем является информационное окружение, предназначенное «для автоматизации происходящих процессов, направленных на обеспечение информационной поддержки пользователей» [13; 36] посредством предоставления цифровых сервисов во время решения задач в СИИО.

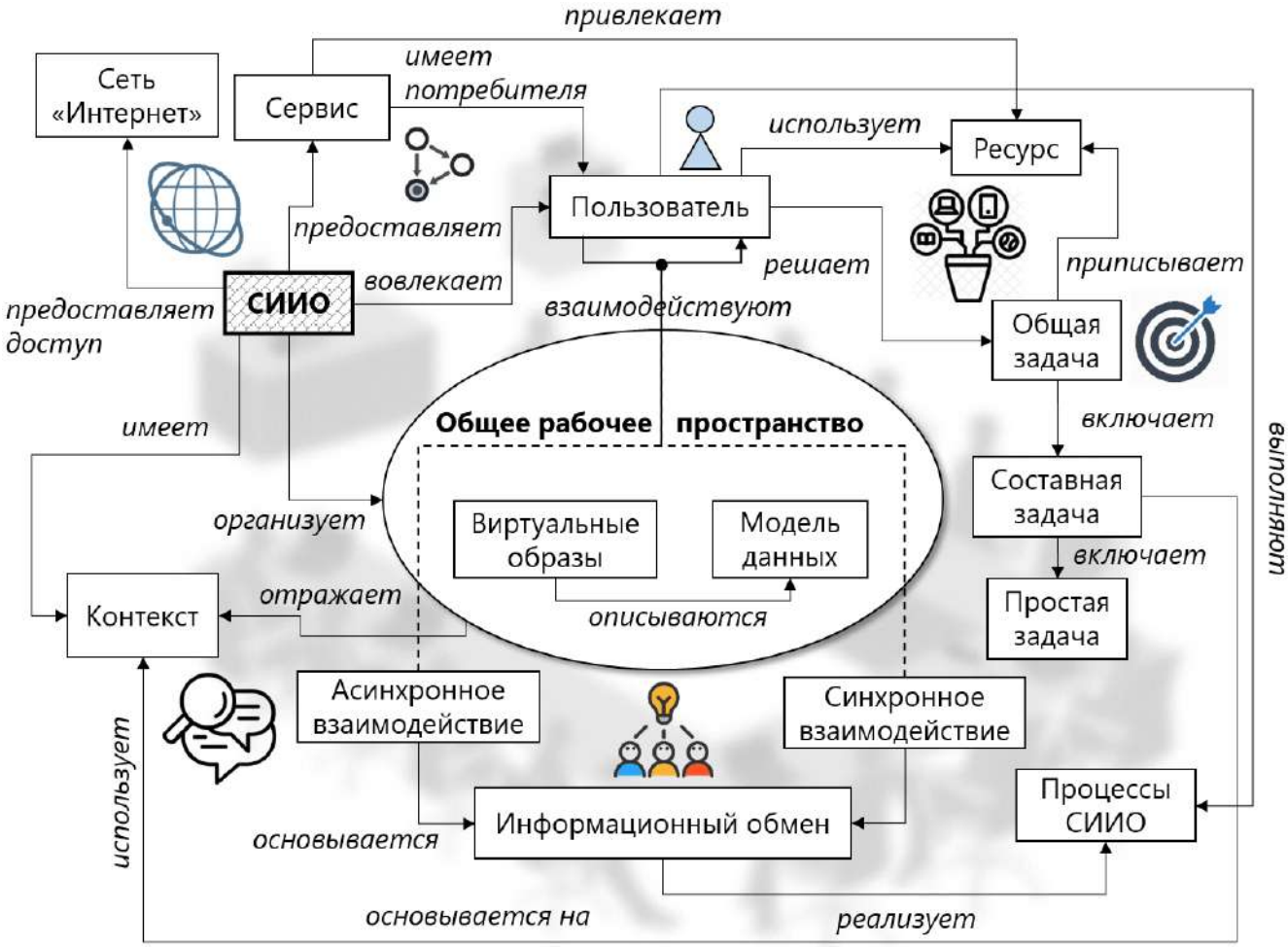


Рисунок 1.2 — Концептуальная модель совместно-используемого информационного интернет-окружения (СИИО)

В соответствии с рисунком 1.2 понятие СИИО описывается с помощью определяющих структурных понятий и отношений между ними. СИИО определяется как технологически оснащенное информационное окружение, предоставляющее множеству пользователей набор цифровых сервисов с доступом к сети Интернет для обеспечения возможности взаимодействовать друг с другом посредством совместного использования доступных ресурсов на основе информационного обмена при решении общей задачи. Центральным понятием в СИИО выступает общее рабочее пространство, которое реализует основные функциональные свойства [121].

В СИИО в качестве пользователей вовлекаются люди для решения ими общей задачи посредством обмена информацией. В соответствии с рисунком 1.3 обмен информацией реализуется на основе выполнения следующих базовых процессов, возникающих в СИИО: коммуникация, координация, сотрудничество. Процессы коммуникации направлены на обеспечение взаимодействия участников посредством обмена информацией. Процессы координации направлены на нахождение наилучшего способа организации выполняемой деятельности для решения задач на основе распределения и привлечения ресурсов. Процессы сотрудничества направлены на обеспечение выполнения совместных действий пользователей в общем рабочем пространстве, которые сопровождаются созданием и изменением различных артефактов совместной деятельности: электронные документы, схемы, расписание и др.

Важным требованием, которое должно удовлетворяться при выполнении процессов, выступает потребность пользователей в осведомленности [67]. Осведомленность обеспечивает способность воспринимать и осознавать события, общие объекты и пользователей, которые выполняют задачи в СИИО. Например, осведомленность может определять наличие сведений или знаний о том, с кем можно взаимодействовать, что делают другие пользователи (или что они делали в прошлом), в какой области общего рабочего пространства они работают, как произошло определенное действие, когда были обновлены данные и т.д. Выполнение процессов в СИИО носит итеративный характер.

Для пользователей в СИИО общая задача описывается возникающей проблемной ситуацией, решение которой приводит к желаемому результату или достижению цели [59]. При решении общая задача раскладывается на составные задачи, каждая из которых следует конкретному набору действий или процессов, выполняемых в некоторой последовательности (ветвление, линейная последовательность) в течение периода времени. Составная задача может быть ориентирована на конкретного пользователя: в зависимости от его предпочтений и персональной информации строится последовательность для ее решения. Конкретное действие, необходимое для решения составной задачи, которое невозможно разложить на составляющие, определяется как простая задача.

При решение общей задачи происходит взаимодействие пользователей, основанное на выполнении процессов обмена информацией. Такие процессы носят асинхронный или синхронный характер. Например, для решения творческих задач, возникающих в результате организации «мозгового штурма», взаимо-

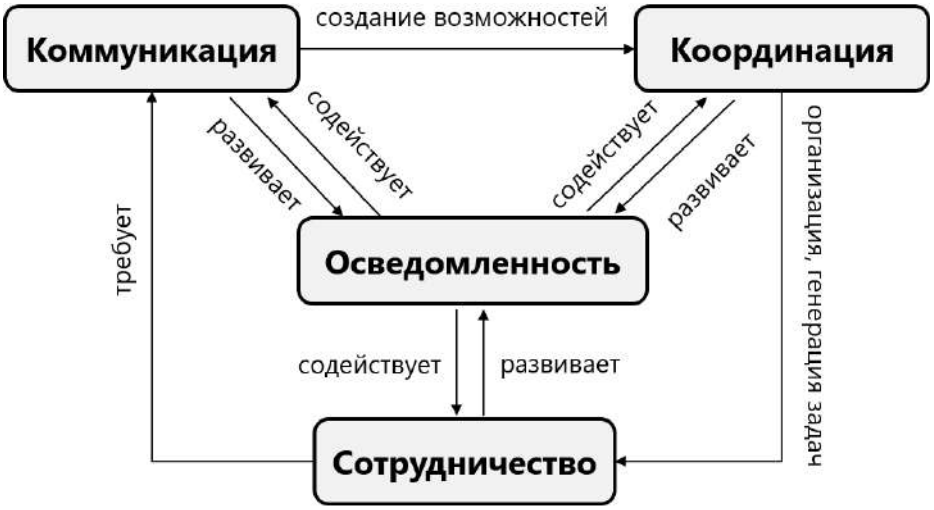


Рисунок 1.3 — Отношения основных процессов в СИИО

действие между пользователями является синхронным, т.к. творческий вклад каждого члена группы необходим для обновления стратегии решения задачи в реальном времени [99]. С другой стороны, при выполнении совместной деятельности, которая имеет заранее сформулированную стратегию решения задачи и часто не требует физического присутствия всех членов группы (например, организация тематического форума), взаимодействие может происходить асинхронно, т.е. в течение длительного периода времени. Другой важной особенностью СИИО выступает возможность пользователей обмениваться информацией с доступными ресурсами, т.е. использовать их для решения возникающих задач.

Ресурс является средством, обеспечивающим решение задач, возникающих при совместной деятельности человека или группы людей в СИИО. В соответствии с рисунком 1.4 для предоставления цифровых сервисов в СИИО интегрируются информационные, технические и экспертные ресурсы. Для интерактивной мультимедийной информационной поддержки и видеоконференцсвязи по обмену экспертными знаниями используются широкоформатные экраны, аудиосистемы помещения, персональные мобильные устройства пользователей. Множество распределенных ресурсов является динамическим и неоднородным.

Информационным ресурсом является любой вид активного кода или программного обеспечения, а также любой элемент данных или цифровое представление объекта или процесса, которые используются в информационных системах. Такие ресурсы важны с точки зрения удовлетворения информационных потребностей людей при решении задач определенной предметной области. Техническим ресурсом является любая система, машина, прибор, механизм, устройство и дру-

гие виды оборудования, которые используются при решении возникающих у людей задач в определенной предметной области.

Рисунок 1.4 — Типы интегрируемых ресурсов в СИИО

Взаимодействие между пользователями происходит в общем рабочем пространстве, где пользователи могут использовать ресурсы, связанные с их деятельностью. Такое пространство обеспечивает свойство осведомленности, отражая текущее состояние окружающего контекста и виртуализируя образы пользователей, ресурсов, происходящих процессов с помощью некоторого централизованного информационного хранилища [110]. Оно реализует механизм для манипулирования разнородной информацией (текст, электронные документы, медиафайлы), которая представляется на основе специализированной модели данных. Реализуются следующие важные возможности.

1. Привлечение, интеграция, общий доступ к ресурсам с учетом множества пользователей.
2. Контекстная осведомленность пользователей.
3. Обратная связь: реагирование СИИО на действия конкретного пользователя.
4. Реакции (отклик): реагирование СИИО на действия других пользователей и предоставление персонифицированной информации конкретному пользователю на основе выполненных действий.

5. Сохранение истории во времени, доступ к уже произошедшим событиям.

Для пользователей предоставляются цифровые сервисы, выполняющие информационную поддержку (обмен и анализ) за счет использования доступных в СИИО ресурсов, в том числе и самих пользователей. В СИИО такая информационная поддержка пользователей усиливается за счет наделения цифровых сервисов свойством контекстно-зависимости, т.е. сервисы приобретают способность учитывать состояние пользователей и имеющихся ресурсов. Сервисы приобретают также свойства «интеллектуальности»: распознавание текущей ситуации, анализ поведения человека, предоставление информации в ненавязчивой форме [40]. СИИО начинает действовать так, как будто его восприятие контекста дублирует восприятие пользователя, а выполняемые СИИО действия при этом происходят незаметно (неотличимы от выполняемых действий информационного окружения и вычислительной среды) для пользователей [53]. Необходима поддержка следующих дополнительных возможностей: учет персональных предпочтений (без необходимости явно вводить их), применение человеко-ориентированного многомодального пользовательского интерфейса, упреждающая доставка сервисов и адаптивность.

Использование современных информационных, компьютерных и телекоммуникационных технологий для совместной деятельности оказывает влияние на процесс информатизации общества и развитие связанных с этим следующих ключевых сфер жизни общества: (1) цифровая экономика, (2) наука и образование, (3) культура.

Формирование цифровой экономики определено в качестве национального интереса и приоритета Стратегии развития информационного общества в Российской Федерации на 2017 – 2030 годы (утверждена Указом Президента Российской Федерации от 09.05.2017 г. № 203). Повсеместное применение информационных и коммуникационных технологий для организации совместной деятельности людей как внутри компаний, так и между ними способствует развитию экосистемы цифровой экономики на основе использования технологий сбора и анализа данных, обмена данными, управления производственных процессов.

В результате, создается программное обеспечение, которое, как правило, предоставляет цифровые сервисы для использования компаниями в СИИО. Применение облачных технологий, а также технологий и устройств Интернета вещей является технологической основой разработки такого рода программного обеспечения. Использование СИИО в компаниях способно решить следующие основные

задачи развития социально-экономической сферы (по аналогии с некоторыми сценариями Юханссона [74]): (1) создание условий для развития электронного взаимодействия участников экономической деятельности; (2) обеспечение работников условиями для дистанционной занятости; (3) повышение производительности и эффективности труда [30].

В отношении к организации СИИО внутри крупных компаниях, как правило, выделяют две области, которые вызывают наибольшие проблемы: (1) неэффективность и ограниченность внутриорганизационных коммуникаций и взаимодействий как внутри отдельных команд, так и между ними, а также (2) неэффективность управления сопровождающей эти действия информацией. В первом случае, неэффективность определяется потерями результатов работ (особенно между вертикальными каналами связи внутри иерархической организации), вызванными проблемами координации, взаимодействия и непреднамеренной фильтрацией информации, недостаточностью корпоративной памяти, переполнением информации и ее неполным использованием, а также анализом информации. Во втором случае недопустимые модели и стратегии управления информацией могут препятствовать своевременному доступу к предоставленной информации другим членам команды.

Эти проблемы указывают на экономическую и технологическую целесообразность использования, продвижения и поддержки ИКТ в организации совместно используемых окружений, что обеспечивает создание более гибких и эффективных компаний [31]. Во-первых, сглаживание иерархии внутри компании сокращает маршруты передачи информации, что улучшает поток информации и ускоряет процесс принятия решений. Во-вторых, потенциал инноваций и технологий в области организации СИИО предлагает дополнительные синергетические эффекты в отношении множества различных компетенций членов команд. В таких условиях команды носят временный и целеориентированный характер: организуются для решения конкретных проблем и расформировываются после завершения работы. Основными видами ПО совместной деятельности выступают: системы мгновенных сообщений, системы электронной почты, системы управления содержимым, системы проведения электронных совещаний, а также системы управления и контроля производственных процессов.

С ростом зарубежного и отечественного ПО для решения задач информационного взаимодействия в СИИО возникает задача их объединения и взаимодополнение в организационно-технологических процессах предметной области

таким образом, чтобы обеспечить синергетический эффект их взаимодействия и предоставить им информацию о происходящих производственных процессах с помощью технологий Интернета вещей, используемых внутри компаний.

Сфера науки и образования представляет еще одну сферу деятельности, где использование ИКТ для организации СИИО играет важную роль в развитие процессов информатизации общества. Более того, темпы роста и развития цифровой экономики определяют компетенции и производительность трудовых ресурсов, а также научные инновации [14]. В соответствии со Стратегией развития информационного общества в РФ развитие науки и образования является одним из основных направлений для формирования информационного пространства знаний, которое создает условия для удовлетворения потребностей общества в постоянном развитии, получении качественных сведений и новых компетенций.

Применение ИКТ в СИИО предоставляет обучающимся и преподавателям новые возможности в организации информационно-методического обеспечения образовательного процесса. Например, системы электронной почты и мгновенных сообщений используются как для поддержания учебных взаимодействий между преподавателем и обучающимся, так и между самими обучающимися. Участникам образовательного процесса также предоставляется различное ПО для организации интерактивных форм обучения на основе СИИО, среди которых асинхронные и синхронные компьютерные конференции, электронные лекции и семинары, дистанционное обучение.

Применение ИКТ для организации СИИО в научно-исследовательской деятельности способствует инновационному развитию национальной цифровой экономики, предоставляя различные инструменты для организации эффективной командной работы научных кадров, направленной на решение конкретных проблем [11]. Использование СИИО позволяет сопровождать и развивать как социально, так и технически процесс публичного представления результатов и популяризации науки, активно вовлекая всех участников процесса. При этом, активно используются системы асинхронных и синхронных компьютерных конференций, электронных семинаров, систем управления содержимым [4]. В качестве основного подхода эффективного использования ПО для организации СИИО выделяют подход интеграции, при котором нужно стремится к взаимодополнению различных технологий, синергетическому эффекту их взаимодействия [3].

Программные системы для организации СИИО содержат в себе огромный потенциал для сохранения и развития национальной культуры. Деятельность

людей внутри СИИО приобретает черты социально-культурной деятельности, где взаимодействие людей направлено на создание, обогащение, сохранение и распространение общественно значимых объектов культурного наследия посредством процессов информационного обмена. В Стратегии развития информационного общества в РФ подчеркивается важность программных систем для установления таких культурных связей с проживающими за рубежом соотечественниками, иностранными гражданами и лицами без гражданства, являющимися носителями русского язык, что оказывает непосредственное влияние на формирование информационного пространства знаний. В тоже время учреждения культуры, в которых организуются СИИО, являются основными местами возникновения совместной социально-культурной деятельности.

Процесс взаимодействия с посетителем в современных учреждениях культуры до сих пор имеет, в основном, однонаправленный характер. Например, в музеях «сообщение» посетителю передается с помощью различных экспозиционных средств в процессе осмотра экспозиции, а традиционными формами обратной связи являются анкеты, в которых посетителям предлагается дать оценку экспозиции и работе музея в целом, а также книги отзывов и предложений. Переход учреждений культуры на цифровое представление информации об объектах культурного наследия и использование цифровых сервисов расширяет приложения СИИО. В настоящее время технологии и оборудование Интернета вещей меняют традиционный способ активности посетителей в учреждениях культуры [38]. Экспонаты преобразуются в объекты Интернета вещей, предоставляющие информацию о себе и непосредственно взаимодействующие с посетителями и другими объектами, выполняя тем самым функции пополнения и распространения объектов культурного наследия [1].

С помощью организации СИИО в учреждениях культуры можно расширить однонаправленный процесс передачи информации между посетителями, организаторами и другими участниками с помощью механизмом обратной связи для коллективного выполнения функций социально-культурной деятельности над объектами культурного наследия. Учреждения культуры будут предлагать некую вычислительную среду, которая будет включать в себя инструментальные средства и ПО, в основном направленные на асинхронное взаимодействие участников, предоставление виртуального рабочего пространства и управления содержимым посредством предоставления цифровых сервисов.

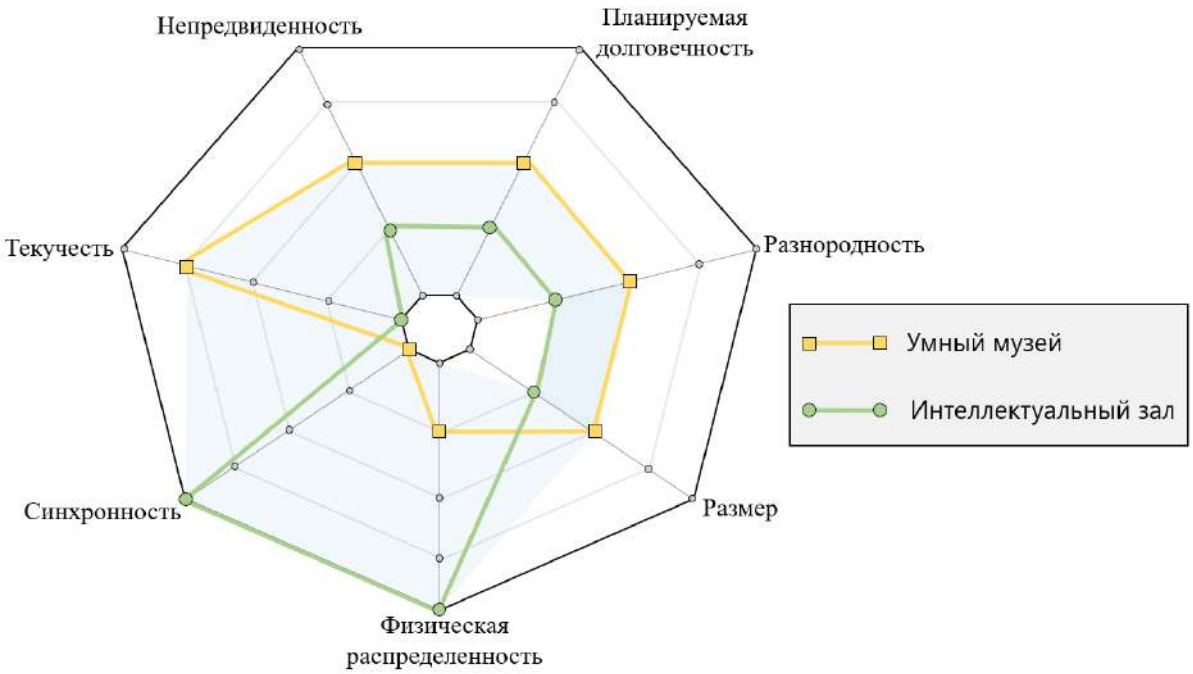


Рисунок 1.5 — Применение MoCA-модели для описания характеристик СИИО

На рисунке 1.5 представлено применение MoCA-модели [75; 83] для описания характеристик СИИО для таких предметных областей, как интеллектуальный зал и умный музей. Каждая характеристика представляется с помощью собственной шкалы, значение которой увеличивается от центра к краям модели. Размерность каждой шкалы для свойства описывается следующим образом:

1. *Синхронность*. Взаимодействие пользователей может проводится в одно и тоже время, либо в разное время.
2. *Физическая распределенность*. С одной стороны шкалы, информационное взаимодействие проходит в одной и той же географической области. С другой стороны, информационное взаимодействие может происходить в различных географических точках.
3. *Размер*. Количество пользователей в СИИО.
4. *Разнородность*. Характеризуется количеством разнородных групп среди пользователей. С одной стороны шкалы, небольшие, однородные группы пользователей с аналогичными опытом и образованием. С другой стороны, большое количество разнородных групп среди пользователей СИИО.
5. *Непредвиденность*. Характеризует степень новизны, возникновения, зарождения информационного взаимодействия. С одной стороны шкалы, стабильные процессы информационного взаимодействия с ожидаемыми непредвиденными обстоятельствами — простые варианты организации СИИО. С другой стороны, информационное взаимодействие с частыми

изменениями и типичными условиями организации, без явной цели и установленных шагов проведения — сложные нестабильные и нестандартные варианты организации СИИО. Самые эффективные варианты СИИО, часто располагаются на середине этой шкалы.

6. *Планируемая долговечность*. Длительность выполнения скоординированных действий. С одной стороны, краткосрочное планируемое устойчивое состояние, с другой стороны – долгосрочное планируемое устойчивое состояние. Более длительная планируемая устойчивость требует больших накладных расходов на создание организационных и координационных методов и инструментов, специализированных терминов и языка. Более краткосрочная планируемая устойчивость наоборот страдает недостаточностью использования общих методов, инструментов, терминов и языка.
7. *Текучесть*. Характеризует оборот, сменяемость, текучесть пользователей. Определяет стабильность состава пользователей СИИО. С одной стороны находится закрытые, частные СИИО, в которой состав пользователей меняется медленно (например, конференция). С другой стороны, полностью открытая совместная деятельность, в котором могут участвовать «массы» (например, веб-сайты вики).

В качестве представителей СИИО в диссертационной работе рассматриваются окружения умного музея [43], интеллектуального зала [23] и промышленного предприятия [53]. На них обосновывается актуальность темы исследования, а также демонстрируются возможности практического применения полученных результатов. Окружение умного музея реализуется на базе Музея истории ПетрГУ, материалы которого относятся к истории повседневной жизни преподавателей, исследователей и студентов (например, питание, обстановка в комнате общежития, одежда). Окружение умного музея обеспечивает построение сервисов для коллективного семантического аннотирования, связывания информации и персонализированного доступа к корпусу источников по истории университета. Окружение интеллектуального зала предназначено для обеспечения информационной поддержки людей в организации таких мероприятий совместной деятельности, как конференция и семинары.

Для рассматриваемых случаев окружений процесс создания СИИО с описанными свойствами требует решения следующих проблем: а) нет достаточного (для имеющегося разнообразия ресурсов) метода разработки ПО с интеграцией

динамичных и неоднородных ресурсов в СИИО; б) нет достаточных (для имеющегося разнообразия предметных областей) моделей и шаблонов проектирования контекстных сервисов, где сервис строится как распределенная динамическая неоднородная система; в) нет развитых (с существенной долей кодогенерации) средств программирования сервисов. Для решения данных проблем в диссертационной работе предлагается метод автоматизированной разработки программной инфраструктуры, обеспечивающий семантическую интероперабельность агентов при построении ими сервисов на основе интеграции ресурсов в СИИО.

1.2 Обзор исследований в области организации совместно используемых информационных интернет-окружений

Исследованием вопросов создания и организации СИИО активно велись и ведутся как в России, так и за рубежом. Одна из основных областей исследования посвящена изучению явления совместной деятельности людей с использованием ИКТ. Совместная деятельность является неотъемлемой частью многих современных производственных и социальных процессов, которая определяет вид групповых действий, направленных на получение общего результата [14; 31]. Такая деятельность вовлекает в себя множество участников (сотрудники, клиенты, партнеры, поставщики, заказчики, эксперты) с общими целями, взаимодействующих вместе (коммуникация, обсуждение, координирование, одобрение) путем совместного использования и разделения информации (электронные документы, пункты плана мероприятия, расписания, потоки работ), идей и происходящих процессов (торговля, маркетинг, обслуживание клиентов, инженерия, исследование и разработка, бухгалтерия).

В 1991 году Роберт Юханссон (Robert Johansen) представил доклад «Teams for tomorrow», в котором представил свое видение изменения явления совместной деятельности под влиянием развития ИКТ в ближайшем будущем на основе шести сценариев [74]. Сценарии определили интеграционные тенденции развития во временном (сценарий «Своевременное обучение»), пространственном (сценарий «Комнаты виртуальной реальности»), организационном (сценарий «Координируемый рабочий процесс») и культурном отношении (сценарий «Межкультурное взаимодействие») на основе решения задач распределенных вычислений на бо-

лее низком уровне (сценарий «В любое время/в любом месте» и сценарий «Окно куда угодно»). По большей части, предположения, сделанные Юханссоном, оказались верными, не считая использования технологий виртуальной реальности (основная область применения на данный момент – индустрия развлечений) и поддержки межкультурного взаимодействия среди мультикультурных групп в процессе совместной деятельности [12].

Общая область исследований, к которой относятся эти сценарии, называется «совместная деятельность с использованием ИКТ» (от англ., «computer-supported cooperative work», CSCW). Айрин Грейф (Irene Greif) и Пол Кэшман (Paul Cashman) впервые ввели термин CSCW на семинаре в 1984 году для представления роли ИКТ в поддержке различных видов деятельности людей [66]. В русской литературе нет устоявшегося унифицированного термина, характеризующего данную область исследования. Основные результаты в этой области с явным выделением соответствующего термина в большинстве случаев встречаются в работах, посвященных исследованиям организации образовательных процессов, проблемам психического развития, развитию социального программного обеспечения [2; 33; 37]. Используются термины: «компьютерно-опосредствованная совместная деятельность», «совместная деятельность на базе компьютеров», «совместная деятельность с использованием ИКТ», «компьютерно-обеспеченная совместная работа».

Термин «совместная деятельность с использованием ИКТ» определяет междисциплинарную область, ориентированную на изучение характеристик и особенностей совместной деятельности людей с целью разработки интеллектуальных информационных и коммуникационных технологий для поддержки такой деятельности. Можно выделить два основных направления развития данной области исследования. В первом направлении многие исследователи рассматривают термин со стороны технологий, вычислительной аппаратуры и программного обеспечения совместной деятельности (от англ., «groupware»). Во втором направлении некоторые исследователи рассматривают этот термин для представления идеи о сотрудничестве среди групп людей, использующих компьютеры [56; 76], изучая инструменты и методы программного обеспечения совместной деятельности, а также психологические, социальные и организационные аспекты данной идеи. В диссертационном исследовании уделяется внимание первому направлению, с акцентом на использование распределенных вычислений для создания

программной инфраструктуры СИИО, рассматривая проблему с точки зрения информатики и программной инженерии.

Говоря об отдельном программном обеспечении (программных системах) в области организации окружений совместной деятельности, его можно отнести к одному из классов в зависимости от выбранного способа классификации. На основе выполненного обзора исследований можно выделить три способа классификации на основе: 1) матрицы «время-место», 2) базовых процессов совместной деятельности (коммуникация, координация, сотрудничество), 3) области применения программных систем. Стоит отметить, что матрица «время-место», построенная по принципу дихотомии и использующая два измерения, характеризующиеся с помощью оси «время», которая разбивается на синхронный и асинхронный тип деятельности, и оси «место», которая разбивается на удаленный и «в рамках одного помещения» тип деятельности, является весьма ограниченной, и, на самом деле, изначально использовалась для определения концепции только электронных совещаний [99]. Отдельная программная система для организации СИИО, как правило, обладает широким спектром функций, которые обеспечивают поддержку базовых процессов совместной деятельности, в том числе и свойства обеспечения осведомленности. Однако, в зависимости от интенсивности происходящей совместной деятельности и использования конкретной системы прослеживаются отличия между процессами, что позволяет выявить характер преобладающих ключевых функций системы.

Повсеместный доступ к сети Интернет, а также рост пропускной способности сети и вычислительных мощностей способствовал в последние годы созданию множества программных систем для решения отдельных задач в организации СИИО. В соответствии с таблицей 1 приведены основные распространенные классы программных систем для организации СИИО, используя в качестве основных критериев область применения и преобладающий процесс совместной деятельности, а также будут приведены примеры реальных систем.

Для интеграции людей, происходящих процессов, технологий, программных систем, связанных с совместной деятельностью людей, создаются специализированные информационные окружения (от англ., «collaborative work environment», CWE) [10; 39]. Они улучшают межфункциональное и виртуальное взаимодействие пользователей, способствуют обучению и принятию высококачественных решений. Основными концепциями для создания таких окружений являются человеко-машинное взаимодействие и повсеместные вычисления [112].

Таблица 1 — Классификация программных систем для организации СИИО по областям применения

Категория	Описание	Примеры
Системы для поддержки коммуникации		
Системы обмена сообщениями: (1) системы электронной почты, (2) системы мгновенных сообщений	(1) Передача и получение электронных сообщений, синхронное взаимодействие пользователей. (2) Предназначены для обмена сообщениями в реальном времени, используется сеть «Интернет». Позволяют передавать текстовые сообщения, звуковые сигналы, видео, изображения. Могут использоваться для организации групповых текстовых чатов и видеочатов.	(1) Gmail, Яндекс.Почта; (2) Telegram, Skype, WhatsApp
Системы обеспечения видео- и телеконференцсвязи	Предназначены для интерактивного взаимодействия в реальном времени нескольких (трех и более) удаленных пользователей, также обеспечивается обмен аудио- и видеоинформацией.	Adobe Connect, GoToMeeting, WebEx
Системы потоковой трансляции аудио и видео	Предоставляет пользователям потоковую трансляция различной мультимедиа информации в режиме реального времени, использует технологию потокового вещания.	Twitch, YouTube, Periscope
Системы для поддержки координации		
Системы управления и контроля производственных процессов	Повышение эффективности производства на основе автоматизации планирования и координации различных процессов: обеспечения материалами, работа склада и др.	Opti-Wood, Master SCADA, Wonderware Software
Системы группового планирования и составления расписаний	Автоматизация задач календарного планирования и составления расписаний, распределение событий во временном периоде в условиях различных ресурсных ограничений.	Planday, Scoro, Teamup Calendar
Системы для поддержки сотрудничества		
Системы виртуального рабочего пространства: (1) многопользовательские редакторы, (2) системы совместного использования приложений, (4) системы управления версиями, (3) многопользовательские интерактивные доски, (4) вики-системы	Позволяют создавать «общее пространство» для возможности работы с ним «в любом месте», «в любой время» и «с любого устройства». (1) Совместное редактирование (чаще через веб-приложение) в реальном времени электронных документов или данных. (2) Управление изменениями (отслеживание, отмена и т.д.) в различных наборах информации (чаще исходный код). (3) Позволяют отображать, изменять, обрабатывать изображения, видео, текст в цифровой двухмерной области. (4) Совместное изменение содержимого веб-сайта с помощью вики-инструментов.	(1) Gobby, Google Docs; (2) Subversion, Git; (3) Twiddla, Groupboard, Scribblar; (4) Wikipedia, TikiWiki

Продолжение таблицы 1

Категория	Описание	Примеры
Системы проведения совещаний и коллективного принятия решений	Управление совещаниями с возможностями автоматизации регистрации участников, процессов ведения протоколов, установление обсуждаемых вопросов, контроль принимаемых решений.	Citeck ECOS Community, MeetM, Loomio
Системы проведения конференций: (1) системы проведения асинхронных конференций, (2) системы проведения синхронных конференций	Повышение качества информационного сопровождения различных видов конференций (научные, корпоративные), автоматизация рутинных действий участников и организаторов. (1) Чаще организуются в виде тематического форума, удаленное управление дискуссией (обмен информацией в течение длительного периода времени), структурирование форума. (2) Интерактивное взаимодействие участников в режиме реального времени в рамках секции с докладами, сопровождаются системами обеспечения видео- и телеконференцсвязи.	(1) Moodle, WebCT, Blackboard; (2) ConfTool, Extron

Популярным классом СИИО являются умные комнаты – технологически оснащенные среды, специально предназначенные для поддержки совместной (локальной или удаленной) деятельности людей и обеспечивающие адекватное решение возникающих при этом проблем [53]. Понятие умной комнаты может включать в себя такие понятия как умные телефоны, умная бытовая техника, умные дома, умные здания и даже умные города. Такой вид интеллектуальности связан с понятием повсеместных и проникающих вычислений, поскольку оно выявляет и определяет понятие передачи и обработки объектов, встроенных в наш повседневный мир. Так, «умный» в работе [53] определяется как «способный помочь человеческой деятельности посредством бесшовного использования повсеместных вычислений». Вычислительная система не должна быть способной формировать новые концепции о мире — ей нужно только действовать так, как будто ее восприятие дублирует восприятие пользователя. Чтобы достичь иллюзии мышления и интеллекта система должна осознавать контекст. Контекст может включать: местоположение пользователя, различная информация о среде, состояние ситуаций, окружение, задачи и т.п.

СИИО в основном состоит из готовых компьютерных технологий и средств для организации информационного обмена между пользователями, как электронная почта, обмен мгновенными сообщениями, чаты, доски обсуждений, общие доски, а также мобильная связь, мультимедийное пространство и видеоконференции. Одной из основных проблем, связанных с разработкой СИИО, является

отсутствие стандартов и общих определений для обеспечения обмена информацией и взаимодействия между различными видами используемых ресурсов, обеспечение интероперабельности между ними. СИИО определяет совокупность интегрированных и связанных ресурсов, обеспечивающих общий доступ к содержимому и позволяющих распределенным субъектам беспрепятственно работать вместе в направлении достижения общих целей [50]. Выделяются основные сущности для организации процессов совместной деятельности в СИИО [41]:

1. *Действующее лицо*. Пользователь, программа или сущность с определенными приобретенными возможностями, которая может играть роль во время выполнения, использует устройства и производит некоторые действия.
2. *Действующая группа*. Представляет действующее лицо или группу действующих лиц, особенностью которых является достижение одной или нескольких целей.
3. *Роль*. Является обозначением набора связанных задач. Действующее лицо определенной роли может участвовать в индивидуальных или совместных задачах, а также может быть ответственным для достижения цели.
4. *Цель*. Описывает определенное состояние мира, которое хотело бы достичь действующее лицо.
5. *Мягкая цель*. Это условие в мире, которое хотело бы достичь действующее лицо, но, в отличии от понятия (жесткой) цели, условия ее достижения точно не определены. Мягкая цель обычно является атрибутом качества.
6. *Задача*. Определяет конкретный способ сделать что-то. Абстрактная задача определяет набор конкретных задач. Конкретные задачи: (1) индивидуальная задача выполняется действующим лицом без какого-либо взаимодействия; (2) совместные задачи требуют участия двух или более субъектов.
7. *Ресурс*. Является сущностью (информационной, технической или экспертной), которая необходима действующему лицу, чтобы достичь цели или выполнить задачу.
8. *Ресурс осведомленности*. Этот элемент представляет собой некоторое осознание осведомленности того, что необходимо роли, чтобы выполнить задачу. Эти элементы специализируются в соответствии со своей

временной категорией и классифицируются в соответствии с их важностью (хорошо иметь, желательно, очень желательно и обязательно). Предоставляет информацию и знания о текущем состоянии общего информационного рабочего пространства. Определяет знания о состоянии общего рабочего пространства.

Разработка СИИО все более ориентируется на повсеместные и мобильные вычисления, чтобы обеспечить возможность взаимодействовать пользователям за пределами физического рабочего пространства, создавая виртуальные рабочие пространства, которые могут быть доступны в любое время, в любом месте и решать любые поддерживаемые системой задачи [78]. Проблема организации рационального взаимодействия пользователей решается также в социо-кибер-физических системах, где преобладают такие свойства как массовость и неоднородность участников. Участниками таких систем выступают сами люди, окружающие физические объекты, цифровые сервисы, внешние информационные ресурсы и сервисы сети Интернет. Основной задачей в социо-кибер-физических системах является задача моделирования информационно-управляемого взаимодействия, которое организуется в условиях Интернета вещей и больших данных, обеспечивая кибер-физическую или цифровую конвергенцию между неоднородными участниками [43; 86; 121].

Анализ программных систем для организации СИИО показывает, что различные функции предлагаются как отдельные приложения, либо как отдельные модули, либо конфигурируемый набор модулей. Также отдельные модули и их наборы могут быть рассмотрены как сервисы. В [59; 125] предлагается решение проблемы интеграции различных независимых программных систем для организации СИИО. Проблема решается за счет предоставления конечным пользователям цифровых сервисов, которые интегрируют ресурсы в информационное окружение с использованием промежуточного ПО. Для обеспечения интероперабельности используется сервис-ориентированная архитектура (SOA). В архитектуре SOA набор сервисов обмениваются данными друг с другом, в которых сообщения включают в себя простую передачу данных или координирующие действия. В настоящее время методы разработки архитектуры SOA основаны на использовании стандартов веб-сервисов.

Одним их первых протоколов для архитектуры SOA являлся протокол DCOM (расширение стандарта COM). Протокол имел существенный недостаток: операции, выполняемые в сервисах, включались в транспортный протокол, поэто-

му его нельзя было изменить из-за необходимости редактирования кода сервисов. Протокол вызова удаленных процедур XML-RPC являлся первым решением на основе XML для осуществления обмена данными на основе протокола HTTP. Протокол не поддерживает возможность создавать динамические сервисы, т.е., с использованием механизма поиска доступных сервисов. В настоящее время стандарты и протоколы для веб-сервисов позволяют решать эти проблемы [48]. Одной из наиболее привлекательных особенностей является использование языка XML, обеспечивающего независимость протокола и языка [102].

Популярным языком описания веб-сервисов является язык WSDL, основанный на языке XML [60; 102]. В качестве основного протокола обмена WSDL-описаниями в распределенной вычислительной среде выступает протокол SOAP. Однако, большинство программных систем для организации СИИО не определяют свой API с помощью веб-сервисов. Для решения данной проблемы разрабатываются специализированные программные посредники с инкапсуляцией их функциональных возможностей. В диссертационном исследовании такие системы рассматриваются как внешние информационные ресурсы, которые интегрируются в СИИО посредством автономных программных компонент, взаимодействующих с внешней программной системой на основе API.

Поверхностная интеграция ресурсов может быть достигнута с использованием предоставления сервисов [71]. Однако утверждается, что такие системы обеспечивают интеграцию и интероперабельность только на синтаксическом уровне. Семантические детали, касающиеся реального понимания того, что делает система и какая информацию у ней есть, не поддерживаются.

Модель агента EDA (Epistemic, Deontic, Axiological) используется для разработки метода семиотики в многоагентной архитектуре [100]. Архитектура включает в себя уровень данных, базу данных, репозиторий правил/норм и агентов. Рассматриваются четыре типа агентов:

1. *локальный агент*: представляет человека в организации;
2. *супер-агент*: управляющий агент, используется для администрирования действий локальных агентов и для их достижения интеллектуальности, использует хранилище правил;
3. *интерактивный агент*: обеспечивает взаимодействие локальных агентов, т.е. все запросы на взаимодействие идут через него, перед этим локальные агенты должны быть зарегистрированы в нем;

4. *агент сотрудничества*: обеспечивает взаимодействие нескольких систем, если локальные агенты не могут выполнить какое-либо действие, интерактивный агент обращается к агенту сотрудничества с целью поиска агентов в других организациях/системах.

В [61] предлагается разрабатывать СИИО в виде многоагентной системы, представляя каждый сервис как отдельного агента, который работает в предметной области СИИО и предметной области многоагентной системы. Все агенты представляют сервисы, каждый из которых реализует следующие функции: коммуникация; координация; сотрудничество (производство, генерация артефактов совместной деятельности). Каждый агент проектируется с использованием многоагентной модели PAC: Presentation (входные, выходные параметры), Abstraction (описание основных функций), Control (зависимости, взаимодействие с другими агентами).

Учет семантики происходящих процессов в СИИО позволяет определять и контролировать поток информации между различными программными системами, а также адаптировать информационное окружение в соответствии с потребностями пользователей. В работе [117] представлен семантико-ориентированный подход для разработки СИИО, который обеспечивает интеграцию таких классов программных систем совместной деятельности, как системы управления содержимым и системы мгновенных сообщений. Предлагаются онтологии и набор правил для уведомления пользователей с помощью сообщений об изменениях документов в системе управления содержимым.

Если система должна включать в себя предпочтения пользователей, ей необходим механизм поддержки человеко-машинного взаимодействия. Для этого необходимы семантические аннотации сервисов. Для семантических сервисов используются следующие языки/стандарты: WSMO, OWL-S, WSDL-S [116; 118]. Использование семантических сервисов позволяет разрабатывать программную инфраструктуру для СИИО, которая способна обеспечить поддержку обмена информацией между различными информационными и техническими ресурсами.

В [49; 67; 73] рассматриваются программные системы для организации СИИО, которые предоставляют среду для совместной работы — общее рабочее пространство. В этом ограниченном пространстве люди могут видеть и манипулировать объектами, связанными с их деятельностью. В реальном мире общее рабочее пространство — это физическое пространство, где люди могут выполнять какую-либо совместную деятельность.

1.3 Требования к разработке программной инфраструктуры для организации совместно используемого информационного интернет-окружения

Основной задачей при разработке СИИО является создание интероперабельной программной инфраструктуры, отвечающей за построение и выполнение контекстных сервисов с использованием интегрированных ресурсов и инструментальных средств программирования сервисов.

В [59] определяются требования к разработке информационных окружений для решения задач совместной деятельности. Требования определяют тенденцию следования принципам архитектуры SOA и стандартам семантических веб-сервисов, чтобы обеспечить совместимые, взаимодействующие решения, которые слабо взаимосвязаны, расширяемы и гибки. Также выделяются требования для достижения решения общих задач и организации взаимодействия: масштабируемость, интероперабельность; удобство и простота использования; «в любом месте/в любое время»; архитектура SOA; проактивность; обнаружение необходимой информации; целеориентированность.

СИИО должно предлагать инфраструктуру для предоставления повсеместных, ориентированных на совместную деятельность сервисов. Новые тенденции (в основном, в бизнес-секторе) сосредоточены на поиске окружения, которое позволит сотрудничать и разделять информацию между различными программными системами и информационными окружениями [20]. В [55; 78] представлены функциональные требования к информационным окружениям совместной деятельности. Выделяются требования, направленные на обеспечение поддержки основных процессов совместной деятельности (коммуникация, координация, сотрудничество, осведомленность), а также возможности удаленной работы пользователей. Важным требованием является поддержка механизма обратной связи. Реакция на возникающие события должна быть ориентирована на конкретного пользователя в зависимости от его роли или прав.

В основе процессов визуализации информации для пользователей в СИИО лежит концепция WYSIWIS (от англ., «what you see is what I see») [49; 54], описывающая видение пользовательского интерфейса при решении задач совместной деятельности. Пользователям должны предоставляться визуальные интерфейсы (например, на мобильных устройствах) с индивидуальным оформлением, но с идентичным содержимым, или разные объекты общих данных (например, разные

главы одного и тоже документа в общем редакторе). Информационная безопасность является также одним из важных требований [46]. СИИО должно содержать способы разделения общих данных от частных. Уровень представления данных не должен ограничиваться уровнем файла, должны использоваться более абстрактные объекты данных.

СИИО должно интегрировать различные программные системы совместной деятельности и переключаться между ними в зависимости от задач пользователей. Пользователи должны иметь возможность переключаться с одной программной системы на другую без прерывания аудио- или видео-соединений. Пользовательский интерфейс должен быть модульным, обеспечивать переключение между синхронным и асинхронным способом взаимодействия пользователей, предоставлять возможность создавать индивидуальные представления общих данных.

Программная инфраструктура для СИИО должна быть адаптивной (гибкой) — поддержка различных аппаратных платформ, операционных систем и графических интерфейсов. Для обмена информацией должны поддерживаться различные форматы медиа-информации, а также сетевые протоколы обмена данными. Пользователи в СИИО могут являться удаленными, но им предоставляются схожие сервисы.

Поддержка общей осведомленности пользователей является необходимым условием при совместной работе. Осведомленность о действиях других пользователей в общем рабочем пространстве предоставляет пользователям контекст для выполнения задач, помогает повысить эффективность взаимодействия, помогает определять возможности для оказания помощи другим пользователям [97]. Для этого общее рабочее пространство должно содержать информацию об активности пользователей. Такая информация может быть собрана разными способами: используя окружающие пользователей ресурсы и объекты; анализируя действия самих пользователей.

В [125] определяется требование, касающееся управления общим доступом к ресурсам. Требование касается вопросов координации, связанных с доступом и использованием ресурсов в общем рабочем пространстве. Управление общим доступом требует решения проблем координации, связанных с доступом и использованием ограниченных ресурсов (интерактивные доски, проекторы, общие экраны) в общем рабочем пространстве. Выделяются специализированные действия над ресурсами, где необходимо решать проблемы с общим доступом:

1. *Получение ресурсов*. Прямое немедленное использование. Некоторые ресурсы могут использовать только один пользователь одновременно.
2. *Резервация ресурсов для дальнейшего использования*. Координация этих действий схожа с получением ресурсов, но, в целом, объекты резервируются и могут располагаться в непосредственной близости к запланированному моменту использования.
3. *Сохранение результатов своей деятельности*. Когда пользователь завершил выполнение задачи в общем рабочем пространстве (создал набор артефактов/объектов или каким-то образом организовал их), необходимо обеспечить, чтобы другие не изменили или не уничтожали его работу.

На основе представленного обзора исследований в области СИИО и требований к разработке программной инфраструктуры получен сводный список требований. Требования учитывают возможность интеграции данных, обмена знаниями и интеллектуальных рассуждений в СИИО. Требуется обеспечивать интероперабельность и интеграцию неоднородных динамических ресурсов. Требования подразделяются на три категории: (1) функциональные, (2) нефункциональные и (3) требования к разработке сервисов.

Таблица 2 — Сводный список требований к разработке программной инфраструктуры

№	Требование	Описание
Функциональные требования		
	Осведомленность	СИИО должно отражать текущее состояние окружающего контекста и виртуализировать образы пользователей, ресурсов, происходящих процессов с помощью некоторого централизованного информационного содержимого.
	Общий доступ к ресурсам	Предоставление общего доступа касается решения проблем координации, связанных с доступом и использованием ограниченных ресурсов в СИИО.
	Машинно-читаемая логика	Промежуточное ПО для СИИО должно использовать стандарты W3C для кодирования семантики хранимых данных в терминах аксиом классов и свойств с использованием элементов дескрипционной логики в формате RDF/XML. Это выполняется с помощью OWL, построенного поверх RDF, RDF Schema, и XML Schema.

Продолжение таблицы 2

№	Требование	Описание
	Обнаружение знаний	Обнаружение знаний определяется как нетривиальное извлечение неявной, неизвестной и потенциально полезной информации из данных. Извлечение высокоуровневых знаний из разнородных, мультимодальных наборов данных является важным компонентом в СИИО.
	Объединение данных	Определяет возможность СИИО для интеграции разнородных данных из нескольких распределенных информационных ресурсов в согласованный общий вид. Объединение данных – ключевое требование из-за динамичности и неоднородности ресурсов в СИИО.
	Проактивность	Программная инфраструктура должна предоставлять механизмы для уведомления пользователей (обратная связь, отклики), сервисов и других автономных программных модулей.
		Нефункциональные требования
	Расширяемость	Программная инфраструктура должна обладать способностью расширяться в соответствии с потребностями информационного окружения и не терять уровень производительности.
	Доступность	СИИО требует доступности, что означает, что любой пользователь может получить доступ к ней в любое время и в любом месте. Организуется повсеместный доступ и возможность использования функциональных средств для информационного обмена
	Интероперабельность	СИИО должно обеспечивать одновременную работу множества разнородных ресурсов друг с другом и с пользователями. Семантическая интероперабельность должна обеспечивать обмен данными между другими СИИО. Для обеспечения семантической интероперабельности можно использовать онтологии.
	Ориентация на цель (общую задачу)	Сервисы программной инфраструктуры должны быть ориентированы на решение различных проблем «разумным» способом в соответствии с шаблоном разложения пути достижения цели (или общей задачи) на задачи (или подзадачи), по которому действует пользователь.
	Безопасность	Контекстная осведомленность в СИИО может раскрывать персональную информацию пользователей. Программная инфраструктура должна обеспечить требование безопасности, тем самым организуя, контролируя и ограничивая доступ к хранимой информации.

Продолжение таблицы 2

№	Требование	Описание
	Адаптивность	Адаптивность является важным требованием из-за постоянно растущего множества устройств (смартфоны, маршрутизаторы и т.д.), каждое из которых может выступать ресурсом или устройством пользователя. Программная инфраструктура должна обеспечить независимость от сетевого протокола, протокола обмена медиа-информацией, и т.д. Адаптивность для СИИО необходима, чтобы соответствовать изменениям в вычислительной среде на нижнем уровне.
	Удобство использования	Программная инфраструктура должна реализовывать и предоставлять интуитивные и многомодальные пользовательские интерфейсы. Должен соблюдаться баланс между вкладом и получаемой выгодой от использования СИИО для каждого пользователя.
	Требования к разработке сервисов	
	Абстракции программирования	Программная инфраструктура должна обеспечивать поддержку абстракций программирования, предоставляя высокоуровневые интерфейсы программирования для разработчиков сервисов.
	Семантические веб-сервисы	Семантический веб предлагает универсальные стандарты для разработки семантических веб-сервисов. Таким образом, СИИО должны предоставлять сервисы, имеющие однозначно описанную семантику, доступные в других информационных окружениях посредством сети Интернет и применимые для автоматизированного поиска, композиции, построения и выполнения.
	Автоматизированное проектирование и программирование	Должны использоваться подходы к разработке, не требующие интенсивного вовлечения разработчиков, и с широким использованием средств автоматизированного проектирования и программирования, которые обеспечивают прототипирование сервисов.

Функциональные требования покрывают функции, предоставляемые программной инфраструктурой СИИО. Эти требования связаны со стандартами и технологиями Семантического веба и их реализацией в СИИО. Нефункциональные требования описывают качества программной инфраструктуры. Нефункциональные требования представляют собой общие определяющие свойства, рассматриваемые в целом для СИИО или только для какого-либо отдельного аспекта. Функциональные требования поддерживаются этими требованиями, которые накладывают ограничения на проектирование и реализацию. Требования

к разработке сервисов должны учитываться разработчиками сервисов. С одной стороны, большое число сервисов накладывает требования, связанные с упрощением процесса разработки и сопровождения. С другой стороны, Семантический веб определяет универсальные стандарты разработки сервисов.

Приведенные выше требования представляют собой набор потребностей или условий, которые разработчики СИИО должны стремиться удовлетворить при разработке программной инфраструктуры. При выполнении процессов проектирования, разработки и сопровождения разработчик может учитывать предлагаемые требования. В результате повышается эффективность разработки программной инфраструктуры СИИО.

Требования могут изменяться со временем или быть зависимы от видения программной инфраструктуры. Например, при разработке семантико-ориентированной программной инфраструктуры этап проектирования сводится к созданию онтологических моделей, на основе которых обеспечивается интероперабельность пользователей и ресурсов.

1.4 Концепции, подходы, методы и технологии для разработки программной инфраструктуры

Интернет вещей (англ. Internet of Things, IoT)

Вопросы взаимодействия множества вычислительных устройств для решения совместных задач обработки данных и эффективной доставки информационного сервиса мобильным пользователям зачастую возникают при разработке программных систем в IoT-средах. В условиях таких сред решения, обеспечивающие взаимодействие, должны учитывать массовость и неоднородность участвующих объектов, высокий уровень их физической распределенности, большие объемы и разнообразие структур используемых данных, широкий набор возможных средств сетевых коммуникаций для передачи данных, множество влияющих на коммуникацию параметров и их высокую вариабельность [70; 88; 92].

Существуют работы по исследованию программной инфраструктуры IoT-сред и их требований [69; 87]. Программную инфраструктуру для таких сред

можно рассматривать среди разных точек зрения, одна из них – семантико-ориентированный подход. Последние работы в области IoT подтверждают рост использования семантики [106]. Объем рынка IoT-технологий согласно прогнозу Аналитической компании International Data Corporation (IDC, http://www.idc.com/) к 2020 году превысит $7 млрд, а число устройств составит 28,1 млрд. Кроме того, научное исследование затрагивает такие современные инженерные концепции распределенных вычислений как окружающий интеллект (ambient intelligent), повсеместные вычисления (ubiquitous computing), программно-конфигурируемые сети (Software-Defined Networks) и Семантический веб (Semantic Web).

Развитие технологии «Интернет вещей» привело к появлению в окружении человека вычислительных сред с множеством оцифрованных объектов физического мира, с одной стороны, и с множеством информационных и вычислительных ресурсов глобального пространства современного Интернет с другой. Для решения проблемы интеллектуализации IoT-сред возникает необходимость создания комплекса научно-технических решений, который позволит создавать на его основе сервис-ориентированные интеллектуальные информационные Интернет-системы нового поколения, в том числе и СИИО.

Сервисы направлены на обеспечение взаимодействия с информационными и техническими ресурсами, посредством которых пользователи решают свои задачи заданной предметной области. Сервисы позволяют интегрировать данные из разных источников, доступ к сервисам происходит на основе унифицированного интерфейса, с помощью которого определяется протокол взаимодействия. Поддержка автоматизированных механизмов вывода знаний и принятия решений определяют основные отличительные возможности таких сервисов.

Перечисленные тренды меняют видение на информационную поддержку и программное обеспечение. На данный момент получают развитие многоагентные сервис-ориентированные системы [79; 123], когда программные агенты выполняются на разнообразных окружающих IoT-устройствах, совместно создавая нужные в данный момент сервисы и доставляя их требуемым пользователям.

Интернет вещей позволяет подключать и интегрировать широкий спектр технологий и вычислительных устройств, от традиционных технологий бытовой автоматизации и появившихся технологий для умных городов [29], до любых видов датчиков, исполнительных устройств, приводов, RFID-меток и объектов (вещей) вместе через сеть Интернет [72; 101]. Основными строительными блоками в технологии IoT являются умные объекты, действующие автоном-

но, чтобы принимать собственные решения, воспринимать окружающую среду, взаимодействовать с другими объектами, получать доступ к существующим интернет-ресурсам и взаимодействовать с человеком.

Технология IoT будет играть решающую роль в решении многих сегодняшних проблем общества (в том числе и совместной деятельности людей). В рамках описания концептуальной модели СИИО (раздел 1.1) был использован термин «общее рабочее пространство», предполагая, что совместная деятельность не облегчается просто путем предоставления общей базы данных, а требует динамического построения пользователями и ресурсами общего информационного пространства, где обсуждается и разрешается смысл разделяемых в нем объектов [103]. Это вызывает вопрос о том, (1) как можно поддерживать такую динамическую конструкцию единого информационного пространства с помощью взаимодействия объектов с друг другом и людьми, (2) какие доступные способы анализа взаимодействия объектов, (3) какие аспекты локального контекста должны быть доступны людям, и (4) как принятые людьми решения могут становиться доступными и понятными для объектов (и наоборот).

Для развертывания программной инфраструктуры СИИО используются IoT-среды, которые встраиваются в физическое окружение человека и обеспечивают сетевую поддержку для коммуникаций и доступа к внешним Интернет-ресурсам [52; 65]. В разделе 2.1 исследуются концептуальные уровни программной инфраструктуры для организации СИИО в условиях IoT-сред.

Веб вещей (англ. Web of Things, WoT)

Следующий шаг в технологии IoT состоит в том, чтобы подключить умные IoT-объекты к вебу (глобальному пространству сети Интернет), образовав тем самым, так называемое, глобальное пространство вещей (веб вещей) [124]. Взаимодействие между этими объектами осуществляется с помощью интерфейсов прикладного программирования (API) по протоколу HTTP на основе веб-сервисов, следуя архитектуре RESTful [28]. Соответственно, сервисы и информация, предоставляемые объектами могут быть встроены в веб. Преимуществами использования веб-стандартов для IoT состоят в том, что умные объекты могут использовать тот же язык, что и другие ресурсы в Интернет. Таким об-

разом, становится достаточно легко интегрировать физические вычислительные устройства с любой веб-страницей. В результате, веб-пользователи и сервисы могут беспрепятственно исследовать физический мир и действовать в нем [63].

Семантический веб вещей (англ. Semantic Web of Things, SWoT)

Следующим шагом после WoT является создание семантического веба вещей (SWoT). Технология SWoT сосредоточена на обеспечении широкомасштабной интеграции и интероперабельности, что позволяет глобально распределять и повторно использовать умные IoT-объекты для предоставления семантических сервисов следующего поколения. Одной из проблем, связанной с переходом к технологии SWoT, является определение общих семантических описаний (онтологий), которые позволяют сделать данные универсальными и понятными. Технология SWoT обеспечивает расширение технологии IoT, позволяя интегрировать физический и цифровой миры [68; 104].

Семантический веб (англ. Semantic Web)

Разработка взаимодействующих систем и сервис-ориентированных архитектур в настоящее время идет в направлении использования стандартов семантических веб-сервисов. Основная особенность стандартов и технологий Семантического веба (например, RDF, OWL, SPARQL) — это унифицированное представление и обработка данных для обмена данными между неоднородными устройствами. Стандарты RDF и OWL используются для представления данных и их семантики в виде троек. SPARQL используется для обновления и извлечения данных из общего информационного содержимого. Семантическое представление информации в программных системах позволяет обеспечить их взаимодействие при решении людьми совместных задач [6]. Одним из перспективных направлений для создания таких интегрированных решений является использование онтологий. Существуют онтологии описывающие совместный аспект таких систем: Friend of a Friend (FOAF) [114]; Semantically Interlinked

Online Communities (SIOC) [107]; Online Presence Ontology (OPO) [113]; Standard Ontology for Ubiquitous and Pervasive Application (SOUPA) [112]. Эти онтологии полезны для представления информации, связанной с людьми, системами и онлайн-присутствием. Однако они не описывают действия пользователей в СИИО, что необходимо для обеспечения взаимодействия.

В другой стороны, концепция Веб 3.0 [81] задает развитие сервисов, определяя их на основе Семантического веба. Такие сервисы определяются как семантические, т.е. а) имеют однозначно описанную семантику, б) доступны в других информационных окружениях посредством сети Интернет, в) применимы для автоматизированного поиска, композиции, построения и доставки. Таким образом, используя концепцию Веб 3.0 в вычислительных IoT-средах, можно достигнуть их объединения посредством глобальной семантической паутины для решения совместных задач на основе семантических сервисов. К тому же Веб 3.0 служит технологической платформой для Веб 4.0, где будут использоваться технологии ИИ на основе семантического анализа для решения возникающих задач пользователя.

По мере того, как объемы информации в Интернет увеличиваются, преобладающий способ фильтрации по ключевым словам будет неэффективным [81]. Семантический поиск позволяет решить эти проблемы. Персональные агенты в сочетании с Семантическим вебом для решения задачи интерпретации метаданных рассматриваются как важное будущее направление, которое позволит интегрировать различные приложения, модули, сервисы. Большинство интерфейсов программных прикладных систем уже пытается основываться на онтологиях [78].

Интеллектуальные пространства (англ. Smart Spaces)

Подход объединяет технологии IoT, WoT и SWoT, создавая определенный класс повсеместных вычислительных сред. Фактически, эти среды являются специализированными типами вычислительных IoT-сред, которые обеспечивают слияние физического, информационного и социального миров с использованием технологий WoT и SWoT. Вычислительная среда представляет собой совокупность программных и аппаратных средств и сетевых коммуникаций для организа-

ции ИП в пространственно-ограниченной области. ИП (как среда) предоставляет доступ к различным ресурсам и обеспечивает динамическое множество участников (представленных программными агентами) контекстными сервисами тогда, когда они требуются и, по возможности, упреждающим образом [110; 111].

Ресурсы, в свою очередь, представляют собой вычислительный элемент, предоставляющий доступ к информации и/или оказывающий управляющее воздействие на физические (например, сенсоры) и цифровые объекты (например, ресурсы сети Интернет). Люди также могут выступать ресурсами, например, предоставляя экспертные знания для ИП. Каждая вычислительная среда представляется одним или несколькими семантическими информационными брокерами, они поддерживают семантическую базу знаний. Независимые программные агенты, называемые процессорами знаний, обеспечивают взаимодействие внутри ИП и синхронизацию различных участников среды.

В диссертационной работе предлагается использовать подход интеллектуальных пространств (ИП) для создания СИИО. Данный подход направлен на использование технологий Интернета вещей и Семантического веба для организации взаимодействия множества распределенных программных агентов посредством динамического построения общего информационного содержимого для совместного решения задач с помощью семантически связанных ресурсов [47]. Общее информационное содержимое поддерживает разнообразные онтолого-ориентированные, интероперабельные примитивы доступа. Они позволяют разнообразным ресурсам, которые представлены программными агентами, взаимодействовать друг с другом и с пользователями на основе обработки и анализа общих семантических данных, описанных с помощью онтологий. Построение сервисов в ИП реализуется как вычислительный процесс на основе информационно-управляемого взаимодействия агентов. В настоящее время доступны открытые программные реализации исследовательских прототипов для создания таких ИП [115].

Участниками взаимодействия выступают программные агенты, называемые процессорами знаний, которые являются потребителями и производителями общего информационного содержимого. Программные агенты запускаются на различных вычислительных устройствах среды (например, мобильные устройства, серверные ЭВМ, настольные ЭВМ, одноплатные компьютеры, маршрутизаторы беспроводной локальной сети). Взаимодействие между такими агентами носит событийно-ориентированный, информационно-управляемый характер, т.е.

взаимодействие агентов происходит в зависимости от информации, доступной в общем информационном содержимом.

Для развертывания ИП используется специализированное промежуточное ПО (платформа), которое представляет собой программный слой, реализующий совместное использование и создание общего информационного содержимого программными агентами. Общее информационное содержимое реализуется не как традиционная реляционная база данных с предопределенной структурой, а как графовая нереляционная база данных, обслуживанием которой занимается связующий информационный центр. Его задача обеспечивать семантику происходящих процессов. Для информационно-управляемого взаимодействия используются известные модели «классной доски» и «публикации/подписки», которые реализуют механизм информационных уведомлений и подписки. В результате, каждый программный агент осведомлен о текущем, актуальном состоянии общего информационного содержимого.

Программные агенты запускаются на различных вычислительных устройствах IoT-среды. В результате, следует разделять программные агенты на два класса: (1) представители пользователей и (2) другие автономные участники интеллектуального пространства. Первые есть клиентские программные агенты, запущенные на вычислительных устройствах людей (смартфоны, планшетные компьютеры, настольные ЭВМ, ноутбуки и другие пригодные для персонального использования устройства). Пользователи предоставляют свою персональную и контекстную информация для построения сервисов и являются потребителями сервисов [19]. Остальные автономные участники интеллектуального пространства представляют собой программные агенты, которые реализуют построение и доставку сервисов. Одной из основных задач автономных участников является обеспечение виртуализации внешнего ресурса, интегрированного в ИП.

В соответствии с таблицей 3 представлены виды автономных участников ИП. Технический ресурс представляет собой физический объект (например, машина, механизм, прибор), а информационный ресурс является вычислительным элементом или элементом данных. Активный информационный ресурс является любым типом активного программного кода или программного обеспечения. В свою очередь, пассивный информационный ресурс является цифровым представлением объекта, хранящегося в некоторой информационной системе.

Таблица 3 — Автономные участники интеллектуального пространства

Ресурс	Класс	Пример
Технический ресурс	Мультимедийные устройства	Проектор, микрофон
	Серверная ЭВМ	Суперкомпьютер
	Персональная ЭВМ	Ноутбук
	Встроенная ЭВМ	Одноплатный компьютер
	Устройства сетевого взаимодействия	Маршрутизатор беспроводной сети
	IoT-устройства	Сенсор, актуатор, RFID-метка
Информационный ресурс	Активный информационный ресурс	внешний Интернет-сервис, стороннее ПО
	Пассивный информационный ресурс	База данных, информационное хранилище

Многие исследования в области ИП направлены на разработку программных платформ, определяемых как промежуточное ПО. Ключевые особенности промежуточного ПО состоят в интеграции гетерогенных вычислительных устройств и программных компонентов, а также в поддержке взаимодействия между различными устройствами ИП. Промежуточное ПО представляет собой программный слой с сетевыми возможностями, расположенный между системным программным обеспечением и прикладными системами, который направлен на поддержку требований для организации ИП. В подходе интеллектуальных пространств существует значительное разнообразие как в используемых коммуникационных технологиях, так и технологиях, используемых на уровне системного ПО. Промежуточное ПО должно учитывать обе эти особенности. Растет количество приложений ИП во многих предметных областях СИИО. Например, совместная деятельность, мобильное здравоохранение, музеи и культурное наследие. Требуется промежуточное ПО, которое предоставляет возможности для разработки таких приложений, причем ориентированных на контекст окружения и участников и основанных на многоагентном подходе [43; 64; 86; 115; 121].

В диссертационном исследовании для реализации программной инфраструктуру СИИО используется платформа Smart-M3 [111]. Данная платформа предоставляет решения в виде инструментальных средств в плане неоднородности и масштабирования программных систем, а поддерживаемые операции позволяют решать проблемы передачи данных и разнообразия структуры исполь-

зуемых данных через использования онтологий. Анализ возможностей платформы для создания СИИО подтверждается работами [110; 111].

1.5 Выводы

Определена концептуальная модель для содержательного описания структуры СИИО. Центральным понятием в СИИО выступает общее рабочее пространство, которое реализует основные функциональные свойства. Выполнен обзор основных классов программных систем, применяемых для разработки СИИО с учетом реализуемой функции (обмен сообщениями, обеспечение видео- и телеконференцсвязи, трансляция аудио и видео, проведение совещаний и др.). Выделены основные вычислительные элементы, обеспечивающие процесс совместного использования СИИО. В программной инфраструктуре эти элементы представлены агентами, каждый отвечает за определенный ресурс. Требуется реализация взаимодействия агентов в программном коде этих агентов. Для автоматизации программирования взаимодействия агентов сформулированы требования к интеграции ресурсов при построении сервиса взаимодействующими агентами. Также выполнен обзор существующих подходов и технологий в области исследования, чтобы оценить возможности повышения эффективности разработки для предлагаемого в следующей главе метода разработки интероперабельной программной инфраструктуры.

Глава 2. Разработка программной инфраструктуры для совместно используемого информационного интернет-окружения

В главе представлен новый метод разработки интероперабельной программной инфраструктуры СИИО для интеграции разнообразных ресурсов за счет унификации процесса разработки сервисов как системы взаимодействующих агентов с поддержкой автоматизации программирования взаимодействия агентов. Предлагается концептуальная модель информационного сервиса СИИО, предназначенная для проектирования сервиса как системы взаимодействующих агентов за счет создания унифицированного онтологического описания процессов построения сервиса с учетом контекста взаимодействующих агентов и задействованных ресурсов. Проектирование осуществляется за счет использования и расширения онтологического описания семантических веб-сервисов (онтология OWL-S) с использованием выделенных категорий сервисов и типовых моделей информационно-управляемого взаимодействия программных агентов при их построении. Предлагается алгоритм автоматизации программирования взаимодействия агентов на основе генерации программного кода с использованием онтологии предметной области и спецификации сервиса.

2.1 Метод разработки интероперабельной программной инфраструктуры

Метод разработки интероперабельной программной инфраструктуры обеспечивает информационно-управляемое взаимодействие агентов для интеграции динамических и неоднородных ресурсов в СИИО при построении сервиса с поддержкой автоматизации программирования взаимодействия агентов. Метод требует применения концептуальной модели информационного сервиса для проектирования сервиса как системы взаимодействующих агентов. Метод поддерживает применение предметно-ориентированных моделей проектирования при разработке сервиса. В качестве инструментального средства метод включает алгоритм автоматизации программирования взаимодействия агентов.

Метод направлен на создание интероперабельных программных инфраструктур СИИО, следуя технологиям IoT и Семантического веба: (1) цифровая

конвергенция информационных/технических объектов и людей, (2) сервисы определяются в контексте взаимодействия пользователей и ресурсов в среде обмена информацией (интеллектуальное пространство), (3) сервисы доступны единообразным семантическим способом. Таким образом, программная инфраструктура предоставляет возможности для разработки сервис-ориентированных прикладных систем на основе многоагентного подхода.

В программной инфраструктуре СИИО основным управляющим элементом обмена информацией является семантический информационный брокер, который предоставляет общую базу знаний, организует информационно-управляемое взаимодействие и поддерживает множество семантических интероперабельных примитивов доступа, в том числе и онтолого-ориентированных. Примитивы доступа обеспечивают возможность взаимодействия разнородных ресурсов на основе обработки и анализа общей информации, представляемой с помощью онтологий. На основе онтолого-ориентированных примитивов поддерживаются семантические рассуждения, которые позволяют преобразовать интегрированные низкоуровневые данные в высокоуровневые знания для решения возникающих проблем, а также для обеспечения самоорганизации участников [105].

За организацию процесса построения и доставки сервисов в СИИО отвечает программная инфраструктура, которая делает возможным функционирование сервисов. Программная инфраструктура развертывается в вычислительной среде ИП. В случае ИП с информационно-управляемым взаимодействием программная инфраструктура включает в себя следующие уровни (см. рисунок 2.1): (1) промежуточное программное обеспечение для обеспечения информационно-управляемого взаимодействия агентов; (агенты для обеспечения интероперабельности ресурсов; (3) сервисы как системы взаимодействующих программных агентов; (4) композиции сервисов для выполнения задач в СИИО.

Промежуточное ПО (платформа) для развертывания ИП представляет собой программный слой, реализующий совместное использование информационного содержимого программными агентами. В диссертационном исследовании рассматриваются ИП, реализуемые на основе архитектуры M3 и платформы Smart-M3 [110; 111]. Аббревиатура M3 указывает на свойства multi-device (множество устройств), multi-vendor (множество производителей аппаратуры) и multi-domain (множество предметных областей).

Каждый автономный агент в программной инфраструктуре СИИО работает в соответствии с определенной проблемной областью и моделью

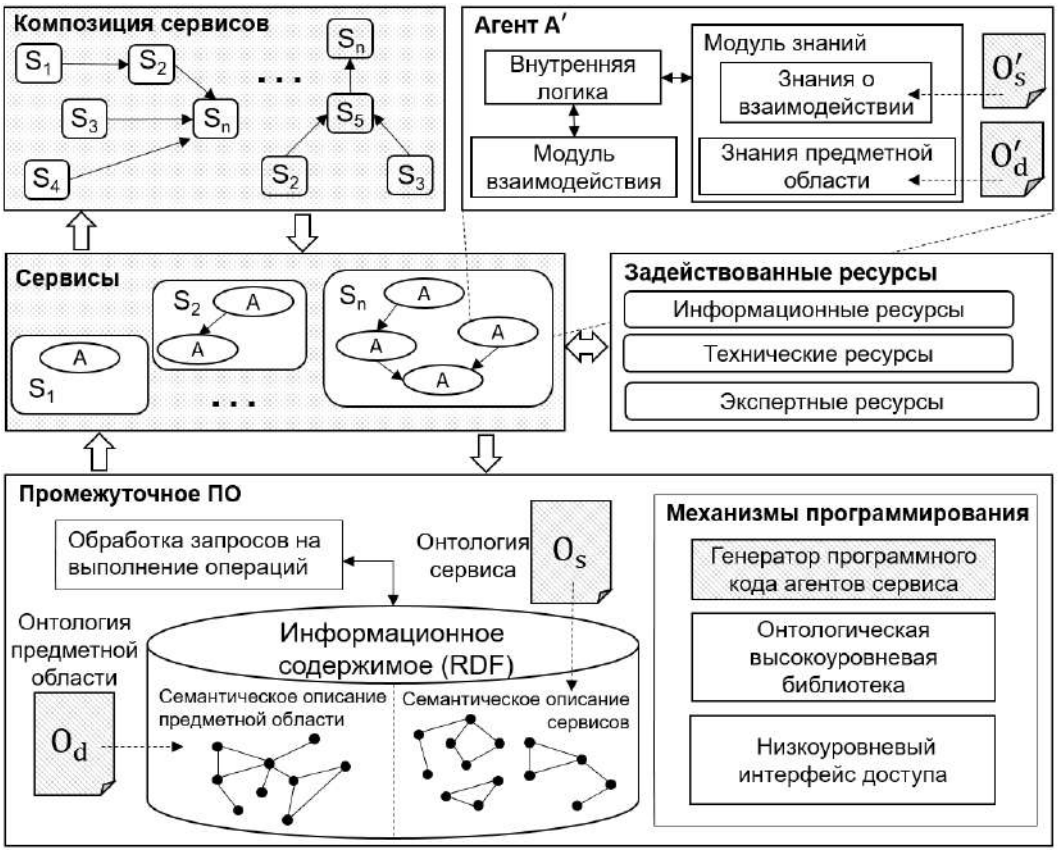


Рисунок 2.1 — Уровни программной инфраструктуры СИИО

информационно-управляемого взаимодействия с другими агентами в процессе построения и доставки сервисов. Разработчик программной логики агентов использует механизмы программирования для организации доступа к базе знаний (общее информационное содержимое). С точки зрения агентов хранение информации организованно как RDF-граф, как правило, в соответствии с OWL-онтологиями предметной области и спецификаций сервисов. Доступ к информационному содержимому может быть либо низкоуровневым, либо высокоуровневым. Низкоуровневый доступ требует, чтобы код агента работал с RDF-тройками напрямую, используя операции протокола доступа, где тройки являются основными элементами обмена. Высокоуровневый доступ определяет организацию кода агента на основе высокоуровневых объектов онтологии (классы, отношения, индивиды) с предопределенными структурами данных и методами API. Такие методы используют низкоуровневые операции доступа в качестве базового нижележащего слоя.

Построение сервиса в СИИО реализуется как распределенный вычислительный процесс, который позволяет создавать более сложные системные решения на основе информационно-управляемого взаимодействия агентов. Другими словами, сервис создается в результате совместной работы агентов. Такие агенты

выполняют пошаговый процесс изменения общего информационного содержимого для интеграции информационных, технических и экспертных ресурсов. Потребителями сервисов являются пользователи, поэтому процесс извлечения информации и доставки сервиса может быть персонализирован с учетом приоритетов и предпочтений пользователей, рассматриваемых в контексте текущей ситуации и состояния СИИО. Композиция сервисов определяется на основе последовательности сформулированных задач, каждую их которых выполняет тот или иной сервис.

Использование технологий Семантического веба для создания сервисов в рамках подхода ИП меняет требования к разработке программной инфраструктуры СИИО. В связи с постоянно растущим и динамически меняющимся набором ресурсов возрастает сложность этапов разработки и сопровождения сервисов. Определим основные требования к программной инфраструктуры СИИО для реализации их в заявленном методе разработки.

Унифицированная онтология сервиса. Проектирование семантических сервисов должно выполняться на основе общей унифицированной онтологии. Такая онтология определяет не только интерфейс сервиса с точки зрения передаваемых данных и возвращаемых значений, но и назначение сервиса. Описывается процесс его построения и однозначно определяется его семантика. Благодаря такому способу проектирования, сервисы различных СИИО приобретают возможность взаимодействовать друг с другом независимо от предметной области. Обеспечивая таким образом сетевое взаимодействие информационных окружений, можно добиться интеграции программных систем из разных СИИО для решения совместных задач.

Автоматизация процессов программирования агентов. Необходим способ разработки программной инфраструктуры, позволяющий уменьшить количество программного кода, создаваемого прикладным разработчиком во время выполнения рутинных задач, за счет использования инструментальных средств автоматизированного проектирования и программирования. В частности, автоматизация процессов программирования агентов при построении сервисов может быть достигнута за счет использования онтологий семантических сервисов. Онтологии принимаются в качестве входных параметров для генерации объектной модели и шаблонов программного кода для объектно-ориентированных языков программирования. Благодаря пониманию семантики сервиса, а также сведений о доступных ресурсах среды, на основе онтологий может быть достигнута самоорганизация

агентов путем определения их функциональных ролей, моделей взаимодействия и операций/функций в процессе построения и доставки сервиса [5].

Разработка программной инфраструктуры СИИО следует принципам онтолого-ориентированной разработки. Этап проектирования сводится к созданию спецификации определенной предметной области и сервисов в виде описаний в формате OWL/RDF. Использование онтологий позволяет добиться общего понимания структуры информационного содержимого между агентами, упрощения «повторного использования знаний за счет уже описанных в других онтологиях понятий и отношений» [6], а также поддержки формальной логики и логических рассуждений [122]. Таким образом, выгодно использовать особенности онтолого-ориентированного подхода в случае применения онтологий на всех этапах разработки.

Для обеспечения автоматизации этапов разработки традиционные методы проектирования и разработки заменяются методами, которые способствуют реализации подхода с широким использованием инструментальных средств автоматизированного проектирования (Computer-Aided Design, CAD) [27] и автоматизированного программирования (Computer-Aided Programming, CAP) [16]. Такие средства поддерживают прототипирование приложений.

Средства CAD используются для автоматизации процессов, направленных на создание и поддержку различных онтологических и графических представлений во время разработки программной инфраструктуры СИИО. В свою очередь, средства CAP ориентированы на упрощение задачи программирования агентов. Вместо прямого кодирования исполняемого кода разработчик предоставляет онтологию предметной области и онтологию спецификации сервиса. Такие онтологии является входными параметрами для алгоритмов генерации кода, которые позволяют создавать корректную объектную модель, структуры данных, методы, функции кода и другие элементы целевого языка программирования. Интеграция CAD- и CAP-средств приводит к автоматическому созданию программного кода непосредственно из этапа разработки спецификаций сервисов. Одним из способов распространения таких средств вместе с платформой является предоставление интегрированной среды разработки, развертывания и управления приложениями [93].

Этап проектирования сводится к созданию спецификаций предметной области и сервисов в виде RDF/OWL-описания. Существует большое количество работ, направленных на решение задач с помощью CAD-средств на этапе проек-

тирования для онтолого-ориентированной разработки ПО [122]. Например, такие инструментальные средства разработки онтологий, как Protégé [77] и OWL-S editor [116], позволяют создавать необходимые онтологические спецификации, предоставляя программную среду для быстрого прототипирования, в которой разработчики онтологий могут предельно быстро создавать индивидов, экспериментировать с семантическими ограничениями, визуализировать онтологические описания.

На этапе реализации агенты создаются из спецификаций и моделей, полученных на этапе проектирования, который предполагает использование языков программирования для кодирования отмеченных проектных решений. На этом этапе существующих CAP-средств [80] недостаточно для решения задачи автоматизации процессов онтолого-ориентированного программирования агентов. Такие CAP-средства только частично решают эту задачу путем преобразования OWL-классов, их экземпляров и свойств в соответствующие объекты языка программирования.

Такой подход сложен как при практической реализации, так и для использования в случае применения статически типизированных компилируемых языков программирования. Следует отметить, что он удобен для динамически типизированных интерпретируемых и объектно-ориентированных языков программирования. Такой подход прежде всего предназначен для создания структур данных и элементов объектной модели предметной области агента. Вместе с тем, он не позволяет создавать методы, функции и другие элементы внутренней программной логики. Одним из примеров реализации такого подхода является SmartSlogCodeGen, который является частью онтологической библиотеки SmatSlog, предназначенной для создания ИП-приложений [17]. Его механизмы позволяют создавать структуры данных для конкретных сущностей OWL-онтологии.

В соответствии с рисунком 2.2 представлена общая схема метода разработки интероперабельной программной инфраструктуры. Концептуальная модель информационного сервиса СИИО M_{srv} предназначена для проектирования сервиса как системы взаимодействующих агентов за счет создания унифицированного онтологического описания процессов построения сервиса с учетом контекста, взаимодействующих агентов и задействованных ресурсов. Проектирование осуществляется за счет использования онтологического описания семантических веб-сервисов с использованием выделенных категорий сервисов и типовых мо-

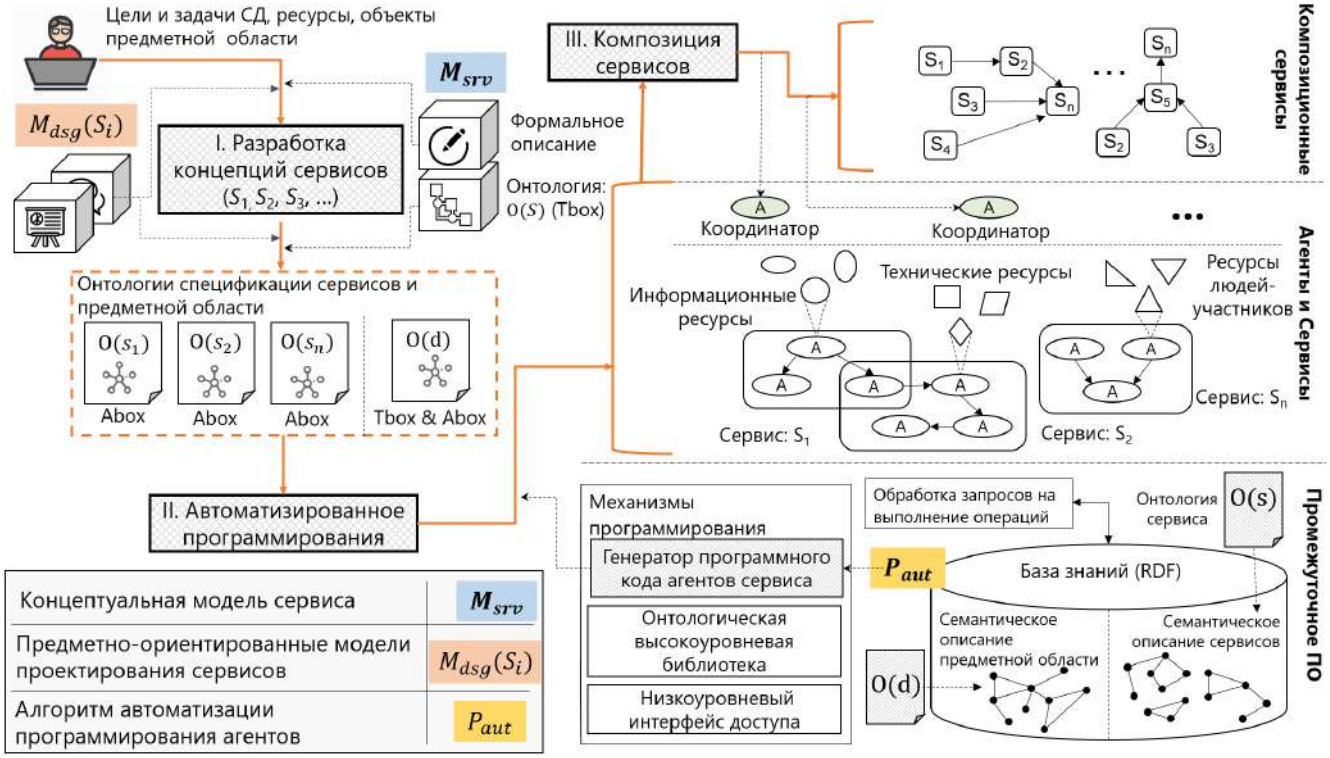


Рисунок 2.2 — Общая схема метода разработки интероперабельной программной инфраструктуры СИИО

делей информационно-управляемого взаимодействия программных агентов при их построении.

Предметно-ориентированные модели проектирования $M_{dsg}(S_i)$ позволяют разрабатывать специализированные сервисы на основе шаблонных решений для востребованных задач в СИИО: а) распознавание присутствия и анализ активности пользователей, б) сопровождение и визуализация плана деятельности людей, в) совместное пополнение информационного содержимого информацией о предметной области, г) мониторинг объектов физической среды. Эти модели (см. далее в главе 3) предлагают готовые архитектурные и поведенческие абстракции информационно-управляемого взаимодействия агентов, полученные на основе концептуальной модели информационного сервиса СИИО и подхода ИП.

Алгоритм автоматизации программирования взаимодействия агентов P_{aut} позволяет реализовывать интероперабельные структуры модели данных и реализации сценариев взаимодействия агентов при программировании сервиса на основе генерации программного кода. Алгоритмы используют в качестве входных параметров разработанные онтологические модели сервисов, полученные на основе моделей M_{srv} и $M_{dsg}(S_i)$.

В целом, метод разбивает разработку программной инфраструктуры на следующие этапы:

1. Разработка концепций необходимых сервисов (S_1, S_2, S_3, …): в соответствии с M_{srv} и $M_{dsg}(S_i)$.
    а) Определение подзадач для достижения общей задачи СИИО, задействованных ресурсов, объектов предметной области, прикладных функций.
    б) Определение основных сервисов и их категорий.
    в) Выбор типовой модели взаимодействия ролей агентов для построения сервиса, описание протокола взаимодействия агентов.
    г) Разработка онтологии спецификации сервиса (профиль, модель процессов) и онтологии предметной области.
2. Автоматизированное программирование взаимодействия агентов: в соответствии с P_{aut}.
    а) Генерация программного кода объектной модели.
    б) Генерация программного кода элементов сценариев поведения.
    в) Реализация индивидуальной внутренней логики.
3. Композиция сервисов: в соответствии с M_{srv}.
    а) Определение последовательности подзадач для достижения общей задачи.
    б) Разработка координирующего агента для распределения построения сервисов в соответствии с решаемыми задачами.

Таким образом, на первом этапе прикладной программист определяет цели и задачи СИИО, задействованные ресурсы, прикладные функции системы и объекты предметной области. Используя данную информацию, на основе концептуальной модели сервиса СИИО и ее формального описания программист определяет сервисы, используемые в системе, и их категории, определяет типовые модели взаимодействия агентов, необходимые для построения сервиса. Если для разрабатываемого сервиса есть предметно-ориентированная модель проектирования (некоторые из них предлагаются в главе 3), то программист использует готовые абстракции для упрощения проектирования сервиса. Далее, на основе общей унифицированное предлагаемой онтологии сервиса (содержащей набор терминологических аксиом), программист разрабатывает частную онтоло-

гию сервиса, включающую набор утверждений об индивидах, а также формирует онтологию предметной области.

2.2 Концептуальная модель информационного сервиса

Концептуальная модель информационного сервиса СИИО дает концептуальное описание сервиса СИИО с использованием категорий сервисов и типовых моделей организации информационно-управляемого взаимодействия агентов в соответствии с подходом интеллектуальных пространств. Для этого исследуется понятие сервиса в информационных окружениях и многоагентных системах с целью определения его формального описания в заданных условиях СИИО. Приводится набор функциональных ролей агентов, взаимодействующих между собой для построения сервиса. Основным назначением концептуальной модели является разработка унифицированного онтологического описания сервиса.

Сервис определяет процесс обслуживания пользователя в зависимости от его запросов, желаний. Понятие сервиса может быть определено в зависимости от выбранного уровня абстракции в программной системе. Сервис, например, может определять некоторую отдельную функцию, программный модуль или же специализированный процесс, выполняемый на основе заданного порядка действий. На основе обзора исследований, посвященных концептуальному моделированию сервисов [70; 88; 92; 106], цифровой или информационный сервис чаще всего определяется как отдельная прикладная функция программной системы, которая реализуется элементами программной инфраструктуры для реализации прикладной внутренней логики происходящих процессов как в самой системе, так и в сторонних приложениях других программных систем. Информационный сервис имеет следующие свойства: (1) доступ к сервису определяется с помощью описанного заранее интерфейса и протокола; (2) возможность многократного использования; (3) поддержка возможности взаимодействия сервисов друг с другом: (4) независимость отдельных сервисов.

Сервисы являются основными компонентами программной инфраструктуры СИИО. В результате, программная инфраструктура СИИО интегрирует разнородные ресурсы так, что они становятся частями общего рабочего пространства, доступ к которому происходит единообразным способом. Процесс выполне-

ния сервиса происходит между пользователем и информационным окружением, обеспечивая взаимодействие с информационными, техническими и экспертными ресурсами (в том числе и другими пользователями). Такие взаимодействия носят информационный характер, предоставляя необходимые фрагменты информации для решения возникающих проблем и выполняя воздействия на ресурсы. Процесс выполнения сервиса выполняется также с учетом приоритетов и предпочтений пользователей, определяемых на основе контекста текущего состояния информационного окружения. Потребителями сервиса могут выступать другие сервисы СИИО, реализуя цепочки выполнения или композицию сервисов. Пользователи могут наблюдать эффект выполнения сервиса в визуальной форме представления с помощью средств вычислительной среды и информационного окружения.

На концептуальном уровне сервис описывается на основе предлагаемой общей модели с помощью основных элементов программной инфраструктуры. Сервис представляет собой систему взаимодействующих программных агентов (распределенная динамическая неоднородная система) для реализации предписанной функции с интеграцией доступных ресурсов. Рассматривается как программная система, «доступ к которой предоставляется с помощью описанного заранее интерфейса и осуществляется в соответствии с указанными в описании сервиса ограничениями и политиками» [35], а выполнение следует заданному набору процессов (совокупности действий, повторяемых во времени). С точки зрения общего информационного содержимого сервис определяет процедуру извлечения и обработки знаний с формированием эффекта и результата выполнения, который размещается в общем информационном содержимом для доставки пользователю с помощью встроенных в промежуточное ПО механизмов (уведомления, подписка).

Сервис строится в результате взаимодействия реактивных агентов [34]. Причем один и тот же агент может являться участником построения нескольких независимых сервисов. Реактивные агенты, в отличие от интеллектуальных, «не способны планировать свои действия, поскольку реактивность подразумевает обратную связь, которая не содержит механизмов прогноза» [7]. Реактивные агенты не способны к целеполаганию и содержат знания о требуемых действиях до начала работы. Таким образом, нет необходимости строить подробное внутреннее представление окружения и внешней среды, так как «достаточным является оказание реакции на набор предъявляемых ситуаций, т.е. характер их реакций определяется только текущей информацией» [5; 34] и следует набору правил

«ситуация-действие». Иногда реактивным агентам приходится работать с неполной предопределенной информацией об окружении и среде, а также реагировать на изменения. Тогда в реализации агентов могут быть встроены механизмы машинного обучения для совершенствования своего поведения.

Сервис похож на организацию [34]. Термин организации выражает одновременно действие по организации чего-либо и результат этого действия. В рамках концептуального определения, извлечение информации – это действие, а доставка – результат действия. Сервис как организация выражает схему деятельности и взаимодействия агентов, задавая проблемную область, характер их задач и полномочий, а также определяет составляющие сервис-ориентированной многоагентной программной инфраструктуры. Использование сервисов ведет к организованности СИИО в целом, позволяя разбивать информационное окружение на отдельные части, отвечающие за выполнение определенных функций.

Можно выделить три класса сервисов с точки зрения выполняемых сервисами функций в СИИО:

1. *Сервисы обеспечения коммуникации*. Обеспечивают обмен данными (во всех форматах) между определенными группами пользователей. Данные могут передаваться синхронно или асинхронно.
2. *Сервисы обеспечения сотрудничества*. Позволяют участникам предоставлять свои (промежуточные) результаты другим и участвовать в дискуссионных форумах.
3. *Сервисы обеспечения координации*. Позволяют участникам, принадлежащим к одной команде, работать над одним и тем же набором файлов организованным и контролируемым образом.

Прикладные (пользовательские) сервисы отвечают за визуальное представление информации на персональных мобильных устройствах (напр., смартфоны и планшеты) и общих экранах (разнообразное медиа и видеооборудование, напр., видеопроекторы). Инфраструктурные сервисы производят информацию, используемую при построении прикладных сервисов. Прикладные сервисы виртуализируют технические ресурсы, представляющую медиа-оборудование для доставки информации пользователю. Таким ресурсом может выступать, например, проектор или TV-экран. Как правило, такой сервис реализуется с помощью агента-адаптера, который обеспечивает данный ресурс информацией, доставляя ее пользователю через интерфейсы вывода информации. Если результат работы инфраструктурного сервиса является входом для прикладного сервиса, а имен-

но, агента-адаптера, то в его построении, как правило, участвует агент-агрегатор, который преобразует информацию, полученную данным сервисом, в знания, пригодные для вывода на интерфейсные устройства.

Предлагается категоризация сервисов для СИИО с целью определения отношения сервиса к одной из категорий. Каждая определяет общие свойства сервисов. Сервис может быть отнесен более чем к одной категории, т.е. категории не являются взаимоисключающими. Определим следующие категории и критерии отнесения сервисов к ним:

1. *В зависимости от основополагающего ресурса*: сервис технического ресурса, сервис информационного ресурса, сервис экспертного ресурса.
2. *В зависимости от выполняемого воздействия на ресурс*: информационный сервис, воздействующий сервис.
3. *В зависимости от способа интеграции: неделимый (простой) сервис*, композиционный сервис.
4. *В зависимости от выполняемой функции в СИИО*: сервис обеспечения коммуникации, сервис обеспечение координации, сервис обеспечения сотрудничества, сервис обеспечения осведомленности.

Концептуальная модель информационного сервиса СИИО строится на основе концептуального описания семантических веб-сервисов с использованием категорий сервисов и моделей организации информационно-управляемого взаимодействия агентов, определенных в рамках подхода ИП. Предлагается формальное описание сервиса. Сервис определяется на основе совокупности следующих элементов:

$$S^k(r) = \langle \{A_t^c\}_{t=1}^n, M(k), I_{pr}, I_{rs} \rangle,$$

где $r \in \mathcal{R}$ — задействованный ресурс r из множества доступных ресурсов $\mathcal{R}$; $k \in \mathcal{K}$ — категория сервисов k из множества категорий $\mathcal{K}$; $\{A_t^c\}_{t=1}^n$ — агенты, участвующие в выполнении сервиса (t — номер агента, r — роль агента, n — количество агентов); $c \in \mathcal{C}$ — роль агента c из множества ролей $\mathcal{C}$; $M(k)$ — модель взаимодействия агентов (определяет протокол взаимодействия для каждого агента, архитектурные абстракции и типовую модель взаимодействия); I_{pr} — начальное состояние общего информационного содержимого; I_{rs} — результирующее состояние общего информационного содержимого.

На рисунке 2.3 представлена общая архитектурная схема сервиса, основанная на формальном описании. Разработка взаимодействующих систем и сервис-ориентированных архитектур в настоящее время движется в направлении

использования стандартов веб-сервисов. В сочетании с технологиями Семантического веба такие сервисы определяются как семантические т.е. такие сервисы имеют однозначно описанную семантику, что позволяет сделать их пригодными для автоматизации выполнения, обнаружения и композиции. Одно из назначений концептуальной модели сервиса - ввести унифицированное описание семантики сервиса, т.е. описать интерфейс (назначение сервиса, входные и выходные данные и т.д.) и происходящие процессы. Кроме того, описанная семантика должна учитывать многоагентный подход на основе информационно-управляемого взаимодействия при построении сервисов.

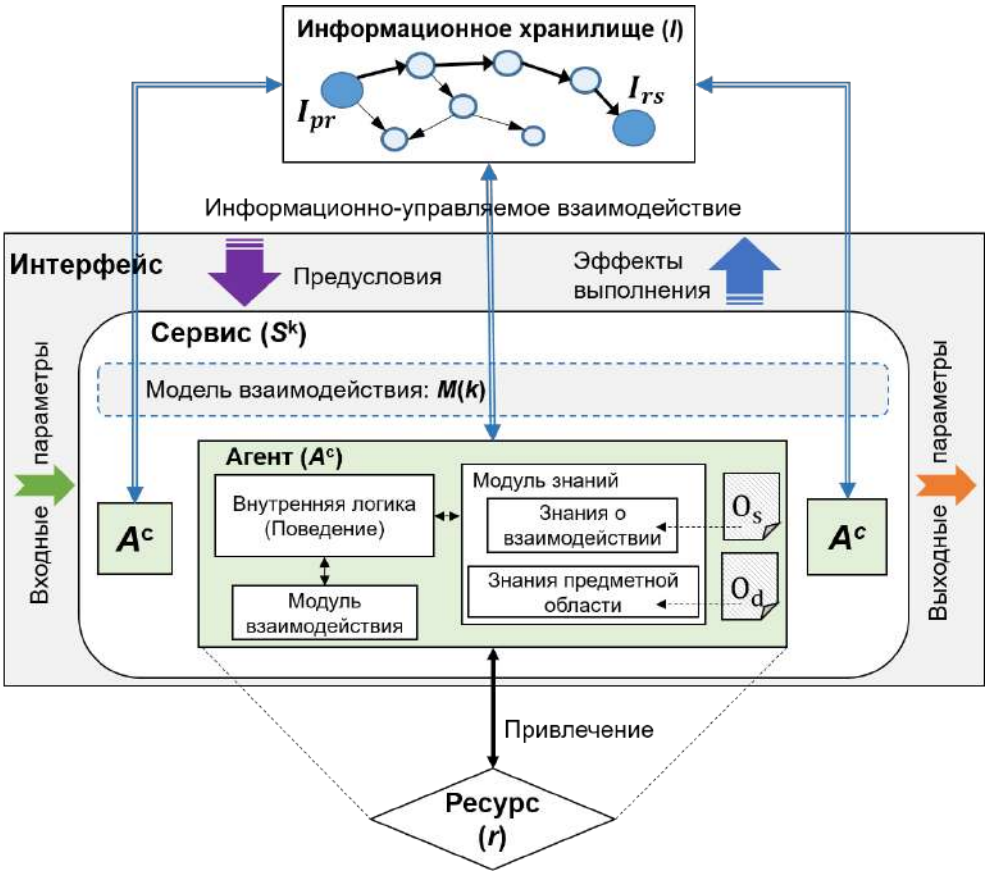


Рисунок 2.3 — Общая схема концептуальной модели информационного сервиса СИИО

В соответствии с таблицей 4 каждый агент выполняет определенную функциональную роль в зависимости от категории сервиса и способа его формирования, в построении которого участвует агент. Функциональная роль представляет собой не что иное, как абстрактное описание функциональных свойств программного агента (от англ. knowledge processor, KP). Роль агента позволяет определить общие принципы реализации его внутренней логики, а также принципы взаимодействия с другими агентами. Любая роль агента описы-

вается с помощью протокола, который определяет способы его взаимодействия с другими ролями. Такой протокол может строится, например, на модели IOPEs (inputs/outputs/preconditions/effects – входные параметры/выходные параметры-/предварительные условия/эффекты выполнения) [116]. На основе категорий сервисов и функциональных ролей формируются типовые модели взаимодействия агентов, которые включают описания поведения агентов на основе информационно-управляемого взаимодействия.

Таблица 4 — Функциональные роли программных агентов (ОИС — общее информационное содержимое)

Роль	**Описание агента**	**Задача**
Потребитель (Consumer)	Выступает посредником между потребителем-пользователем и ОИС.	Пополняет ОИС данными от пользователя, а также доставляет сервисы пользователю.
Адаптер (Adapter)	Выступает посредником между внешним ресурсом и ОИС.	Реализует цифровую виртуализацию ресурса, пополняя ОИС данными и доставляя необходимую информацию ресурсу.
Агрегатор (Aggregator, Reasoner)	Выполняет анализ накопленного ОИС.	Извлекает знания из имеющейся информации, включая публикацию результатов в ОИС.
Контроллер (Controller)	Выполняет динамическое обновление семантических связей в ОИС.	Обеспечивает отражение в ОИС текущего состояния СИИО и его контекста.
Фильтр (Filter)	Выполняет отбор и, при необходимости, преобразование информации из внешних источников для публикации в ОИС.	Обеспечивает ОИС содержимым на основе отбора информации из одного или более ресурсов.
Монитор (Monitor)	Отслеживает заданные изменения в ОИС и ведет соответствующую историю.	Извлекает информацию о проходящих в ОИС процессах во времени.
Искатель (Finder)	Выполняет поиск информации в ОИС или во внешних информационных ресурсах в зависимости от контекста запроса.	Обеспечивает агентов информацией в зависимости от их поискового запроса.
Оригинатор, создатель (Originator, Producer)	Представляет абстрактную сущность проблемной области сервиса, формируя ОИС.	Выполняет наполнение ОИС необходимой информацией для функционирования сервиса, а также отправляет управляющие воздействия другим агентам.

Продолжение таблицы 4

Роль	Описание агента	Задача
Координатор (Coordinator)	Позволяет распределить выполнение задачи между набором других агентов.	Отправляет агентам управляющие воздействия посредством операций публикации и подписки, позволяет инициировать работу других агентов.
Интегратор (Integrator)	Позволяет сформировать общий результат на основе результатов решения задач другими агентами.	Выполняет анализ ОИС, а затем композицию и интеграцию частных результатов, в зависимости от его цели.
Интерпретатор (Interpreter)	Представляет посредника между агентом и другим информационным ресурсом, не имеющим возможности для взаимодействия с ОИС.	Обеспечивает ОИС информацией от информационного ресурса и наоборот, выполняет интерпретацию передаваемой информации, делая ее понятной той и другой стороне.

Модель взаимодействия программных агентов служит для получения информации о принципе процесса взаимодействия, возникающего между агентами во время операций построения и доставки сервисов. Такая модель представляется с помощью языка моделирования AUML — расширения языка UML, специализированного для описания многоагентной системы [90]. С помощью таких моделей могут быть построены типовые модели взаимодействия, которые включают список ролей агентов, участвующих в построении и доставке сервиса, а также шаблон взаимодействия. Например, для сервиса класса «Неделимый, сервис технического ресурса, информационный сервис» может соответствовать список ролей агентов вида «Технический ресурс – Адаптер – Агрегатор – Монитор», которые взаимодействуют на основе шаблона «Взаимодействие один-ко-многим». В соответствии с рисунком 2.4 представлен пример модели взаимодействия агентов для такого класса сервисов.

Онтологическая модель сервиса СИИО позволяет описывать контекст окружений и его участников, взаимодействующих агентов и задействованных ресурсов. Онтологическая модель строится на основе концептуального описания семантических веб-сервисов с использованием моделей организации информационно-управляемого взаимодействия агентов [16], определенных в рамках подхода ИП. За счет такого унифицированного онтологического описания процессов построения сервиса достигается повышение качества проектирования сервисов. Таким образом, сервисы становятся пригодными для автоматизированного поиска, композиции, построения и доставки пользователям посредством

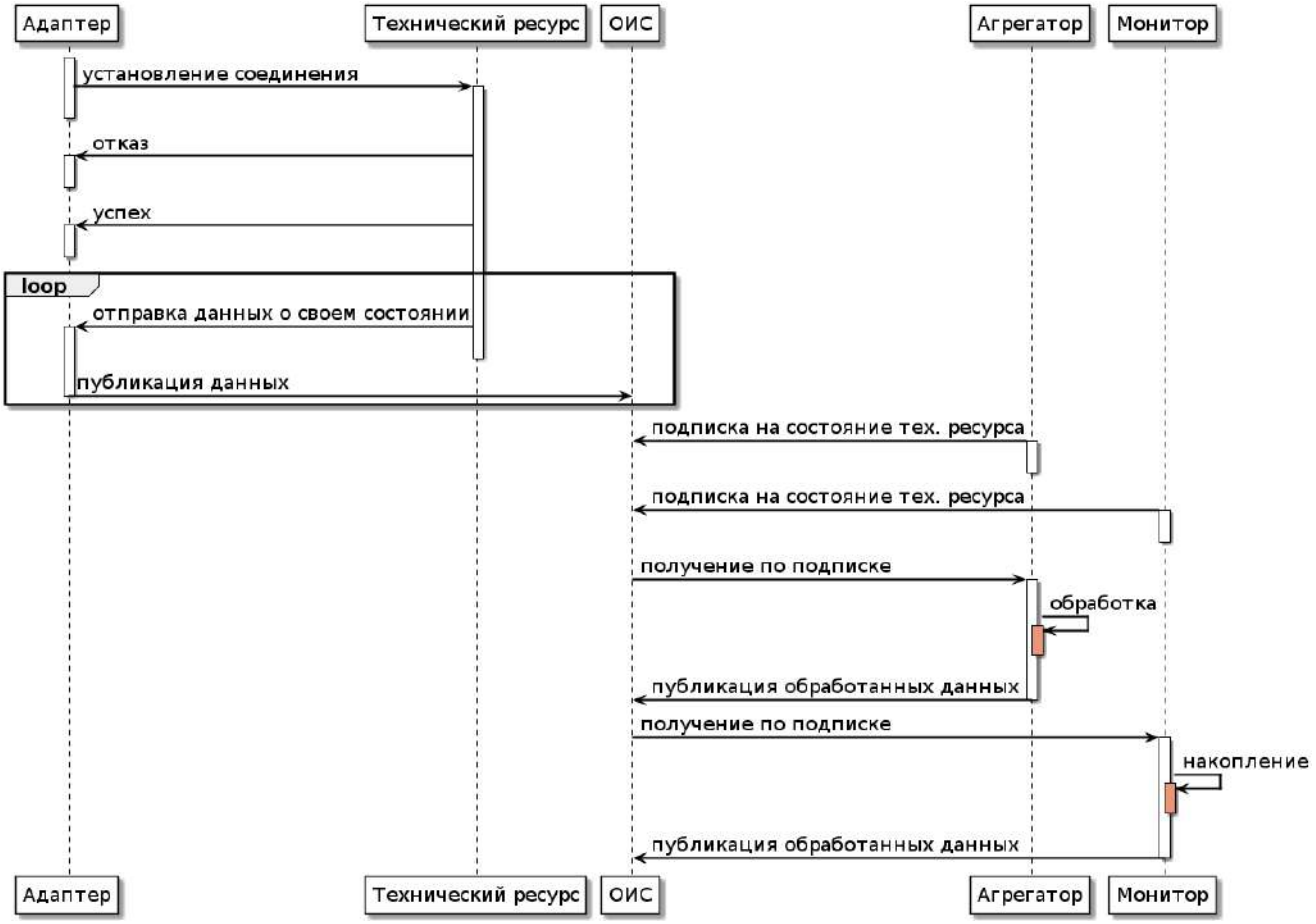


Рисунок 2.4 — Пример типовой модели взаимодействия агентов (технический ресурс)

описания интерфейса (назначение сервиса, входные и выходные данные, и т.д.) и происходящих процессов.

Известна онтология OWL-S, разработанная поверх OWL для описания семантических веб-сервисов [94; 116]. Основная цель онтологии OWL-S: предоставить пользователям возможность с высокой степенью автоматизации обнаруживать, выбирать, вызывать, составлять, отслеживать и контролировать веб-ресурсы. Онтология OWL-S позволяет описывать характеристики сервисов, используя три концепции верхнего уровня, представленные в виде следующих онтологий: профиль сервиса (service profile); модель процесса сервиса (service model); основание сервиса (service grounding).

Назначение профиля сервиса — определить сервис единообразным образом для дальнейшего использования, детализируя содержание запросов и условий, при которых будут возникать конкретные результаты и, при необходимости, поэтапные процессы, приводящие к этим результатам. Модель процесса описывает, как получать доступ к сервису и что происходит, когда выполняется

сервис. Модель процесса сервиса описывает сервис как набор атомарных и составных процессов. Атомарный процесс соответствует процедуре, которая получает входные параметры, обрабатывает их и затем возвращает результат. Составные процессы разлагаются на более простые процессы, определяя их декомпозицию с помощью управляющих конструкций (например, If-Then-Else, Split, Repeat-While). Основание сервиса определяет, каким образом сервис вызывается потребителем сервиса, включая протокол взаимодействия и формат сообщений. В случае архитектуры M3 [111] онтология основания сервиса является необязательной частью, т.к. взаимодействие агентов организуется посредством протокола SSAP, который не требует отдельного описания.

Предлагается унифицированное онтологическое описание семантики сервисов для СИИО, содержащее набор терминологических аксиом (см. рисунок 2.5). Профиль сервиса в онтологии описывает сервис с помощью модели IOPEs: входные параметры (input), выходные параметры (output), предварительные условия (precondition), эффекты выполнения (effects). Свойство has_output используется для представления выходных данных от ресурса сервиса к потребителю. Если сервису необходимо задать входные данные для обработки с помощью ресурса или управляющее воздействие в виде уведомлений, то используется свойство has_input.

Исходное (начальное) состояние общего информационного содержимого для инициализации процесса построения сервиса задается с помощью предварительного условия (has_precondition). Желаемое состояние общего информационного содержимого определяется с помощью результирующего условия (has_result). Предусловия и результаты задаются с помощью логических выражений. Существует несколько возможных подходов, которые позволяют использовать правила и логику для RDF/OWL представления [95]. Ключевой идеей является рассмотрение логических выражений как литералов (literals), используя SPARQL-выражения, которые определяются с помощью класса SPARQL-Expression.

Свойство has_category описывает категории сервисов, основанные на категориях сервисов для СИИО. Категории определяются в зависимости от таких критериев, как оказываемое воздействие, основополагающий ресурс, способ интеграции, выполняемая функция. Категория определяет типовую модель информационно-управляемого взаимодействия агентов.

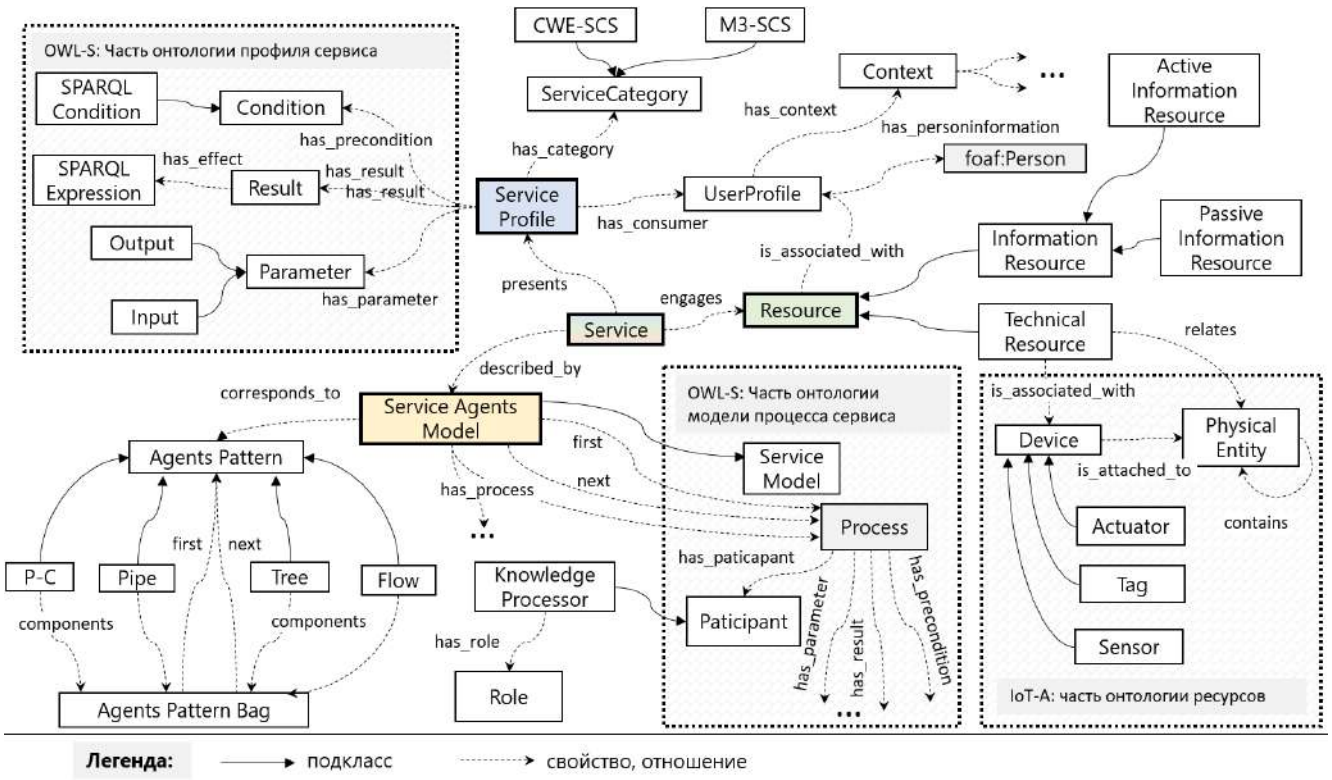


Рисунок 2.5 — Онтология сервиса для СИИО

Онтология OWL-S довольно гибкая, однако решений с ее использованием недостаточно, чтобы без изменений использовать ее для описания семантики сервисов для СИИО. Вводятся новые сущности, чтобы учесть приоритеты и предпочтения потребителя (в случае, если потребитель является человеком), обеспечивая персонификацию сервиса. Существующая онтология профиля расширяется свойством has_consumer и его объектом класса UserProfile, который описывается в терминах пользовательской контекстной информации. Такая информация отражает состояние окружения пользователя (has_context), а также его предпочтения, интересы, персональную информацию, представленную частью онтологии FOAF (has_personinformation) [114]. Онтология OWL-S не поддерживает многоагентный подход для описания модели построения и доставки сервиса. Представление агентов в онтологии OWL-S является редуцированным: агенты учитываются, прежде всего, как потребители или «искатели» сервисов, в то время как процесс построения сервиса в подходе ИП представляет собой процесс взаимодействия нескольких агентов.

Онтология модели процесса сервиса расширяется за счет внедрения класса ServiceAgentsModel для описания процесса построения сервисов для СИИО. Процесс описывается моделью информационно-управляемого взаимодействия агентов, которая определяется в зависимости от категории сервиса. Взаимо-

действие каждого агента (KnowledgeProccessor), представленного как расширение класса участника, определяется его функциональной ролью (свойство has_role), выполняемой в модели. Логика отдельного агента ведет к взаимодействию агентов во время построения сервиса на основе шаблонов взаимодействия (AgentsPattern), описанных с помощью таких архитектурных абстракций, как Поставщик-потребителей (P-C), Конвейер (Pipe), Дерево (Tree), Поток (Flow) [15]. Процесс взаимодействия агентов может состоять из нескольких шаблонов, представленных в некоторой последовательности (AgentsPatternBag).

Использование технологий Семантического веба для создания сервисов в рамках подхода ИП меняет требования к разработке программной инфраструктуры СИИО. В связи с постоянно растущим и динамически меняющимся набором ресурсов возрастает сложность этапов разработки и сопровождения сервисов. Должны использоваться подходы к разработке, не требующие интенсивного вовлечения разработчиков, и с широким использованием средств автоматизированного программирования, которые обеспечивают прототипирование сервисов. Представленная онтология, которая является частью концептуальной модели сервиса, позволяет описывать контекст СИИО и его пользователей, взаимодействующих агентов и задействованных ресурсов. За счет такой унифицированной онтологической модели сервисы становятся пригодными для автоматизированного программирования посредством описания интерфейса и происходящих процессов. В разделе 2.3 предлагается алгоритм автоматизации программирования взаимодействия агентов, направленный на создание инструментального средства, позволяющего уменьшить количество программного кода, создаваемого прикладным программистом вручную при выполнении задач по разработке программной инфраструктуры.

2.3 Алгоритм автоматизации программирования взаимодействия агентов

В диссертационной работе предлагается решение, направленное на разработку алгоритма автоматизации программирования взаимодействия агентов СИИО, обеспечивающего кодогенерацию программных механизмов информационно-управляемого взаимодействия по заданной спецификации проектирования сервиса. Создается генератор программного кода агентов по

онтологии при использовании объектно-ориентированных языков программирования (C++, Java) [21; 22]. Общая схема процесса генерации программного кода показана на рисунке 2.6. Основными особенностями предлагаемой схемы генерации кода являются: использование в качестве входных онтологий, совместно с OWL онтологиями предметной области, онтологий сервиса СИИО на основе онтологии OWL-S [94]; генерация, в дополнение к структурам данных, элементов программной логики агентов на основе API ИП-платформы с целью построения и доставки сервисов, а также объектной модели предметной области.

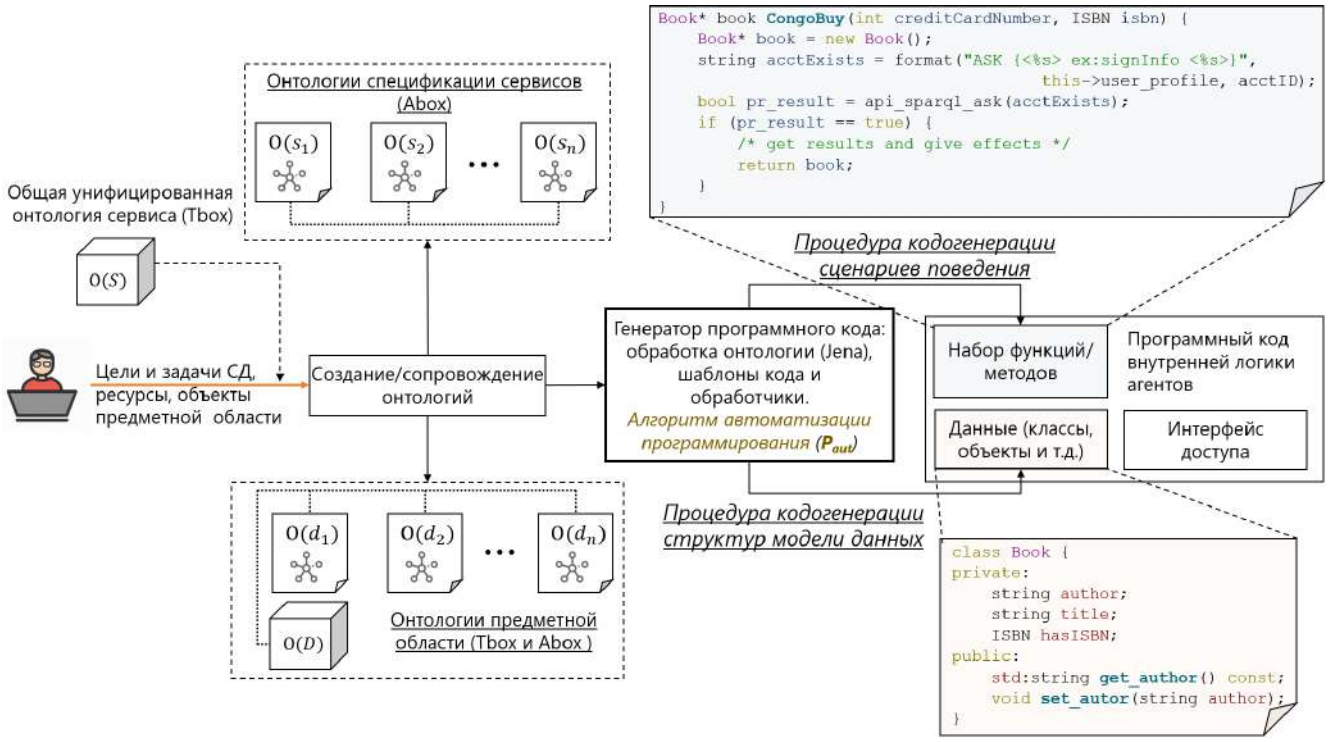


Рисунок 2.6 — Общая схема процесса генерации программного кода взаимодействия агентов

Разработчик агентов предоставляет спецификацию проблемной области как описание OWL и спецификацию сервисов СИИО как описание, основанное на OWL-S. Генератор использует статическую схему шаблонов и обработчиков. Шаблоны кода представляют собой «предварительный код» классов, атрибутов и функций, которые реализуют сущности и свойства онтологий OWL, а также сервиса, основанного на OWL-S. Обработчики могут преобразовывать один или несколько шаблонов в конечный код, заменяя специальные теги именами и элементами, взятыми из онтологий. Код может быть сгенерирован для нескольких агентов, в зависимости от того какие агенты указаны в спецификации сервиса для его построения и доставки. Преобразование происходит во время обхода RDF-графа онтологии с использованием фреймворка Jena. С его помощью строится

метамодель для представления графа. На основе информации, полученной во время обхода генератором узлов метамодели, обработчики преобразуют шаблоны в конечный код.

Автоматизация процессов программирования агентов, участвующих в построении и доставке сервисов, достигается за счет использования генератора программного кода. В результате выполнения этапа проектирования сервисов разработчик имеет набор онтологий, которые делятся на две группы: (1) онтология проблемной области и (2) онтология спецификации сервиса для СИИО на основе OWL-S. Онтологии предоставляют необходимую семантику, которая используется для генерации кода объектно-ориентированных языков программирования. Генератор использует алгоритмы автоматизации процессов программирования агентов для реализации структур объектной модели данных и для взаимодействия агентов при программировании сервисов. Алгоритм кодогенерации объектной модели данных агентов принимает в качестве входного параметра онтологию предметной области. Алгоритм кодогенерации взаимодействия агентов принимает в качестве входного параметра онтологию спецификации сервиса для генерации блоков программного кода агентов КР.

Объектная модель объединяет данные и функциональные возможности в переменной абстрактного типа данных (класса) — объект. Объектная модель позволяет отразить «понятия и объекты реального мира, которые важны для разрабатываемой системы» [9], т.е. определяют ее предметную область. В то время как структура онтологии содержит определения понятий/концептов (классов) и отношений между ними (свойства, аспекты, параметры), объектная модель использует классы для представления объектов и функции для моделирования отношений объектов и атрибутов. Сходство понятий в онтологии с объектной моделью определяет применимость объектно-ориентированного подхода для моделирования онтологий [51]. Однако онтология представляет более богатую информационную модель, чем объект языка программирования (например, Java), поскольку поддерживает такие отличительные особенности, как наследование свойств, симметричные/транзитивные/обратные свойства, полное множественное наследование среди классов и свойств [108].

Таблица 5 — Сводное изложение правил отображения онтологии в объектную модель

Обозначение	**Разъяснение**
1. Ontology classes → Object classes	Онтологические классы близки к классам в объектной модели, представляя абстрактные группы физических или логических объектов, тогда как объектный класс может быть рассмотрен как определение типа для объекта. Классы онтологии отображаются как классы объектов.
2. Instances → Objects	Экземпляры в онтологии используются для представления определенных представителей классов. Класс объекта служит шаблоном описания его объекта. Экземпляры в онтологии отображаются как объекты классов.
3. Data type properties or slots → Data attribute variables & get/set methods	Слоты данных представляют свойства (атрибуты) данных класса онтологии. Эти слоты описываются простыми типами (целочисленные, логические, строковые и т.д.) и наборами значений переменных атрибутов этих типов. Переменные атрибутов в объектной модели описывают характеристики объектов различными типами данных. Свойства данных или слоты отображаются как переменные данных атрибутов вместе с комбинацией методов get/set.
4. Object type properties or slots → Object attribute variables & get/set methods	Объектные свойства или слоты используются для описания взаимосвязи между двумя понятиями (концептами). Первый должен быть экземпляром класса, который является доменом слота (domain), второй – экземпляр класса, который описан допустимым диапазоном слота (range). Объектные свойства или слоты отображаются как объектные переменные атрибутов вместе с комбинацией методов get/set.
5. Value-type/space facets → Attribute variables types & if-then-else statements	Аспекты или грани (facets) для типов данных в онтологии – это некоторое бинарное отношение, которое присоединяется к слоту и описывает какие типы значений данных могут заполнять слот и какие ограничения они имеют (макс. и мин. значения и т.д.). Грани применимы к переменным атрибутов в объектной модели, например, аспект «xsd:type» может использоваться для обозначения типа переменной, а аспект «xsd:lenght» может быть представлен набором операторов «if-then-else» в методе set для соответствующего атрибута, задавая ограничения на длину значения переменной.
6. Cardinality facets → Additional attributes & if-then-else statements	Аспекты или грани кардинальности определяют, сколько значений может иметь свойство или слот. Некоторые аспекты кардинальности позволяют указать минимальное и максимальное значение кардинальности для более точного описания количества значений слотов. Дополнительные переменные атрибутов могут быть добавлены к объектной модели для указания аспектов кардинальности. Некоторые из этих аспектов могут быть представлены как набор операторов «if-then-else» в методе set для соответствующего атрибута, ограничивая его кардинальность.

Продолжение таблицы 5

Обозначение	Разъяснение
7. Single class inheritance → Single object inheritance	Одиночное наследование классов в онтологии отображается в одиночное наследование классов объектов. Для этого название родительского класса в форме онтологического наследования, подставляется в форму наследования для выбранного объектно-ориентированного языка программирования. Родительский класс должен быть определен в объектной модели данных.
8. Multiple inheritance → Single inheritance & multiple interface inheritance	Некоторые языки программирования (например, Java) не поддерживают множественное наследование классов из-за так называемой проблемы ромба (diamond problem). Однако для решения большинства задач может применяться множественное наследования интерфейсов наряду с одиночным наследованием.

Принимая во внимание упомянутые выше аспекты и ограничения, а также результаты исследований в этой области [62; 82], в таблице 5 сведены правила отображения, используемые при генерации исходного кода объектной модели данных по онтологии. Далее представлен разработанный алгоритм автоматизации программирования взаимодействия агентов СИИО, использующий правила отображения из соответствующей сводной таблицы и представленный в виде блок-схемы (см. рисунок 2.7). Основная идея этого алгоритма заключается в создании набора классов и объектов таким образом, чтобы каждый онтологический класс со своими экземплярами, свойствами, слотами, аспектами и ограничениями имел свой эквивалент в структурах объектно-ориентированного языка программирования.

Построение и доставка сервиса ИП может быть рассмотрена как набор вызовов программных функций/процессов агентов. Онтология на основе OWL-S для сервисов ИП предоставляет декларативное, интерпретируемое компьютером описание, которое включает семантику модели IOPEs, которая должна быть указана для каждого такого вызова. Сущность процесса из онтологии сервиса ИП может быть использована для генерации процедур/функций и других элементов целевого языка программирования (см. рисунок 2.8). Экземпляры класса AtomicProcces используются для генерации кода функций. Атрибут rdf:ID класса AtomicProcess определяет имя функции. Каждое свойство hasInput с атрибутом rdf:ID соответствует входным параметрам функции. Тип входного параметра получается путем просмотра свойства parameterType.

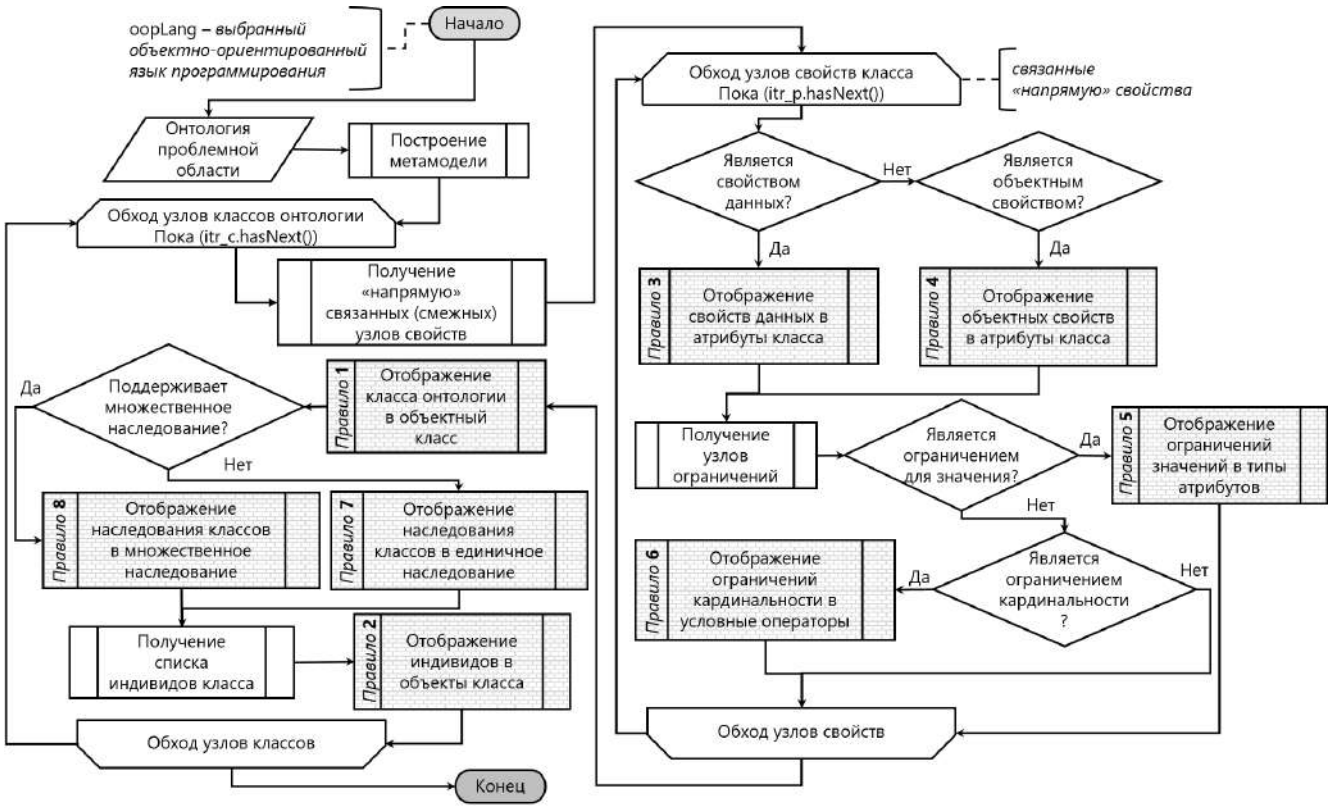


Рисунок 2.7 — Алгоритм автоматизации программирования взаимодействия агентов: процедура кодогенерации объектной модели данных агентов

Основная внутренняя логика функций реализуется с использованием запросов SPARQL и наборов операторов языка программирования. Свойство has_precondition с выражением SPARQL определяет блок кода для проверки предварительного условия, описывающего начальное состояние общего информационного содержимого необходимое для инициализации построения сервиса. Генерируется блок кода, который вызывает функцию API промежуточного ПО (платформы ИП) для выполнения запроса SPARQL (обычно запрос ASK) и проверяет результат запроса с помощью оператора «if-then-else». Аналогичный процесс генерации выполняется для блока кода, представляющего результирующее условие (has_result), набор действий, выполняемых в конце вызова функции.

Свойство has_output со свойством parameter_type соответствует выходному параметру и определяет возвращаемое значение функции. С помощью языка описания схем XSD (XML-Schema) описываются необходимые типы данных (string, unsignedLong и т.д.). Элементы объектной модели могут использоваться в качестве входных и выходных параметров функций. Класс CompositeProcess по аналогии с составным процессом определяет функцию, которая вызывает внутри себя другие функции, которые описываются сущностями CompositeProcess или

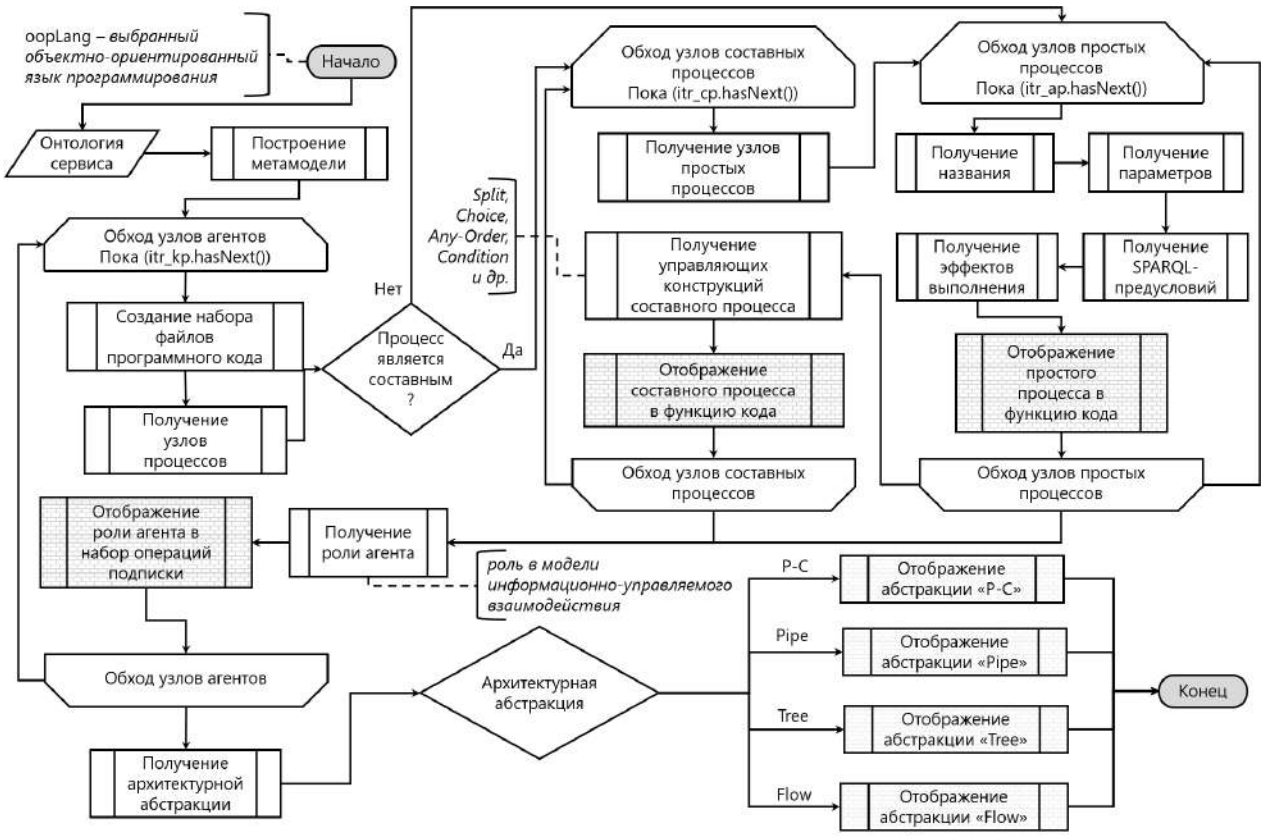


Рисунок 2.8 — Алгоритм автоматизации программирования взаимодействия агентов: процедура кодогенерации сценариев взаимодействия агентов

AtomicProcess. В этом случае вызовы внутренних функций могут быть заданы с помощью управляющих конструкций (If-Then-Else, Repeat-While, и т.д.), которые, в свою очередь, могут быть преобразованы в соответствующие выражения (конструкции) языка программирования.

Экземпляр класса ServiceAgentsModel используется агентами для определения их роли в процессе построения и доставки сервисов, а также способа информационно-управляемого взаимодействия, основанного на модели «публикации/подписки». Для этого в коде каждого агента генерируется блок с необходимыми операциями подписки с использованием внутренних функций, обработчиков и функций API. Каждый запрос операции подписки задается запросом SPARQL, который может быть получен из свойства has_subprecondition, где выражение подписки задается с использованием SPARQL-запросов, которые используют классы и свойства предметной области. Кроме этого операции подписки могут быть определены на основе результатов выполнения процессов агентов, определенных в классах Result. Каждый такой класс определяет какие изменения ОИС производятся во время выполнения процессов агента KP. Во время выполнения операций агентами может возникать их самоорганизация, поскольку каждый

агент КР осведомлен о своей роли в модели информационно-управляемого взаимодействия, основанной на шаблонах взаимодействия (AgentsPattern).

2.4 Выводы

В главе представлен новый метод разработки интероперабельной программной инфраструктуры СИИО для интеграции разнообразных ресурсов за счет унификации процесса разработки сервисов как системы взаимодействующих агентов с поддержкой автоматизации программирования взаимодействия агентов. Метод основан на применении в нем других результатов исследования. Предложенная концептуальная модель информационного сервиса СИИО позволяет проектировать сервис как систему взаимодействующих агентов за счет использования унифицированного онтологического описания процессов выполнения сервиса с учетом контекста, взаимодействующих агентов и задействованных ресурсов. Полученная онтология сервиса предоставляет необходимую семантику, которая используется для генерации кода взаимодействующих агентов для объектно-ориентированных языков программирования. Предложенное проектное решение генератора основано на использовании алгоритма автоматизации программирования взаимодействия агентов. Разработанный алгоритм позволяет увеличить объем автоматически генерируемого программного кода для реализации интероперабельных структур модели данных и сценариев поведения агентов при программировании сервиса.

В следующей главе предлагаются предметно-ориентированные модели проектирования сервисов СИИО, которые используются в представленном методе на этапе разработки концепций и моделей сервисов в качестве шаблонных решений для востребованных приложений СИИО.

Глава 3. Проектирование контекстных сервисов

В главе представлены предметно-ориентированные модели проектирования сервисов для выполнения прикладной разработки на основе шаблонных решений для следующих востребованных приложений СИИО: распознавание присутствия и анализ активности пользователей; сопровождение и визуализация плана деятельности людей; пополнение информационного содержимого знаниями о предметной области; мониторинг объектов физической среды. Востребованность выбранных приложений обосновывается актуальным направлением происходящего процесса информатизации в таких сферах жизни общества, как цифровая экономика, наука и образование, культура. Более того, каждое из приложений направленно на обеспечение осведомленности пользователей, что является необходимым условием для их совместной работы в СИИО. Представленные предметно-ориентированные модели проектирования сервисов используются разработчиком при выполнении первого этапа («разработка концепции необходимых сервисов») предложенного метода для снижения трудозатрат на проектирование за счет предоставления архитектурных и поведенческих абстракций взаимодействия агентов, а также частной онтологии сервиса.

3.1 Понятие предметно-ориентированной модели проектирования сервиса

Процесс разработки программной инфраструктуры для рассматриваемых востребованных приложений СИИО приводит к значительным трудозатратам. Из-за отсутствия унифицированной онтологии для проектирования сервиса прикладному разработчику необходимо для каждого сервиса, количество которых может динамически изменяться в зависимости от привлекаемых ресурсов, проектировать собственную онтологию. Такая онтология, как правило, не определяет интерфейс сервиса и процесс его построения, а описывает только задействованные ресурсы и другие объекты предметной области. Отсутствие единообразного интерфейса у сервисов (входные и выходные параметры, предусловия и результаты) ограничивает возможности их композиций. Например, для случая интеллектуального зала, результат выполнения сервиса распознавания присут-

ствия участников (уровень присутствия) может служить входным параметром для сервиса сопровождения и визуализации плана деятельности. Этот результат инициализирует процесс формирования приветственных сообщений на экранах и используется для предоставления контекста о присутствии участников во время управления программой конференции или семинара.

Предметно-ориентированные модели проектирования сервисов СИИО предназначены для выполнения прикладной разработки на основе шаблонных решений для востребованных приложений и задач. Такие модели используются на первом этапе предложенного метода разработки интероперабельной программной инфраструктуры (раздел 2.1), что позволяет снизить трудозатраты на разработку. Предметно-ориентированные модели проектирования сервисов предлагаются разработчику в виде архитектурных и поведенческих абстракций (моделей) информационно-управляемого взаимодействия агентов.

Такие абстракции строятся на основе предложенной концептуальной модели сервиса СИИО (раздел 2.2) и подхода интеллектуальных пространств. Для каждого рассматриваемого приложения СИИО предлагается: 1) архитектурная модель сервиса (взаимодействующие агенты, участвующие в построении сервиса); 2) поведенческая модель в виде диаграммы последовательности (взаимодействие агентов и используемые при этом данные предметной области).

Востребованность выбранных приложений обосновывается актуальным направлением происходящего процесса информатизации в таких сферах жизни общества, как цифровая экономика, наука и образование, культура. Более того, каждое из приложений направленно на обеспечение осведомленности пользователей, что является необходимым условиям для их совместной работы в СИИО.

В зависимости от выбранной предметной области формируется совокупность информационных объектов с помощью частной онтологии сервиса, включающей набор утверждений об индивидах и знаний предметной области. Определенное таким образом семантическое содержимое используется для построения и доставки сервисов в СИИО. Взаимодействие программных агентов сводится к информационному обмену структурированной информацией с помощью основных операций агентов.

Проектирование сервисов основано на сценариях, включающих набор действий по извлечению знаний из общего информационного содержимого. Сценарий определяет набор действий для выполнения в некоторой последовательности шагов процесса построения сервиса. Выполнение сценария инициируется пользо-

вателями в СИИО и завершается процессом доставки, в том числе и проактивной (на основе известных предпочтений и потребностей пользователей). Более того, пользователь может запросить явное выполнение сервиса. Результат сервиса предоставляется пользователю с помощью графического интерфейса или пользователь может наблюдать изменения в СИИО.

3.2 Сервис распознавания присутствия и анализа активности пользователей

Предлагается предметно-ориентированная модель проектирования для сервиса распознавания присутствия и анализа активности пользователей (S_{prs}). Осведомленность сервисов для СИИО расширяется за счет предоставления информации о присутствии и об активности пользователей, а также об использовании их персональных мобильных устройств (например, смартфоны, ноутбуки). Такая информация позволяет инициализировать процессы построения других сервисов. Например, результат выполнения сервиса S_{prs}, содержащий информацию о присутствии пользователей, может использоваться сервисом визуализации программы совместной деятельности, отображая на общем мультимедийном экране статус присутствия пользователей или приветственные всплывающие сообщения. Сбор всех участников мероприятия к определенному времени в помещении может служить поводом для начала выполнения программы совместной деятельности [45; 85].

Информация о присутствии и об активности пользователей накапливается на основе отслеживания сетевой активности мобильных устройств на уровне передачи пакетов данных. Процесс отслеживания реализуется с применением технологии пассивной радиолокации на основе измерения уровня принимаемого сигнала (от англ. «received signal strength indication», RSSI) для передаваемых пакетов. Сервис позволяет отслеживать сетевую активность зарегистрированных мобильных устройств, характеризующую присутствие пользователей и степень их активности в решении задач в СИИО [84; 91]. Устройства пользователей во время использования цифровых сервисов СИИО в беспроводной локальной сети генерируют радиосигналы, которые принимаются и обрабатываются выделенным сетевым RSSI-сенсором, который подключается к той же сети. Сервис на основе полученной информации от выделенного сетевого RSSI-сенсора периодически

вычисляет параметры сетевой активности и уровень присутствия пользователей. Сервис представляется как контекстный, используя разработанную онтологию сервиса для СИИО.

Персональные мобильные устройства принадлежат конкретным пользователям. Таким образом, измеряемая информация о нахождении мобильного устройства в помещении (частный случай пространственно-ограниченной области СИИО) характеризует и присутствие пользователя. В СИИО пользователь работает с одним мобильным устройством для доступа к сервисам. Следовательно, можно задать соответствие между MAC-адресом устройства и пользователем. Рассмотрим три группы сценариев, основанных на использовании такой информации о присутствии пользователей.

Группа S_1. Пользователи приходят на мероприятие. Перед началом основного мероприятия пользователи собираются в помещении СИИО, находясь в состоянии ожидания и подготовки. Обнаружение прибывающих пользователей активирует персонализированные приветственные сервисы и обеспечивает начало основной деятельности. Например, в случае конференции, можно определить всех пользователей, уже готовых к докладам.

Группа S_2. Пользователи приходят в помещение и выходят из него во время мероприятия. Текущий статус каждого пользователя важен для управления повесткой мероприятия или планом деятельности. Например, СИИО может отменить запланированный доклад, если нет докладчика, или перенести выступление на другое время.

Группа S_3. Анализ активности пользователей. В ходе мероприятия накапливается персонализированная информация о пользователях. По итогам формируется отчет, который содержит общий уровень активности и вклад в него каждого пользователя. Например, для анализа итогов мероприятия может быть полезно оценить объем времени, который пользователя провел внутри помещения.

В СИИО каждый пользователь может находиться в различных состояниях присутствия. Такое состояние отражает статус участия в текущей деятельности. На рисунке 3.1 представлена модель состояний присутствия пользователей в СИИО. Каждое состояние определяется совокупностью следующих логических параметров: $s = (R,D,L)$, где R определяет зарегистрирован ли пользователь в системе, D – обнаружено ли в помещении мобильное устройство пользователя сенсором, L – зашел ли пользователь в систему с помощью клиента [85].

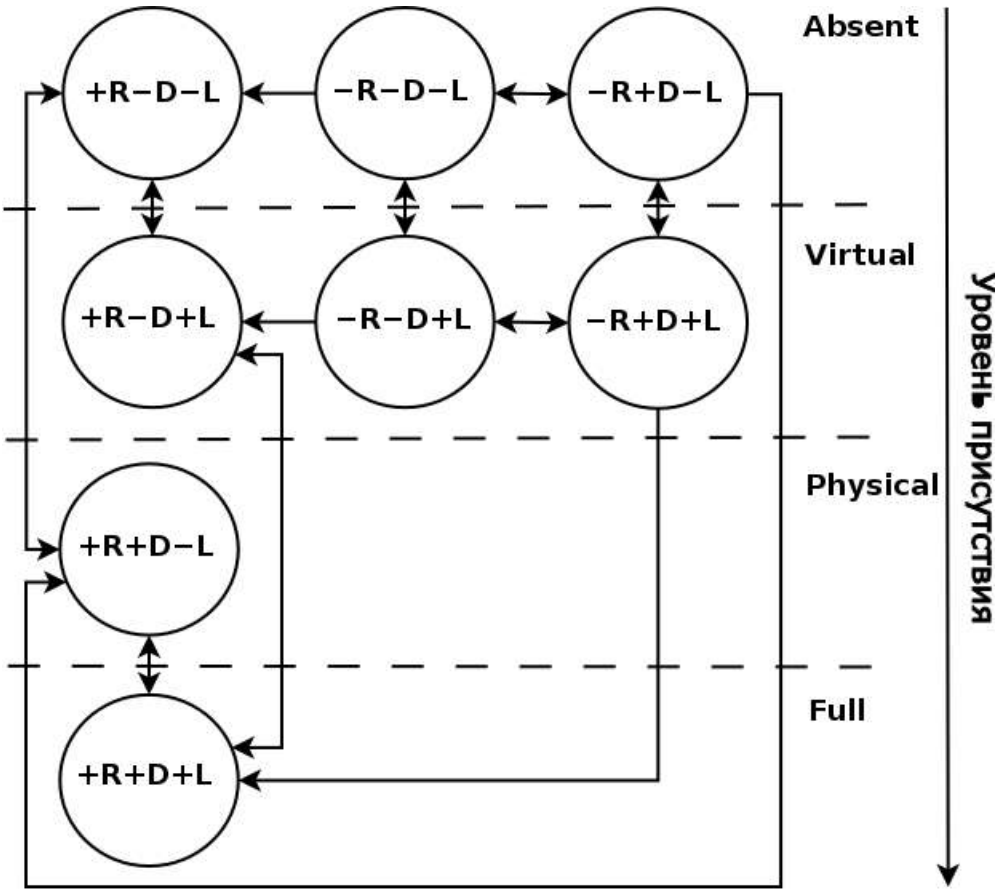


Рисунок 3.1 — Модель состояний присутствия пользователей

В зависимости от состояния присутствия пользователь может быть классифицирован по одному из следующих уровней присутствия: $Absent$ (отсутствие), $Virtual$ (виртуальное присутствие), $Physical$ (физическое присутствие), $Full$ (полное присутствие). Пользователь не может отменить регистрацию в системе. Пользователь может покинуть помещение или выйти из системы, используя клиент. Только зарегистрированные пользователи, предоставившие MAC-адрес своего мобильного устройства, отслеживаются сервисом определения присутствия в пределах области действия сенсора. Например, в сценариях группы S_2 статус пользователя может быть визуализирован на экране сервиса повестки мероприятия в зависимости от текущего уровня присутствия.

Рассмотрим состояния, когда изменяется параметр D, а параметры L и R остаются неизменными. Группы сценариев S_1 и S_2 основаны на обнаружении переходов между данными состояниями. Группа сценариев S_3 не требует мгновенного обнаружения изменения состояния пользователя, однако для этой группы сервис S_{prs} может сохранять состояния присутствия пользователя для дальнейшего анализа.

Группа сценариев S_1 использует однонаправленные переходы между состояниями, т.к. первоначальная идентификация пользователя в помещении происходит один раз перед началом мероприятия:

$$+\mathrm{R}\ -\mathrm{D}\ -\mathrm{L}\ \rightarrow\ +\mathrm{R}\ +\mathrm{D}\ -\mathrm{L}. \tag{3.1}$$

Далее активируется персональный приветственный сервис (на одном из экранов в СИИО), прежде чем пользователь зайдет в систему с помощью клиента. Группа сценариев S_2 использует двунаправленные переходы, которые выполняются после первоначальной идентификации пользователя:

$$+\mathrm{R}\ -\mathrm{D}\ -\mathrm{L}\ \leftrightarrow\ +\mathrm{R}\ +\mathrm{D}\ -\mathrm{L}, \tag{3.2}$$

$$+\mathrm{R}\ -\mathrm{D}\ +\mathrm{L}\ \leftrightarrow\ +\mathrm{R}\ +\mathrm{D}\ +\mathrm{L}. \tag{3.3}$$

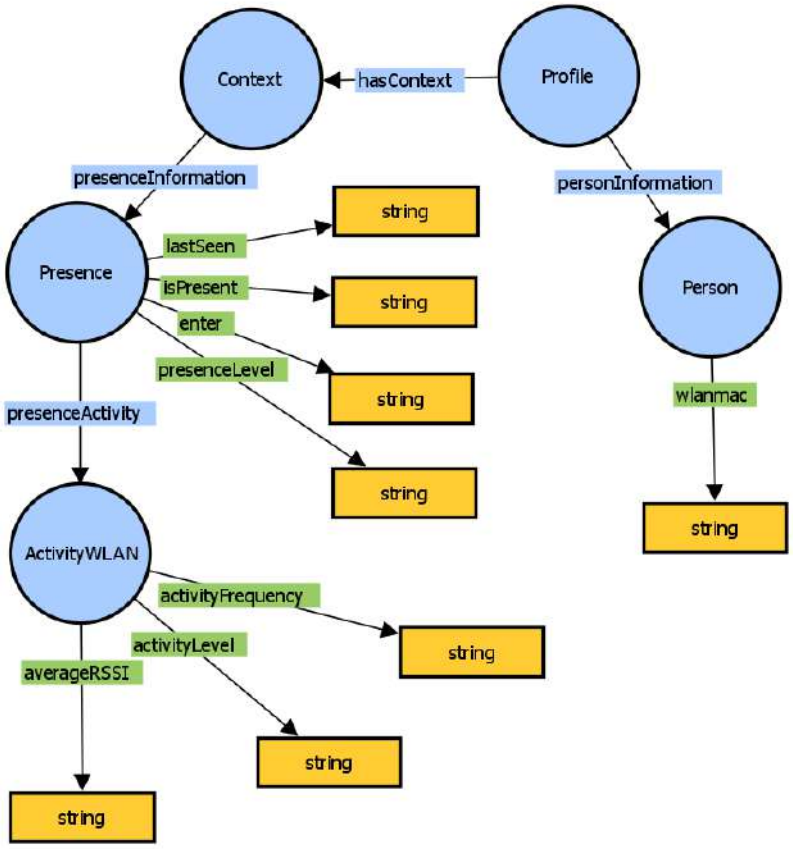


Рисунок 3.2 — Онтология предметной области для сервиса определения присутствия

Определим основные информационные объекты и их связи для сервиса определения присутствия (см. рисунок 3.2). Контекст пользователя определяется как текущее состояние пользователя и используемые им сервисы. Контекст

сервиса определяет, кем и как используется сервис. Рассмотрим онтологическое представление информации о присутствии пользователя, необходимой для создания контекста для сервиса. Сервис S_{prs} связан с профилем участника. Онтология профиля описывает персональную информацию, включая MAC-адрес персонального мобильного устройства и контекст используемых сервисов, с помощью свойства wlanmac класса Person и класса Context соответственно. Когда мобильное устройство пользователя обнаруживается, то сервис находит соответствующий профиль и добавляет информацию о присутствии пользователя: свойства enter (время появления), lastSeen (время последней активности), presenceLevel (уровень присутствия) и isPresent (статус присутствия). Сервис отслеживает и изменяет информацию об уровне присутствия участника, а также периодически обновляет параметры сетевой активности, в том числе уровень активности (activityLevel), степень активности (activityFrequency) и среднее значение RSSI (averageRSSI).

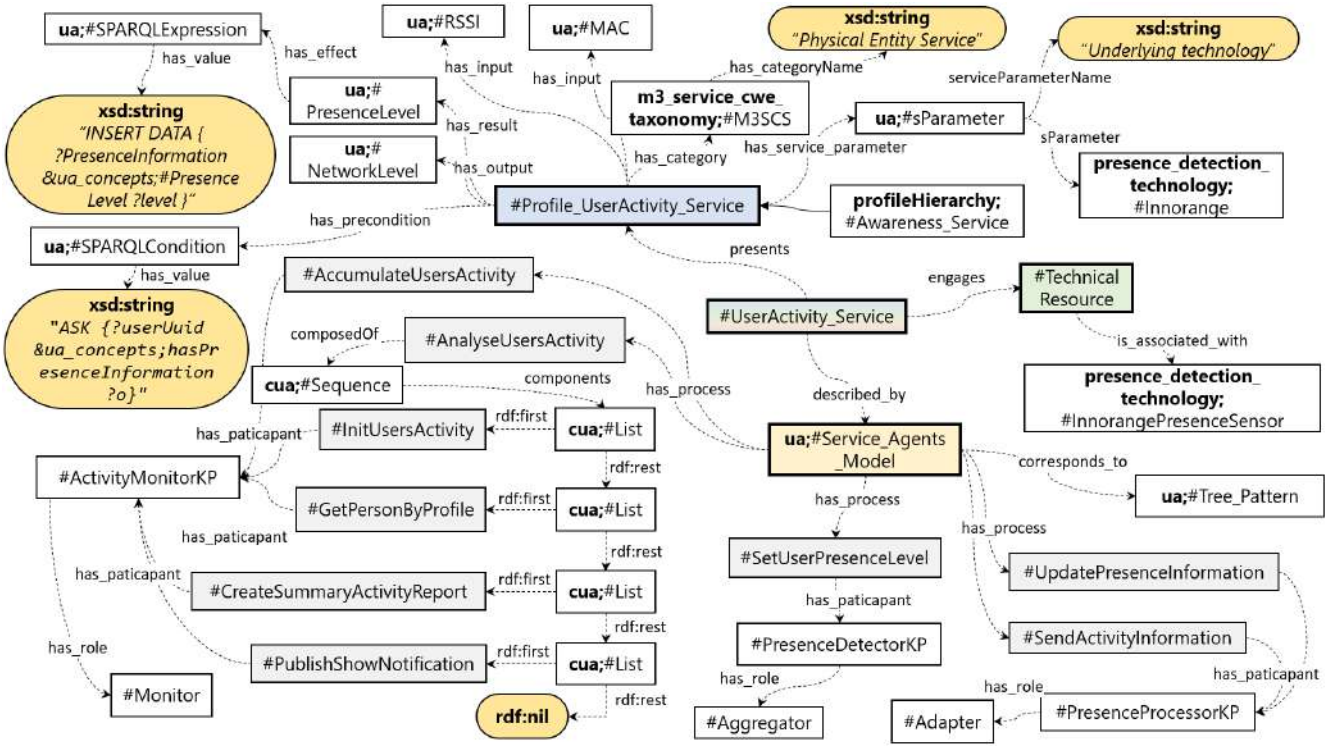


Рисунок 3.3 — Фрагмент онтологии сервиса S_{prs} для описания процесса его построения

Выделенный RSSI-сенсор является основополагающим техническим ресурсом этого сервиса и представляет физическую сущность. Сервис является информационным, предоставляя информацию о состоянии сенсора и его измерениях в ОИС, а также обнаруживая новые знания об активности участников

на основе этой информации. На рисунке 3.3 представлена часть онтологии данного сервиса, включающей набор утверждений об индивидах. Экземпляр класса ServiceProfile (#Profile_UserActivity_Service) предоставляет необходимую информацию о функциональном описании и категориях сервиса. Например, входные параметры определяются следующими индивидами: (а) мощность сигнала (ua;#RSSI), измеренная RSSI-сенсором; (б) зарегистрированные MAC-адреса мобильных устройств (ua;#MAC). Каждый процесс сервиса определяются с помощью необходимых индивидов и их свойств с указанием модели IOPEs. Следующие программные агенты и их процессы участвуют в построении сервиса.

1. Агент-адаптер сенсора (PresenceProcessorKP) взаимодействует с сетевым RSSI-сенсором, который запускается на персональной ЭВМ. Агент реализует функциональность HTTP-сервера и обрабатывает измерения, полученные от RSSI-сенсора в формате JSON. Агент обеспечивает общее информационное содержимое следующими показателями: MAC-адрес устройства, временная метка, значение RSSI. Агент принимает участие в реализации следующих процессов, представленных экземплярами класса AtomicProcess (атомарный процесс): процесс UpdatePresenceInforamtion (обновление информации о присутствии пользователя) и процесс SendActivityInformation (публикация информации об активности в общем информационном содержимом).

2. Агент-агрегатор присутствия (PresenceDetectorKP) следит за обновлениями показателей сетевой активности устройств, реализуя подписку на необходимое ему содержимое. Сопоставление обновленных показателей активности и пользователя происходит благодаря наличию в профиле пользователя информации о MAC-адресе его персонального устройства. В конце мероприятия или по запросу агент на основе накопленных данных вычисляет параметры, необходимые для расширенного анализа сетевой активности: среднее значение RSSI; уровень сетевой активности; степень активности. Агент принимает участие в реализации одного процесса, представленного экземпляром класса AtomicProcess – процесс SetUserPresenceLevel (публикация значений показателей сетевой активности в общем информационном содержимом).

3. Агент-монитор активности (ActivityMonitorKP) анализирует вычисленные показатели сетевой активности пользователей. Агент также может визуализировать активность и доставлять ее другим сервисам СИИО для отображения на общественных экранах. Процесс AnalyseUsersActivity (анализ метрик сетевой активности) представлен как экземпляр класса CompositeProcess (со-

ставной процесс). Этот составной процесс раскладывается на следующие атомарные (простые) процессы: InitUsersActivity (получение с общего информационного содержимого метрик сетевой активности), GetPersonByProfile, CreateSummaryActivityReport (подготовка визуального отчета об активности пользователей в формате PNG) и PublishShowNotification (публикация уведомления в общее информационное содержимое для оповещения других агентов). Их декомпозиция задается с помощью управляющий конструкции «Последовательность» (Sequence), в числе агент участвует в реализации одного независимого атомарного процесса – процесс AccumulateUsersActivity (локальное накопление информации о сетевой активности).

В соответствии с рисунком 3.4 представлена поведенческая модель информационно-управляемого взаимодействия программных агентов, участвующих в выполнении сервиса. Для сценариев S_1 и S_2 переходы между состояниями, представленными в 3.1, 3.2 и 3.3, описываются с помощью поведенческой модели для сервиса S_{prs}. Для описания процесса обмена данными и взаимодействия агентов используются базовые операции протокола SSAP.

3.3 Сервис сопровождения и визуализации плана деятельности людей

Предлагается модель взаимодействия программных агентов для проектирования сервиса сопровождения и визуализации программы совместной деятельности (S_{cws}). Следующие агенты участвуют в построении сервиса: агент управления программой, агент визуализации программы, агент визуализации медиа-информации [18; 25; 26]. Рассматривается онтология сервиса, описывающая его предметную область (см. рисунок 3.5). Процессы совместной деятельности (CollaborativeWork), выполняемые в СИИО, направлены на организацию отдельных мероприятий (Activity), в рамках которых организуется информационный обмен между пользователями и ресурсами. Количество мероприятий ограничивается возможностями вычислительной среды и функциональностью реализованного информационного окружения. Рассмотрим конференцию в качестве примера мероприятия совместной деятельности. Структурными элементами конференции являются секции (Section), которые определяются посредством названия (sectionTitle), даты и времени начала

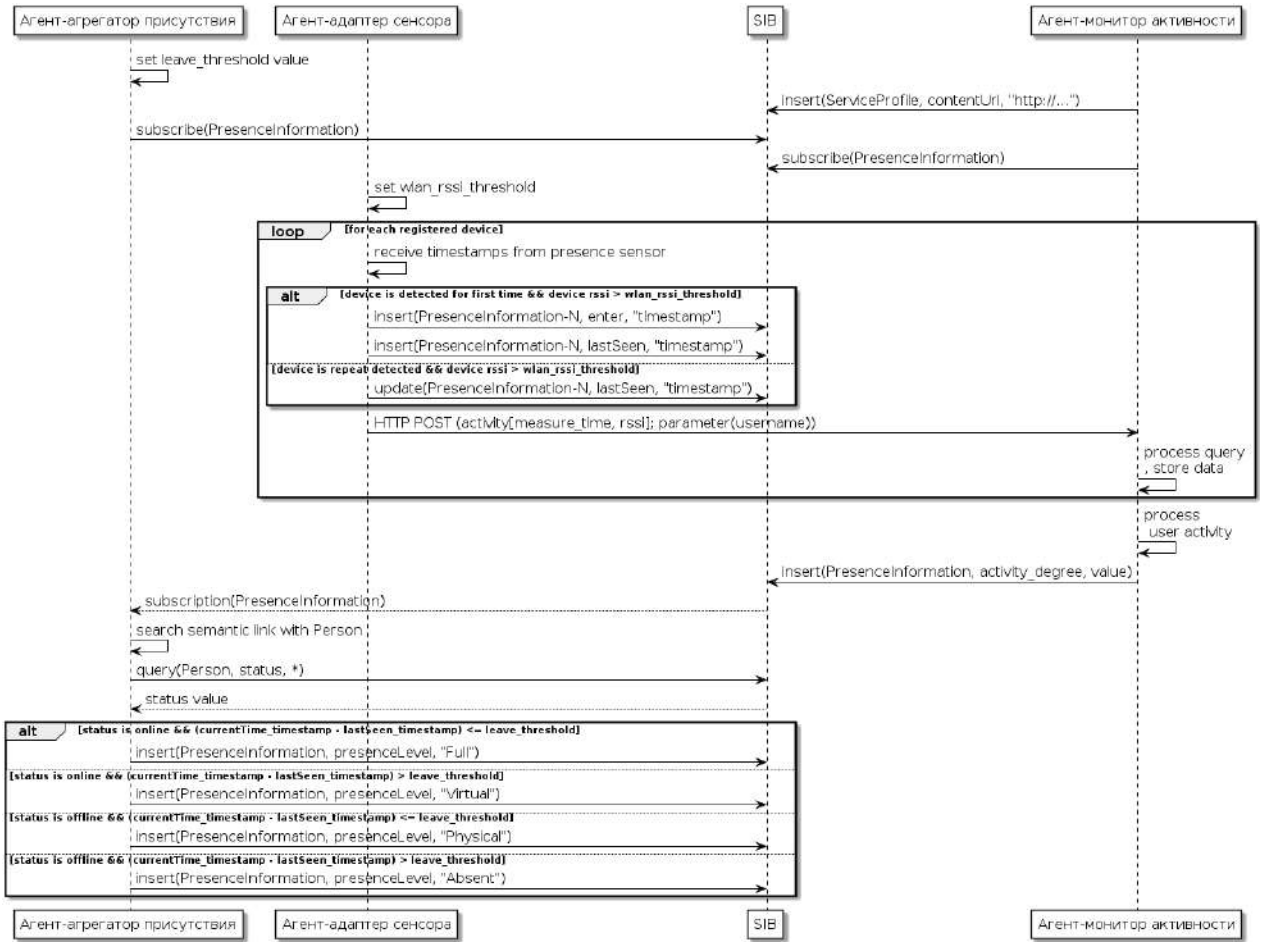


Рисунок 3.4 — Предметно-ориентированная модель проектирования для сервиса S_{prs}: поведенческая модель взаимодействия агентов

(sectionDate), причем время начала может быть запланированным, а может быть фактическим (sectionActualStartTime), в случае, когда конференция началась с опозданием.

Каждая секция состоит из докладов, который представляет собой ограниченный во времени процесс (Timeslot). Такой временной слот связывается с докладчиком и докладом с помощью следующих отношений: имя докладчика (timeslotSpeakerName); профиль докладчика с указанной персональной информацией (timeslotPerson); название доклада или тема выступления (timeslotTitle); профиль презентации выступления (timeslotPresentation); планируемая (timeslotDuration) и фактическая продолжительность выступления (timeslotActualDuration). Такое структурированное представление секции по докладам позволяет программным образом вычислить предварительное временное расписание всего мероприятия-конференции, зная время начала каждой секции и планируемую длительность каждого выступления. Фактическое время начала

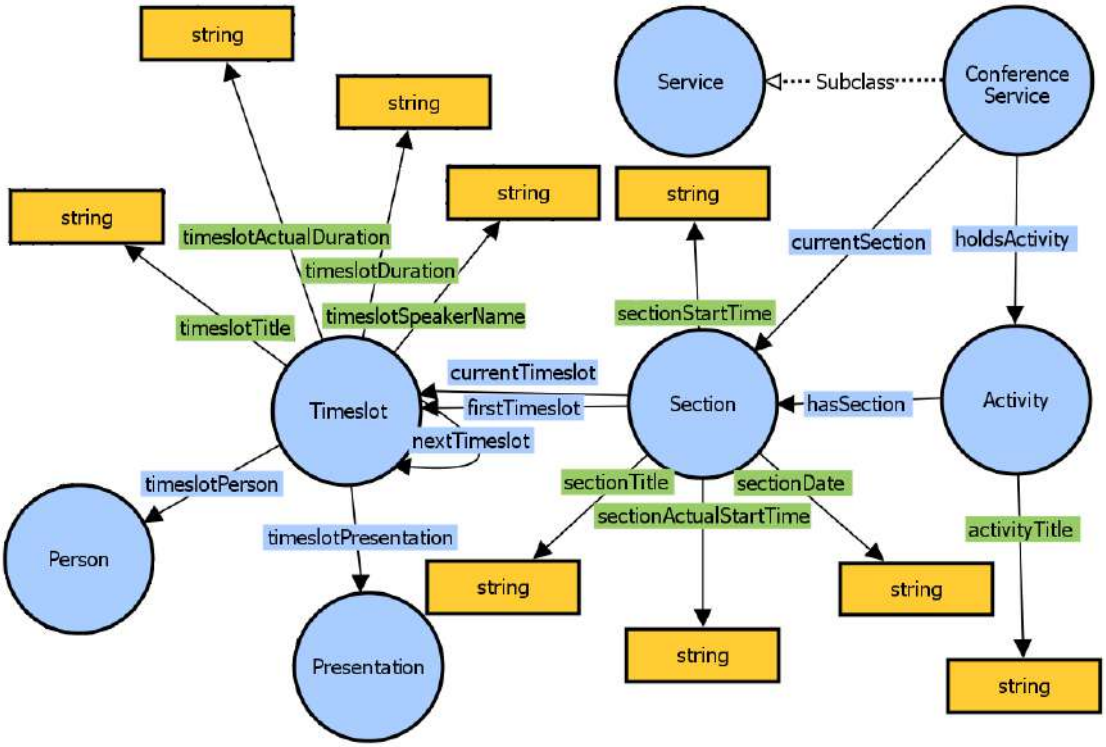


Рисунок 3.5 — Онтология предметной области для сервиса S_{cws}

секций и докладов используется механизмом пересчета времени, реализованным в сервисе S_{cws} для поддержки актуального состояния программы мероприятия.

Онтология позволяет представить последовательность докладов-выступлений в рамках секции в виде связанного линейного списка. Для этого у каждого временного слота (Timeslot) имеется свойство с указанием следующего слота (в случае его наличия). Если значение свойства на следующий доклад или слот отсутствует, то доклад является последним в секции. Для определения первого выступления секция содержит отдельное свойство для указания на первый временной слот. При завершении одного доклада происходит автоматическое переключение на следующий, причем сервис осведомляется о текущем докладе благодаря отдельному свойству в онтологии (currentTimeSlot), значение которого отсутствует, если секция не началась или закончилась.

Для составления расписания выступлений может использоваться дополнительная информация о статусах пользователей (status) на основе предложенной модели состояний пользователей (см. раздел 3.2) в СИИО для сервиса S_{prs}. Используя информацию о статусах пользователей сервис S_{cws} может динамически обновлять программу по ходу мероприятия. Например, если докладчик не успел подойти к началу своего выступления, тогда сервис, анализируя значение статуса его присутствия (например, значение «absent»), автоматически изменяет

повестку мероприятия, перенося данный доклад на конец секции и выполняя пересчет времени начала-конца для каждого доклада в секции, который идет следом за текущим. Порядок докладов также может быть изменен ручным способом, например, организатором или администратором мероприятия. При добавлении нового доклада обязательным условием является наличие информации о профиле (Profile) пользователя с названием его доклада в общем информационном содержимом.

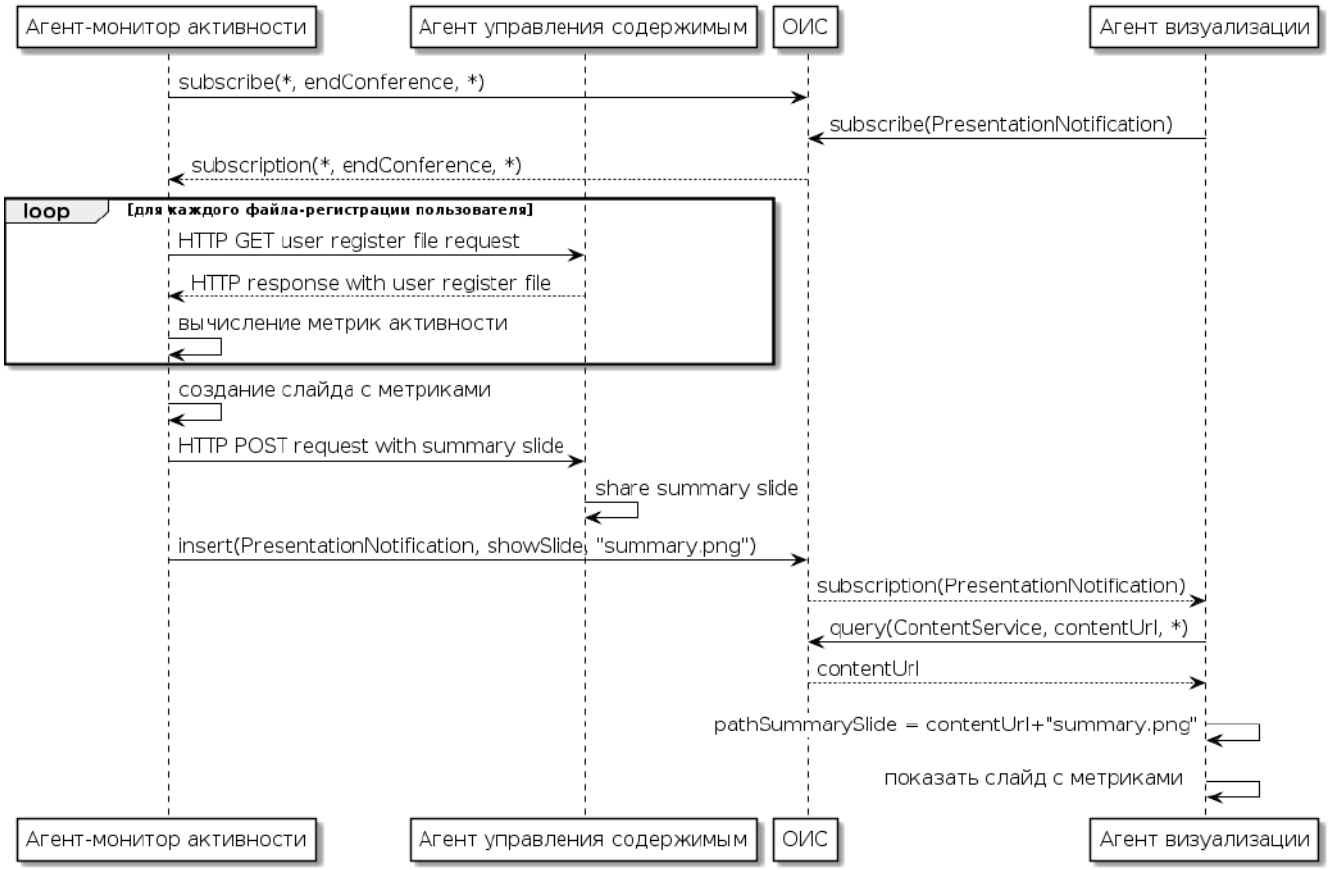


Рисунок 3.6 — Предметно-ориентированная модель проектирования для сервиса S_{cws}: поведенческая модель взаимодействия агентов

Визуализация программы плана деятельности людей происходит с использованием многомодальных пользовательских интерфейсов, адаптированных для широкоформатных общих экранов [89]. Для реализации процесса визуализации используется онтология сервиса S_{cws}. Рассмотрим процесс визуализации программы на примере мультимедийной презентации. Докладчик может использовать файлы электронной презентации для сопровождения своего выступления. Соответствующее онтологическое описание позволяет описать необходимые для этого сведения (название презентации, персональная информация докладчика, ссылка на расположении файла во внешней базе данных). Более того, презен-

тация представляется как последовательный набор слайдов, каждый из которых представлен графическими изображениями с широкоформатным разрешением. Онтология содержит информацию о текущем слайде, которая используется при визуализации соответствующего слайда на общем экране и экранах персональных мобильных устройств. При переключении слайда докладчиком данная информация обновляется в реальном времени. Вспомогательной информацией, отображаемой на экранах, служит информация о порядковом номере отображаемого слайда (currentSlideNum) и об общем числе слайдов в презентации (currentSlideCount).

Вычисленные сервисом S_{prs} значения показателей сетевой активности могут использоваться сервисом S_{cws}, образуя вариант композиции. Информацию о сетевой активности участников эффективно представлять в виде слайда для отображения его на общем экране презентации, представленным агентом визуализации медиа-информации. На рисунке 3.6 представлена предметно-ориентированная модель для сервиса S_{cws}, описывающая поведение агентов. На основе вычисленных показателей формируется сводный слайд для доставки пользователям СИИО. Во взаимодействии участвуют три агента: агент-монитор активности, агент управления содержимым и агент визуализации медиа-информации.

Агент-монитор активности подписывается на свойство, характеризующее конец мероприятия, endConference, и ожидает публикации необходимых RDF-троек в ОИС. Таким образом, в конце мероприятия агент-монитор активности с помощью HTTP GET запросов агенту управления содержимым получает доступ к регистрационным файлам каждого зарегистрированного и предоставившего MAC-адрес своего мобильного устройства пользователя. На основе регистрационных файлов вычисляются необходимые показатели сетевой активности. Далее формируется слайд, содержащий информацию активности пользователей по каждому докладу. Слайд загружается на веб-сервис управления содержимым с помощью HTTP POST запроса с предоставлением к нему внешнего доступа с помощью специализированного URL. Затем агент-монитор активности публикует индивид уведомления с именем файла суммарного слайда для агента визуализации медиа-информации, который подписан на соответствующий класс PresentationNotification. По подписке агент визуализации медиа-информации извлекает из ОИС URL для доступа к веб-сервису управления содержимым и, используя полученное имя файла, формирует URL для доступа к сводному слайду. Полученный слайд отображается на общем экране презентации.

3.4 Сервис совместного пополнения информационного содержимого знаниями о предметной области

Предлагается предметно-ориентированная модель проектирования для сервиса совместного пополнения информационного содержимого (S_{enr}). В качестве предметной области рассматривается окружение умного музея. Сервис обеспечивает пополнение семантической сети информацией из различных источников: (1) интернет-ресурсы исторических данных, (2) индивидуальная информация и исторические знания от посетителей музея, (3) музейная информационная система [43]. В соответствии с рисунком 3.7 представлена архитектурная модель для проектирования сервиса. Сервис важен для музейных сотрудников, поскольку позволяет дополнить описание хранимых экспонатов различной информацией, делая их представление более содержательным. Также посетители приобретают возможность быть не только потребителями информации при изучении экспонатов, но и пополнять информационное описание.

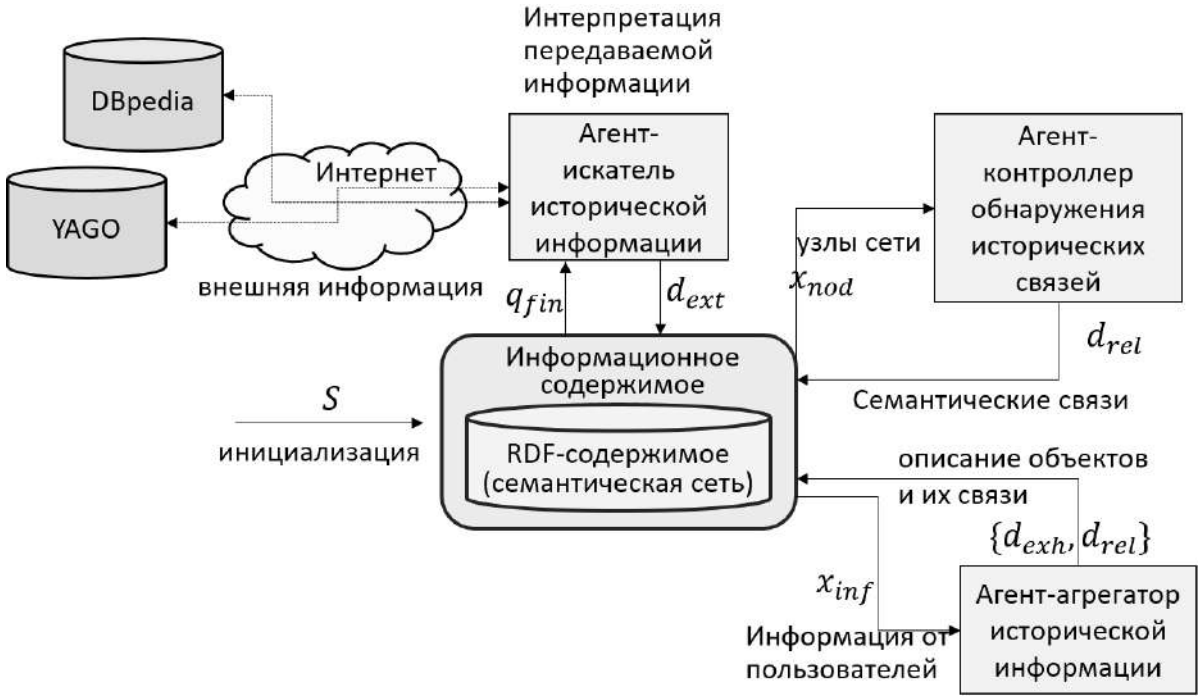


Рисунок 3.7 — Предметно-ориентированная модель проектирования для сервиса S_{enr}: архитектурная модель сервиса

Фрагмент онтологии данного сервиса, включающей набор утверждений об индивидах профиля и модели процесса, представлена на рисунке 3.8. Функциональность сервиса и его интерфейс описываются с помощью индивида #Profile_

Enrichment_Service. Сервис является информационным, предоставляя информацию из различных источников исторических данных. В качестве входного параметра для сервиса указывается URL-адрес для внешней SPARQL-точки доступа или музейной информационной системы (en;#EndpointUrl). Предусловием для выполнения сервиса является наличие информации в ОИС об экспонатах, для которых выполняется процесс пополнения (задается с помощью запроса SPARQL-ASK). Следующие программные агенты и их процессы участвуют в выполнении сервиса.

1. Агент-искатель исторической информации (ExternalFinderKP) отвечает за взаимодействие с внешними интернет-ресурсами (например, сервис DBpedia [58] и музейной информационной системой. Агент обеспечивает процессы совместной деятельности посетителей (организация дискуссии по экспонату, осмотр экспозиции) и объекты музейной экспозиции дополнительной информацией. Агент выступает в качестве семантического посредника, выполняя преобразования информации из внешних источников в RDF-тройки и обратно, формируя семантическую сеть в ОИС. Агент реализует следующие атомарные процессы, направленные на пополнение информации об экспонате: процесс PopulateExhibitInfo (пополнение общей описательной информации), процесс PopulateExhibitRelations (пополнение информации о связях с другими объектами) и процесс PopulateCategories (пополнение информации о категориях экспоната).

2. Агент-агрегатор исторической информации (EnrichmentAggregatorKP) анализирует предоставленную пользователями информацию, чтобы пополнить семантическую сеть более подробными описаниями и отношениями. Пользователи могут добавлять описательную информацию об объектах музейной экспозиции, а также их связи с другими объектами. Один из составных процессов RenewExhibitRelations реализует пополнение связей между экспонатами с помощью сведений, полученных от посетителей. Этот составной процесс раскладывается на следующие атомарные процессы: GetExistExhibitRelations (получение существующих связей между экспонатами), UpdateExhibitRelations (обновление связей экспонатов на основе полученных данных от посетителей и других источников), и publishRenewNotification (отправка уведомления другим агентам через ОИС). Их декомпозиция задается с помощью управляющей конструкции «Последовательность» (Sequence).

3. Агент-контроллер обнаружения исторических связей (SemanticController KP) выполняет связывание виртуальных образов в семантической сети. Агент

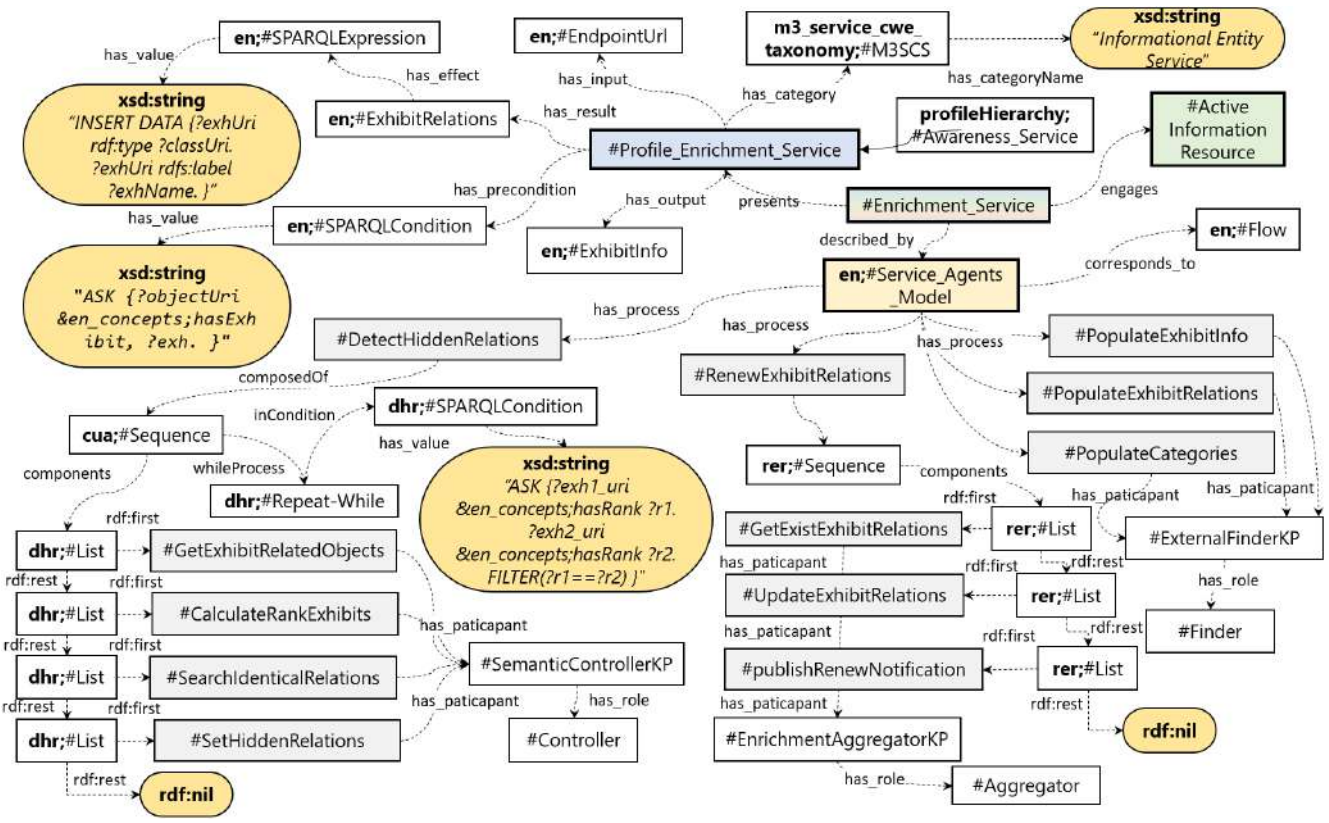


Рисунок 3.8 — Фрагмент онтологии сервиса S_{enr} для описания процесса его построения

извлекает узлы семантической сети, представляющие исторические личности и события, музейные экспонаты и профили пользователей, чтобы впоследствии установить новые семантические отношения между ними, используя реализации алгоритмов рассуждения. Примером такого алгоритма является алгоритм семантического сопоставления похожих отношений. Например, для случая экспоната, алгоритм выполняет поиск объектов в семантической сети, которые имеют похожие с ним связи. На основе ранжирования найденных объектов определяются более близкие экспонаты, что позволяет выявить скрытые связи у целевого экспоната. Алгоритм является итеративным, на каждом шаге которого выполняется пересчет рангов. Составной процесс DetectHiddenRelations, реализующий описанный выше алгоритм, раскладывается на следующие атомарные процессы: GetExhibitRelatedObjects (получить связанные с экспонатом объекты), CalculateRankExhibits (вычислить значение рангов объектов), SearchIdenticalRelations (найти у объектов связи, идентичные с целевым экспонатом), SetHiddenRelations (опубликовать в ОИС найденные скрытые связи). Последовательность выполнения этих процессов задается управляющей конструкцией «Цикл с предусловием», где предусловие задается с помощью запроса

SPARQL-ASK, реализующего условие выхода с цикла (равенство рангов объектов).

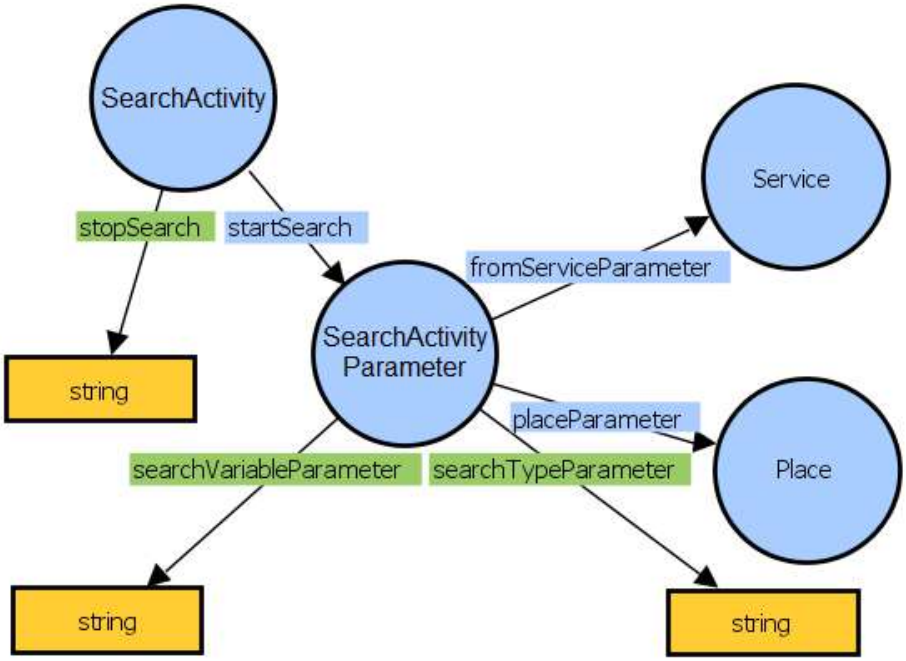


Рисунок 3.9 — Онтология индивида взаимодействия для поиска информации

Для решения проблемы взаимодействия агента-искателя исторической информации и агента-агрегатора исторической информации, используются, так называемые, индивиды взаимодействия. Когда агент-отправитель публикует индивид взаимодействия в общее информационное содержимое, он же подписывается на свойство, характеризующее завершение обработки. Когда агент-получатель завершает соответствующее взаимодействие, он устанавливает статус выполнения индивиду взаимодействия, для которого выполнялись действия. В свою очередь, агент, отправивший запрос на действия, получает уведомление об окончании обработки, проверяет статус выполнения и удаляет соответствующий индивид взаимодействия. При этом уведомления будет получать только один агент-отправитель, а другие агенты, не будут вовлечены, что приводит к увеличению эффективности и уменьшению нагрузки на промежуточное ПО.

На рисунке 3.9 представлен пример индивида взаимодействия для поиска исторической информации во внешнем информационном ресурсе. Индивид используется для взаимодействия между агентом-искателем информации и агентом-агрегатором информации. Индивид содержит параметры поиска (например гео-данные, ключевые слова, тип поиска). Сервис поиска извлекает

параметры и обращается к соответствующим внешним информационным ресурсам (например, DBpedia).

3.5 Сервис мониторинга объектов физической среды

Мониторинг состояния физической среды и контролирование ее параметров является важной проблемой в СИИО. Так, например, такие значения физических параметров как температура, влажность, уровень концентрации углекислого газа, освещенность определяют комфортную, продуктивную и безопасную совместную деятельность людей.

Мониторинг состояния среды совместной деятельности и ее параметров может быть организован посредством использования WSN. Среда совместной деятельности на основе интеллектуальных пространств следует архитектуре SOA. Интеграция и применение возможностей WSN в вычислительных средах совместной деятельности происходит за счет создания необходимых сервисов.

Первый возможный подход интеграции и применения основан на сервисах, которые используют физические устройства/узлы WSN как основополагающие сущности. Одна группа таких сервисов получает, преобразовывает информацию от сенсорных узлов и выполняет ее анализ, агрегацию для формирования контекстной информации для построения другой группы сервисов. Другая группа сервисов оказывает управляющие воздействия с помощью узлов исполнительных устройств, контролируя различные параметры среды, измеряемые сенсорными узлами. Так на каждых узлах запускается агент-адаптер, который реализует виртуальный образ узла, обеспечивая информационное содержимое информацией от него и наоборот. Другие агенты запускаются на устройствах вне WSN, выполняя семантический анализ накопленного содержимого и динамическое обновление семантических связей. Однако такой подход, реализуемый с помощью платформы Smart-M3, имеет ряд ограничений, который в большинстве случаев делает его невозможным.

Производительности устройств WSN недостаточно для запуска агента. Исполняемый файл агента, разработанного с помощью библиотеки C_KPI, используемой специально для малопроизводительных устройств, занимает около 20 Кбайт, тогда как размер флеш-памяти устройства составляет около 48 Кбайт (на-

пример, SADmote, Tmote Sky), что составляет значительную часть. Также агенты могут сильно нагружать центральный процессор микроконтроллера (MSU), значительно увеличивая потребление энергии.

Невозможность компиляции сторонних библиотек, используемых агентами. В большинстве случаев программное обеспечение для узлов WSN использует простые и ясные абстракции языка C или UNIX-абстракции (MansOS, TinyOS) без возможности использовать такие необходимые сторонние библиотеки для C_KPI, как scew.

Взаимодействие агентов с помощью информационного содержимого происходит на основе протокола SSAP, который выполняется поверх протокола TCP/IP. Большинство узлов WSN используют протоколы, основанные на стандарте IEEE 802.15.4, поэтому использование протокола TCP/IP является нецелесообразным (а в некоторых случаях и невозможным) из-за ограничений формата пакета 802.15.4 (например, можно передавать довольно короткое сообщение – 256 байт).

Общее информационное содержимое обеспечивает семантику происходящих процессов и не предназначено для хранения активно обновляющихся данных сенсоров, а тем более для хранения истории значений параметров. Таким образом, сервер с локальной базой данных с информацией, полученной от сенсоров, является необходимой информационной сущностью.

При втором подходе создается отдельная инфраструктура, отличная от архитектуры M3, которая использует специализированные технологии для создания WSN. Основным компонентом в такой инфраструктуре выступает сервер, который содержит базу данных для хранения информации о сенсорах и их измеряемых параметрах среды, программную часть для работы с базой данных и для взаимодействия с узлами WSN, а также веб-интерфейс для сторонних пользователей и других программ. В этом подходе интеграция и применение WSN в средах совместной деятельности основана на сервисе, который использует программную часть сервера как основополагающую информационную сущность. Второй подход не имеет таких ограничений как первый.

Проектное решение для обеспечения интеграции и применения сенсорной сети в средах совместной деятельности, основанное на втором подходе, представлено на рисунке 3.10. Сервис мониторинга объектов физической среды (S_{mnt}) состоит из следующих агентов. Агент-интерпретатор базы данных сенсорной сети обеспечивает информационное содержимое информацией об измерениях, полученной с сервера. Взаимодействие с сервером происходит через

веб-интерфейс. Агент-контроллер узлов сенсорной сети выполняет динамическое обновление семантических образов WSN-узлов и связей между ними. Агент-агрегатор измерений сенсорной сети выполняет анализ семантических образов, извлекает знания из имеющейся информации, а также отправляет управляющие воздействия при необходимости.

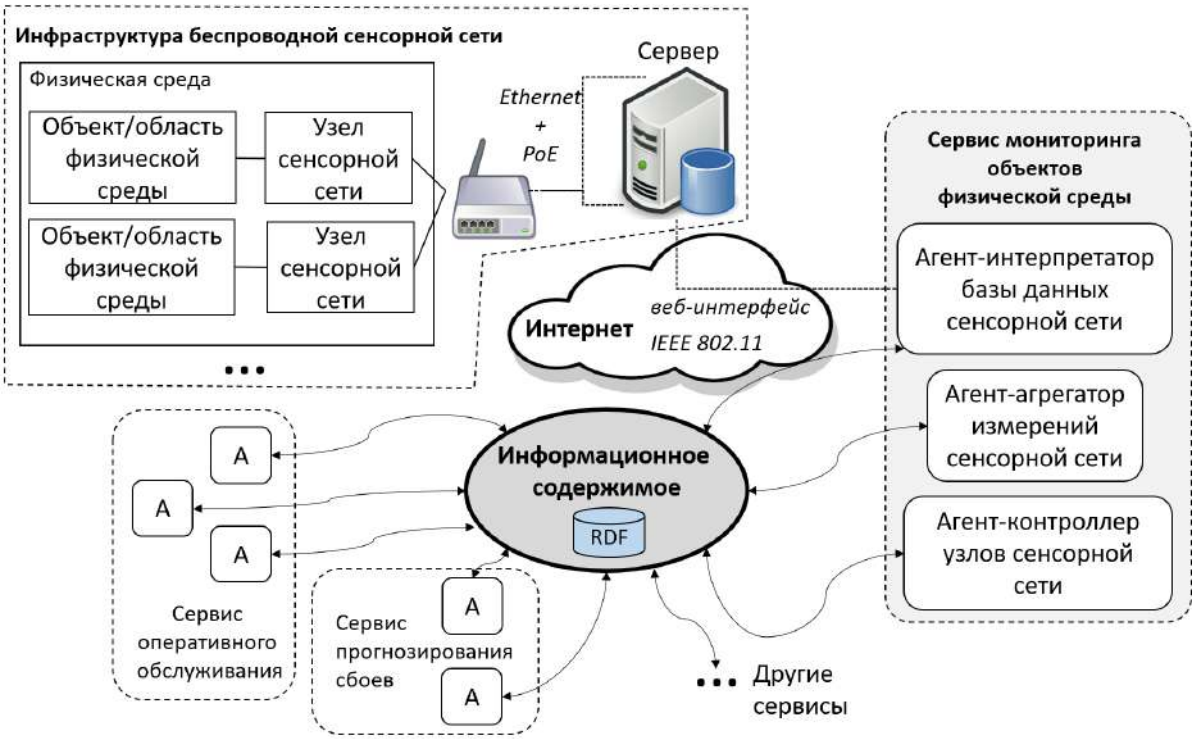


Рисунок 3.10 — Предметно-ориентированная модель проектирования для сервиса мониторинга объектов физической среды: архитектурная модель сервиса

Сервис обеспечивает автоматизацию обработки данных и информационное сопровождение сотрудников. Рассматриваются следующие классы задач многопараметрического мониторинга.

1. Непрерывная диагностика технического состояния и условий эксплуатации оборудования предприятия на основе технологий промышленного интернета и с привязкой к действиям сотрудников.
2. Предупреждение сбоев оборудования в оперативном режиме на основе семантических методов интеллектуального анализа данных мониторинга с предиктивным анализом истории эксплуатации оборудования и с учетом контекста текущей ситуации.
3. Интеллектуальное управление обслуживанием оборудования на основе мобильного информационного сопровождения сотрудника с построением рекомендаций по выполнению действий.

Получение информации от датчиков выполняется по специальным протоколам напрямую или через устройства сбора данных. Каждый датчик взаимодействует с отдельным программным агентом для выполнения первичной обработки данных оперативного мониторинга (кодирование видеопотока, вычисление частотных характеристик, вычисление сводных характеристик и др.). Результаты первичной обработки помещаются в общее информационное содержимое или передаются в виде информационных потоков. Множество программных агентов оперативного мониторинга образуют слой цифровых измерений.

Связывание результатов первичной обработки и формирование общего информационного содержимого выполняется отдельными программными агентами семантического связывания. Каждый агент выполняет анализ результатов первичной обработки одного или нескольких датчиков в пространственном и временном срезах. Результаты работы агентов семантического связывания помещаются в общее информационное содержимое для последующего анализа и представления.

Поиск, анализ и представление данных осуществляется с помощью соответствующих программных агентов. Каждый агент реализует механизмы, обеспечивающие доступ к одной или многим возможностям сервиса S_{mnt} для решения возникающих у пользователя задач. Сервисы обеспечивают доступ с помощью различных устройств ввода/вывода (рабочее место оператора, информационное табло, мобильное устройство, устройство оповещения и др.). Также в класс сервисов включены программные решения для интеллектуального анализа данных и прогнозирования. Эти задачи используют специальные математические алгоритмы и модели обработки построенной семантической сети.

Для обеспечения взаимодействия между программными агентами используется событийная модель на основе информационно-управляемого взаимодействия. Каждое явление, необходимое для дальнейшего анализа и интерпретации и достаточное для сохранения в файловой системе и/или базе данных общего информационного содержимого, представляется в виде событий. Агенты оперативного мониторинга создают простые события, не требующие дополнительных объяснений на уровне рассматриваемой архитектуры (например, появление требуемой частоты в спектре, изменение значения или появление объекта в кадре). Агенты семантического связывания и интеллектуального анализа создают составные события, включающие в себя цепочку из нескольких последовательных событий.

3.6 Выводы

В главе представлены предметно-ориентированные модели проектирования сервисов СИИО для выполнения прикладной разработки для востребованных приложений СИИО: распознавание присутствия и анализ активности пользователей; сопровождение и визуализация плана деятельности людей; пополнение информационного содержимого знаниями о предметной области; мониторинг объектов физической среды. Полученные архитектурные и поведенческие абстракции для каждого сервиса, представленные в виде шаблонных решений, позволяют снизить трудозатраты на проектирование сервисов при использовании предложенного метода разработки программной инфраструктуры. Для каждого рассмотренного сервиса определен набор программных агентов участвующих в его построении, составлена модель информационно-управляемого взаимодействия агентов, приведены основные операции агентов и потоки данных, разработана частная онтология на основе предложенной концептуальной модели сервиса.

На основе предметно-ориентированных моделей в разделе 4.2 следующей главы описывается реализация экспериментальных образцов сервисов для вычислительных сред интеллектуального зала, умного музея и промышленного предприятия, а также исследуется их функциональность с целью оценки качества сгенерированного программного кода.

Глава 4. Комплекс программных средств

В главе описывается программная реализация комплекса программных средств: а) реализация генератора программного кода взаимодействия агентов на основе полученного алгоритма автоматизации программирования и концептуальной модели сервиса; б) экспериментальные образцы предметно-ориентированных сервисов на основе метода разработки интероперабельной программной инфраструктуры для организации СИИО для вычислительных сред интеллектуального зала, умного музея и промышленного предприятия. На основе реализованного комплекса программных средств выполняются экспериментальные исследования с целью оценки эффективности метода разработки программной инфраструктуры, который позволяет снизить трудозатраты на создание сервисов как систем взаимодействующих агентов за счет использования унифицированной онтологии сервиса и генератора программного кода взаимодействия агентов.

4.1 Реализация генератора программного кода взаимодействия агентов

Генерация программного кода взаимодействия агентов основана на предложенном алгоритме автоматизации программирования взаимодействия агентов (см. раздел 2.3). Для реализации интероперабельных структур модели данных и поведения агентов при программировании сервиса получена реализация генератора программного кода с использованием языка Java.

Разработчик агентов предоставляет на вход генератору спецификацию проблемной области как описание OWL и спецификацию сервисов как описание, основанное на OWL-S. Генератор использует статическую схему шаблонов и обработчиков. Шаблоны кода представляют собой «предварительный код» для классов, объектов, атрибутов и функций, которые реализуют сущности и свойства онтологий OWL, а также сервиса, основанного на OWL-S. Обработчики могут преобразовывать один или несколько шаблонов в целевой программный код, заменяя специальные теги именами и элементами, взятыми из онтологий. Код может быть сгенерирован для нескольких агентов в зависимости от того, какие агенты

указаны в онтологическом описании процессов выполнения сервиса. Преобразование происходит во время обхода RDF-графа онтологии с использованием фреймворка Jena [42]. С его помощью строится метамодель для представления графа. На основе информации, полученной во время обхода генератором узлов метамодели (вершин онтологического графа), обработчики преобразуют шаблоны в целевой программный код.

Для построения метамодели онтологии в качестве семантического механизма рассуждения используется механизм Pellet [95] с открытым исходным кодом. Pellet реализуется с помощью одноименной библиотеки, созданной на языке Java. В соответствии с листингом 4.1 создается метамодель онтологии, которая в дальнейшем используется функциями Jena API в обработчике OntHandler. Использование механизма рассуждения Pellet позволяет получить более полную метамодель онтологии, содержащую информацию об отдельных индивидах (предоставляя возможность логического вывода над компонентами утверждений ABox), а также описывающую типы данных из спецификации XML Schema, что позволяет при обработке метамодели использовать все возможности дескрипционной логики OWL-DL.

```
1  OntModel ontm = ModelFactory.createOntologyModel(
2                                  PelletReasonerFactory.THE_SPEC);
3  for (String file : inputFiles) {
4    InputStream in = FileManager.get().open(file);
5    ontm.read(in, "");
6  }
7  CCodeModelContainer ccm = new CCodeModelContainer();
8  if (AppConfig.getHandlerType() == AppConfig.HandlerType.C)
9  {
10   OntHandler.handle(ontm, ccm);
11 }
```

Листинг 4.1 — Создание метамодели онтологии и вызов обработчика

Процесс генерации программного кода основывается на методах трансформации [98]. Специальные теги представляют собой высокоуровневые объекты, которые цепочкой преобразований переходят в низкоуровневые конструкции — директивы языка программирования. Для выполнения процесса преобразования выполняются три шага: (1) поиск соответствующего шаблона программного кода, (2) проверка логических соглашений на возможность преобразования, (3) вызов процедуры преобразования. Проверку на соответствие набору логических соглашений и процедуру преобразования реализуют обработчики. Процедуры преобразований реализуют как горизонтальную трансформацию (например, заменяя теги в шаблонах на имена функций, переменных, объектов), так и вертикальную

трансформацию (заменяя специальные конструкции на условные операторы, циклы, реализуя логику и алгоритмы обхода, запросов, вызовов и т.д.). Два данных вида трансформации чередуются в длинных последовательностях преобразований, обходя узлы RDF-графа онтологии, реализующие различные концепты (процессы, входные данные, управляющие конструкции).

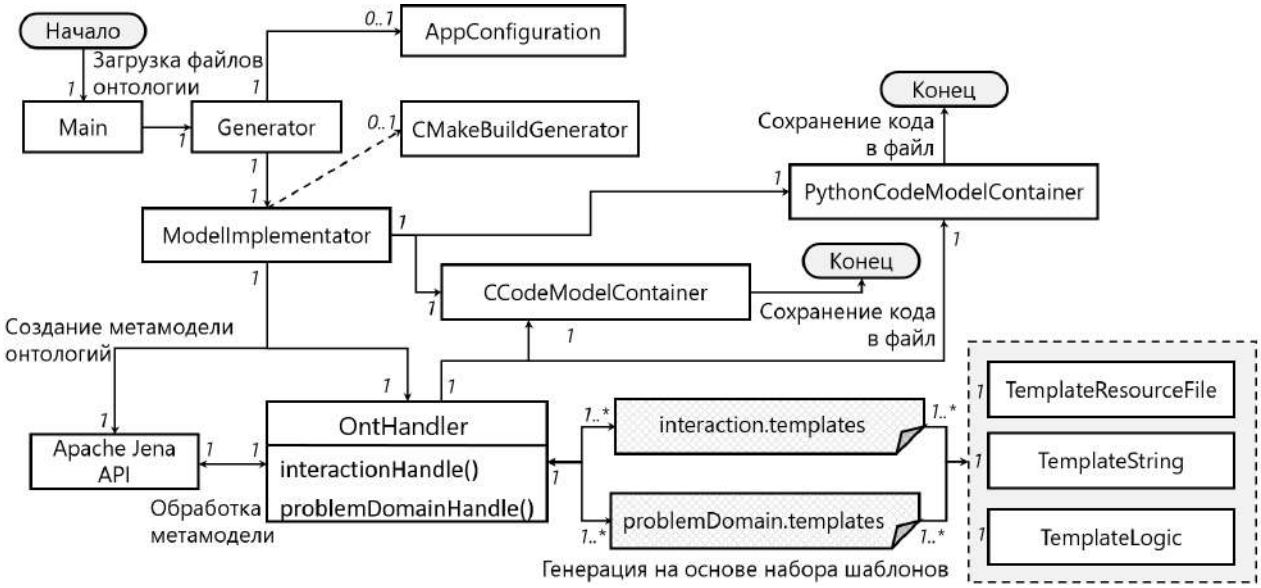


Рисунок 4.1 — Архитектура генератора программного кода взаимодействия агентов

Высокоуровневая архитектура генератора программного кода взаимодействия агентов представлена на рисунке 4.1. Каждый модуль генератора реализуется как Java-класс. Основным модулем, осуществляющим процесс генерации кода, является обработчик OntHandler. Данный разработчик реализует два основных метода: (1) *interactionHandle()* – обработка концептов онтологии, отвечающих на взаимодействие; (2) *problemDomainHandle()* – обработка онтологических концептов, представляющих объекты предметной области и их свойства. Модуль OntHandler взаимодействует с внешним модулем (библиотекой) Apache Jena API. Взаимодействие происходит итеративным образом: сначала загружаются файлы шаблонов (.templates), затем выполняется последовательность действий над метамоделью онтологии для получения из нее необходимых элементов.

В таблице 6 представлен список выполняемых действий для обхода узлов метамодели онтологии в методе *problemDomainHandle()* и функции Jena API для их реализации. Задача данного метода: извлечь всевозможные концепты онтологии, необходимые для генерации объектной модели данных. Основными извлекаемыми концептами являются классы, индивиды, свойства и ограничения.

Для прохождения смежных узлов (для свойств или классов) модели фреймворк Jena API имеет соответствующие варианты функций и их параметров. В некоторых случаях онтология представляет наиболее богатую информационную модель, чем объектно-ориентированная модель языка программирования (например, поддержка множественного наследования классов). В таких случаях соответствующий метод обработчика преобразует найденные концепты в их ближайший аналог на основе определенных в методе правил. Например, использование одиночного наследования наряду с множественным наследованием интерфейсов классов.

Таблица 6 — Действия для обхода узлов метамодели в методе *problemDomainHandle()*

Действия для обхода метамодели	**Функции Jena API**
Извлечь список именованных классов	listNamedClasses()
Извлечь список «смежных» родительских классов	listSuperClasses(true)
Извлечь список индивидов для класса	listIndividuals()
Извлечь список объявленных «смежных» свойств для класса	listDeclaredProperties(true)
Извлечь список «смежных» родительских свойств для заданного свойства	listSuperProperties(true)
Определить тип свойства: объектное или свойства данных	isObjectProperty() isDatatypeProperty()
Получить список ограничений для свойства	listRestrictions()
Определить тип ограничения: ограничение кардинальности, ограничение значения	isCardinalityRestriction() isHasValueRestriction()

Метод *iteractionHandle()* обработчика OntHandler обрабатывает концепты онтологии сервиса (предлагаемой в данном исследовании). Основным элементом взаимодействия выступают функции программных агентов, логика которых направлена на выполнение операций (вставка, обновление, удаление, подписка) для работы с RDF-тройками из ОИС. Для обработки и извлечения информации о процессах выполнения сервисов и взаимодействия агентов из метамодели онтологии используется сочетание функций Jena API для работы с онтологическими утверждениями (ABox): индивиды и значения их свойств с ограничениями. В соответствие с листингом 4.2 представлен фрагмент кода, реализующий обход метамодели онтологии с целью извлечения информации о входных параметрах для простых процессов выполнения сервисов. Данный фрагмент кода используется в методе *interactionHandle()*. Простые процессы представлены индивидами, при-

надлежащими к онтологическому классу с именем AtomicProcess. Обходя узлы, представляющие данные индивиды, извлекаются их локальные имена в модели, а также входные параметры, заданные списком свойств с именем hasInput. Свойства и их типы извлекаются в виде набора утверждений (RDF-узлов или RDF-троек), которые раскладываются на три компонента: субъект, предикат, объект. После получения необходимой информации вызываются методы замены тегов для имени функции и входных параметров в шаблоне для простого процесса APrDeclarTempName.

```
1  Template APrDeclar = new TemplateString(APrDeclarTempName);
2
3  Property hasInput = ontModel.getProperty(PROCESS_NS, "hasInput");
4  Property parameterType = ontModel.getProperty(PROCESS_NS,
5                                                "parameterType");
6
7  ExtendedIterator itrInd = ontModel.listIndividuals();
8  while (itrInd.hasNext()) {
9    Individual indAP = (Individual) itrInd.next();
10   OntClass ocIndAP = ind.getOntClass(true);
11   if (ocIndAP.getLocalName() != null &&
12                   ocIndAP.getLocalName().equals("AtomicProcess") {
13     APrDeclarName.repalceNameTag("<@processname@>", indAP.getLocalName
       ());
14     StmtIterator itrHasInput = ind.listProperties(hasInput);
15     while (itrHasInput.hasNext()) {
16       Statement stHasInput = itrHasInput.next();
17       RDFNode nd = stHasInput.getObject();
18       String inputName = nd.asLiteral..getString();
19       Statement stType = nd.asResource().getProperty(parameterType);
20       String stTypeValue = stType.getObject().asLiteral().getString();
21       APrDeclarInput.repalceInputTag("<@inputname@>",
22                              "<@inputtype@>", inputName, stTypeValue);
23     }
24   }
25 }
```

Листинг 4.2 — Извлечение имен и типов входных параметров для простых процессов

Составные процессы реализуются как функции взаимодействия программных агентов, которые внутри своей логики вызывают другие функции, которые соответствуют сгенерированным по простым процессам функциям. Их вызов определяется управляющей конструкцией (Sequence, Repeat-While, и т.д.). В соответствии с листингом 4.3 представлен фрагмент кода, реализующий обход узлов составных процессов в метамодели. Каждый составной процесс представляет собой древовидную структуру узлов метамодели, содержащую информацию о вложенных простых процессах, типе управляющей конструкции, запрос-условие (при необходимости). Последовательное извлечение узлов и их

связей с помощью функций для извлечения свойств (*getProperty()*) и утверждений (*getObject()*) позволяет обойти «дерево» составного процесса. Каждому типу управляющей конструкции соответствует свой шаблон (например, для цикла while – CPrWhileTemplate), который принимает на вход имена простых процессов и тело запроса на языке SRAQL для записи предусловия. Запрос SPARQL представляет собой RDF-литерал (символьная константа).

```
Resource cr = OnontModel.getOntClass(PROCESS_NS +
                                     "CompositeProcess").asResource();

ExtendedIterator itrIndCP = itrInd ontModel.listIndividuals(cr);
while (itrInd.hasNext()) {
  Individual indCP = (Individual) itrIndCP.next();
  Statement stComposedOf = indCP.getProperty(composedOf);
  Resource cf = stComposedOf.getObject().asResource();
  OntClass ctrConst = ontModel.getOntClass(cf.getURI());
  if (ctrConst.getSubClass().getLocalName().equals("ControlConstruct")
   ) {
    String constructName = cf.getLocalName();
      if (constructName.equals("RepeatWhile") {
        Statement stComposedOf = cf.getProperty(whileCondition);
        Statement stExp = stComposedOf.getProperty(expressionBody);
        String expQuery =  stExp.getObject().asLiteral().toString();
        StmtIterator itrComp = cf.listProperties(components)
        while (itrComp.hasNext()) {
          Statement stPrComp = itrComp.next();
          String apName = stPrComp.getObject().getLocalName()'
          apProcesses.add(apName);
        }
        CPrWhile.replaceConstructTag("<@contrconst@>","<@process*@>",
                    <@query@>, constructName, apProcesses, expQuery);
      }
  }
}
```

Листинг 4.3 — Извлечение имен простых процессов и их композиции в составных процессах

Загрузка файлов шаблонов (.templates) для логики взаимодействия и объектной модели данных предметной области во внутренние строки происходит с помощью модуля TemplateResourceFile. Обращение к модулю TemplateString позволяет генерировать код на основе горизонтальной трансформации (замена тегов на различные имена, представленные строками) по загруженному шаблону для объектной модели данных. Модуль TemplateLogic реализует вертикальную трансформацию, генерируя код для логики взаимодействия с помощью использования операторов if-then-else, while и функции API промежуточного программного обеспечения для организации ОИС. Для каждого поддерживаемого языка (на данный момент, языки C++ и Python) реализованы специальные модули, которые учи-

тывают синтаксические соглашения соответствующего языка для сохранения сгенерированного кода во внутренние строки файлов. Генератор поддерживает масштабируемость с точки зрения добавления нового языка. Необходимо добавить набор шаблонов для соответствующего языка, после этого проверить функциональность обработчиков. Как правило, в большинстве случаев обработчики являются универсальными.

Генератор также реализует возможность создавать необходимые сопровождающие файлы для сборки программных агентов на языке C++ с использованием системы сборки CMake. Пример части сгенерированного кода представлен в соответствии с листингом 4.4. Данную возможность реализует модуль CMakeBuildGenerator, создавая набор конфигурационных файлов CMakeList, в которых прописываются пути для связывания отдельных директорий для каждого агента, участвующего в выполнении сервиса. Указываются пути до необходимых библиотек для сборки в зависимости от целевой операционной системы, различные флаги сборки.

```
1  set_property(GLOBAL PROPERTY LANGUAGE CXX)
2  set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} —Wall —std=c++0x")
3
4  set(SEMANTIC_CONTROLLER_KP "semantic—controller_kp")
5
6  if(NOT DEFINED CMAKE_INSTALL_LIBDIR)
7    set(CMAKE_INSTALL_LIBDIR "\${prefix}/lib")
8  endif(NOT DEFINED CMAKE_INSTALL_LIBDIR)
9
10 if(NOT DEFINED INCLUDE_INSTALL_DIR)
11   set(INCLUDE_INSTALL_DIR "\${prefix}/include")
12 endif(NOT DEFINED INCLUDE_INSTALL_DIR)
```

Листинг 4.4 — Сгенерированный блок кода для системы сборки CMake

В текущей программной реализации генератора для языка C++ используется 29 шаблонов, для языка Python – 19 шаблонов. Генерация основной части кода возможна благодаря сходству понятий в онтологии с элементами объектной модели данных, используемой в объектно-ориентированных и динамически типизированных интерпретируемых языках программирования [32]. Типы шаблонов с их описанием и примерами представлены в соответствии с таблицей 7. На основе представленных типов можно выделить следующие классы шаблонов: объявление, реализация, сборка, отладка. Шаблоны для объявления и сборки не используются для языка Python, т.е. для динамически типизированных интерпретируемых языков программирования. Такие шаблоны необходимы только для компилируемых статически типизированных языков

программирования (в текущей реализации генератора только поддержка языка C/C++). Некоторые шаблоны являются составными, т.е. они включают в себя другие шаблоны. Например, шаблон для реализации функции простого процесса (*AtomicFuncionDeclaration.template*) включает в себя другие шаблоны, которые реализуют предусловия и эффекты выполнения функции. Такая вложенность организуется специальными тегами – <% %>.

Таблица 7 — Типы шаблонов для генерации программного кода

Типы шаблонов	**Описание**
Объявление структур объектной модели	Объявление классов, их конструкторов, атрибутов и специальных методов в заголовочных файлах кода (при использовании языком препроцессора). Пример: *DomainClassDeclaration.template*
Реализация структур объектной модели	Реализация классов, их конструкторов, атрибутов и специальных методов в основном коде агентов. Создание объектов классов в используемых функциях взаимодействия. Пример: *DomainClassImplementaion.template*
Реализация ограничений атрибутов объектной модели	Реализация ограничений в конструкторах и специальных методов классов с помощью условных операторов для проверки кардинальности, максимальных и минимальных значений атрибутов. Пример: *AttributeRestrictionImplementaion.template*
Объявление функций взаимодействия	Создание заголовочных файлов и объявление прототипов функций (при использовании языком препроцессора) с входными и выходными параметрами. Пример: *AtomicFuncionDeclaration.template*
Реализация функций взаимодействия	Создание структуры (возвращаемое значение, имя функции, входные параметры) реализации функций взаимодействия агентов как для простых процессов, так и для составных. Для тела функции на основе структур объектной модели данных создаются необходимые локальные переменные, а также включаются другие шаблоны. Пример: *AtomicFuncionImplementaion.template*
Реализация предусловия функции	Создание SPARQL-запроса (ASK) с телом из онтологии, вызов встроенных функций API для промежуточного ПО, создание условных операторов для ветвления в зависимости от результата выполнения запроса. Пример: *PreconditionImplementaion.template*
Реализация управляющих конструкций вызова функций	Создание блока кода для вызова внутри функции, реализующей составной процесс, других функций на основе заданной управляющей конструкции (например, «последовательность», «цикл с предусловием»). Пример: *ControlConstrcutImplementaion.template*
Реализация эффекта выполнения функции	Создание SPARQL-запроса с телом из онтологии, вызов встроенных функций API для промежуточного ПО с целью изменения информационного содержимого. Пример: *EffectImplementaion.template*

Продолжение таблицы 7

Типы шаблонов	Описание
Сборочные	Создание файлов управления сборкой (CMakeList.txt) для кроссплатформенной системы CMake. Пример: *CMakeListBuild.template*
Отладочные	Шаблоны аналогичные типам «реализация» с включением в них отладочных методов и выводов. Пример: *EffectDebugImplementaion.template*

В соответствии с листингом 4.5 приведен пример шаблона для реализации предусловия функции на языке C++. Теги `<@expression_id@>` заменяются на имена-идентификаторы соответствующих SPARQL-выражений, которые извлекается обработчиком из онтологии. Тело запроса представляется тегом `<@expression_body@>`, который заменяется на выражение-запрос SPARQL ASK, выполняющий проверку ОИС на наличие той или иной информации. Далее происходит вызов соответствующей функции API для промежуточного ПО, результат, которой подвергается проверке с помощью условного оператора. В зависимости от результата проверки выполняется ветвление, где при положительном результате выполняется блок кода, реализующий эффект выполнения, который генерируется на основе шаблона *EffectImplementaion.template*.

```
1  std::string <@expression_id@> = std::string(<@expression_body@>);
2  bool <@expression_id@>_result = sslog_node_sparql_ask(node,
3                                                        <@expression_id@>);
4  if (<@expression_id@>_result == true) {
5    /* get results and give effects;
6       check and set user presence level */
7    <%effects%>
8    return <@output_id@>;
9  } else {
10   /* the condition was not met, a fault occurred */
11   return NULL;
12 }
```

Листинг 4.5 — Пример шаблона для реализация предусловия функции с возвращаемым значением

Таким образом, сгенерированный код накапливается в наборах файлов {name}.cpp и {name}.h или {name}.py для каждого агента (в зависимости от выбранного целевого языка программирования), где {name} — название агента, полученного из онтологии. Каждый набор файлов располагается в отдельном каталоге, который предварительно создается в случае отсутствия, с названием агента.

Индивиды процессов используются для генерации программного кода функций агентов. Модель процессов IOPEs [116] соответствует концепции функций на большинстве языков программирования (включая C++ и Python). Блоки кода для взаимодействия генерируются на основе обработки SPARQL-запросов, представленных в качестве литералов (literals), с помощью которых реализуются предусловия и эффекты выполнения функций, а также механизмы организации подписки с использованием интерфейса библиотеки (API) для организации информационно-управляемого взаимодействия.

С помощью языка описания схем XSD (XML-Schema) в онтологии описываются необходимые типы данных (string, unsignedLong и т.д.), которые преобразуются в их аналоги для соответствующего языка программирования. Структуры объектной модели данных, используемые в функциях, генерируются из онтологии предметной области. В тех случаях, когда каждый процесс описывается с достаточно полной моделью IOPEs, сервис может быть разработан с минимальными затратами времени разработчика на программирование.

4.2 Реализация контекстных сервисов

Реализация экспериментальных образцов предметно-ориентированных сервисов для организации СИИО получена на основе предложенного метода разработки интероперабельной программной инфраструктуры с использованием таких объектно-ориентированных языков программирования, как C++ и Python.

Сервис распознавания присутствия и анализа активности пользователей

В распознавании присутствия пользователей участвуют два агента сервиса S_{prs}: агент-адаптер сенсора, виртуализирующий сенсор присутствия, и агент-агрегатор присутствия, определяющий уровень присутствия пользователей. Для реализации данного алгоритма агент-агрегатор присутствия устанавливает пороговое значение последней активности, которое является временем, прошедшим с момента проявления последней сетевой активности до времени,

когда устройство считается покинувшим помещение. Агент-агрегатор присутствия подписывается на изменения индивидов класса PresenceInformation для отслеживания обновлений информации о присутствии мобильных устройств. Кроме того, агенту-агрегатору присутствия необходимо определить пороговое значение силы сигнала в беспроводной сети. Если значение RSSI устройства превышает порог, то устройство рассматривается как находящееся внутри помещения. Сенсор определяет ближайшие устройства, выполняющие передачу данных, и отправляет информацию о присутствии, включающую значение RSSI, агенту-адаптеру сенсора.

При первом обнаружении устройства агент-адаптер сенсора публикует в общее информационное содержимое индивид класса PresenceInformation с установленными значениями свойств: enter (время появления) и lastSeen (время последнего появления). При последующем обнаружении такого же устройства происходит обновление свойства lastSeen соответствующего индивида. Агент-агрегатор присутствия реагирует на изменения индивидов класса PresenceInformation, выполняет поиск семантической связи индивида с профилем пользователя и его персональной информации, включающую статус пользователя в системе (online/offline). Затем определяется присутствие пользователя. Когда значение свойства lastSeen превышает пороговое значение последней активности, пользователь считается покинувшим помещение. Агент-агрегатор присутствия на основе полученной информации вычисляет уровень присутствия пользователя и публикует значение соответствующего свойства presenceLevel в общее информационное содержимое (Full, Virtual, Physical, Absent).

Для накопления статистики и ее анализа используется сервис управления содержимым, который реализуется как веб-сервер. Агент-агрегатор присутствия создает и отправляет HTTP-запрос сервису управления содержимым с измерениями сенсора присутствия, если значение RSSI каждого измерения превышает пороговое значение последней активности (пользователь присутствует). Сервис управления содержимым формирует регистрационный файл для каждого зарегистрированного пользователя. Такой файл состоит из текстовых строк, каждая строка содержит два параметра: временная метка и значение RSSI.

Строка соответствует обнаруженному пакету беспроводной локальной сети, отправленного устройством пользователя. Уровень сетевой активности участника k определим как $L_k = n_k$, где n_k – количество строк в файле участника k.

Другим важным показателем является степень активности. Известное число измерений и время их получения позволяет вычислить степень активности для участника k:

$$f_k = \frac{j-i}{t(s_{kj}) - t(s_{ki})}, \quad 1 \leqslant i < j \leqslant n_k, \tag{4.1}$$

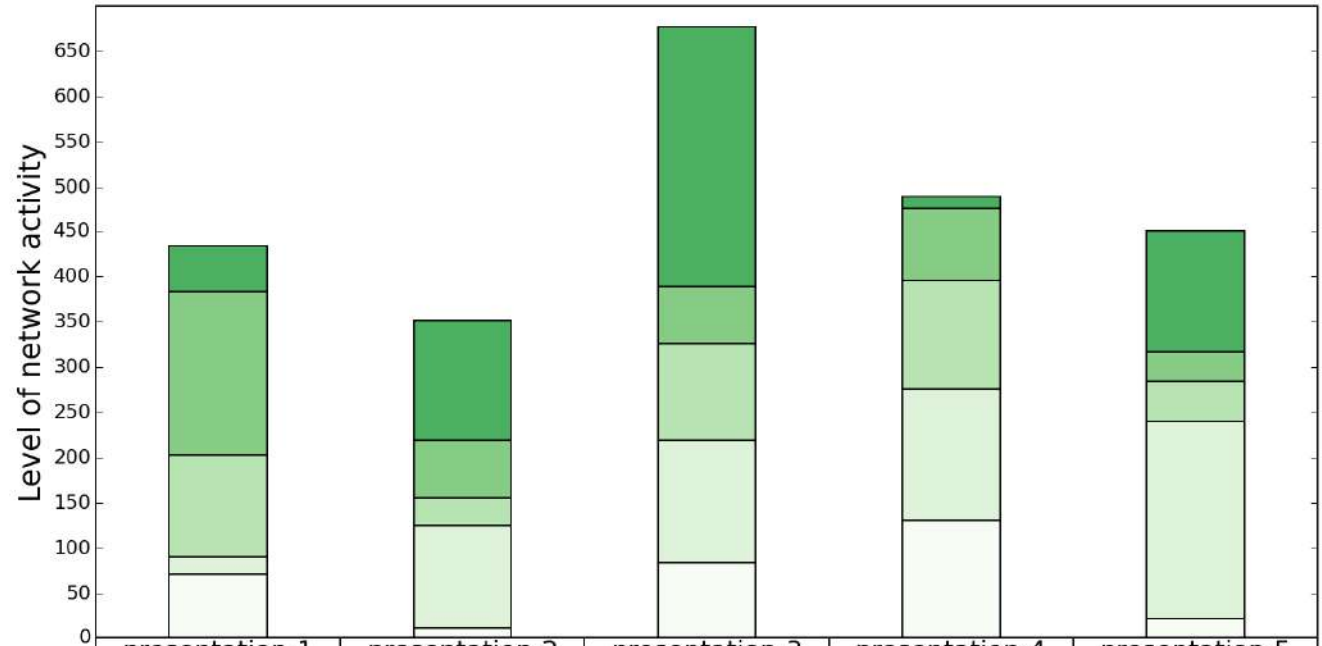
где s_{ki} – измерение i в файле участника k, $t(s_j)$ и $t(s_i)$ - значения временных меток в измерениях j и i в регистрационном файле участника k, соответственно.

Информация о сетевой активности пользователей эффективно представлять в виде слайда, для отображения его на экране сервиса презентации. Для вычисления и визуализации метрик сетевой активности участвуют три агента: агент-монитор активности (сервис S_{prs}), агент-интерпретатор веб-сервиса управления содержимым (сервис S_{cws}) и агент-воспроизводитель мультимедийной информации (сервис S_{cws}).

Агент-монитор активности подписывается на свойство, характеризующее конец мероприятия, endConference и ожидает публикации необходимых RDF-троек в общем информационном содержимом. Таким образом, в конце мероприятия агент-монитор активности с помощью HTTP GET запросов агенту-интерпретатору веб-сервиса управления содержимым получает доступ к регистрационным файлам каждого зарегистрированного и предоставившего MAC-адрес своего мобильного устройства пользователя. На основе регистрационных файлов вычисляются необходимые метрики сетевой активности. Далее формируется слайд, содержащий информацию активности пользователей по каждому докладу. Слайд загружается на сервис управления содержимым с помощью HTTP POST запроса с предоставлением к нему внешнего доступа с помощью специализированного URL. Затем агент-монитор активности публикует индивид уведомления с именем файла суммарного слайда для агента-воспроизводителя мультимедийной информации, который подписан на соответствующий класс PresentationNotification. По подписке агент-воспроизводитель мультимедийной информации извлекает из общего информационного содержимого URL для доступа к веб-сервису управления содержимым и, используя полученное имя файла, формирует URL для доступа к суммарному слайду. Полученный слайд отображается на общественном экране сервиса презентации.

Онтология спецификации для сервиса S_{prs} используется для генерации кода соответствующих программных агентов, участвующих в реализации сервиса. На

рисунке 4.3 представлен пример того, как индивид класса атомарного процесса SetUserPresenceLevel преобразуется в код функции на языке C++ для агента-агрегатора присутствия. Объекты классов (например, PresenceInfo), используемые в функции, генерируются из онтологии проблемной области. Предусловие процесса преобразуется в блок кода, реализующий выполнение и обработку SPARQL-ASK запроса на проверку наличия необходимых RDF-троек в ОИС для объекта класса PresenceInformation (информация о присутствии пользователей).



	presentation-1	presentation-2	presentation-3	presentation-4	presentation-5
user-1	50	132	288	15	133
user-2	180	64	63	80	33
user-3	112	30	107	119	44
user-4	20	115	135	146	220
user-5	71	10	84	130	20

Рисунок 4.2 — Вариант визуализации уровня сетевой активности

Метрики сетевой активности используются для анализа мероприятия. Например, визуализируя уровень сетевой активности для каждого доклада, можно определить какой доклад вызвал наибольшую активность и вклад каждого пользователя. На рисунке 4.2 представлен вариант визуализации такой информации, наибольшую сетевую активность вызвал доклад presentation-3 и устройство пользователя user-1 отправило больше всего пакетов беспроводной локальной сети во время данного доклада.

Агенты сервиса S_{prs} реализованы на языке Python при помощи различных инструментов: микрофреймворк Flask (веб-взаимодействие агента-адаптера и сенсора), библиотека Matplotlib (визуализация суммарного слайда на стороне агент-монитора активности). Для взаимодействия с общим информационным

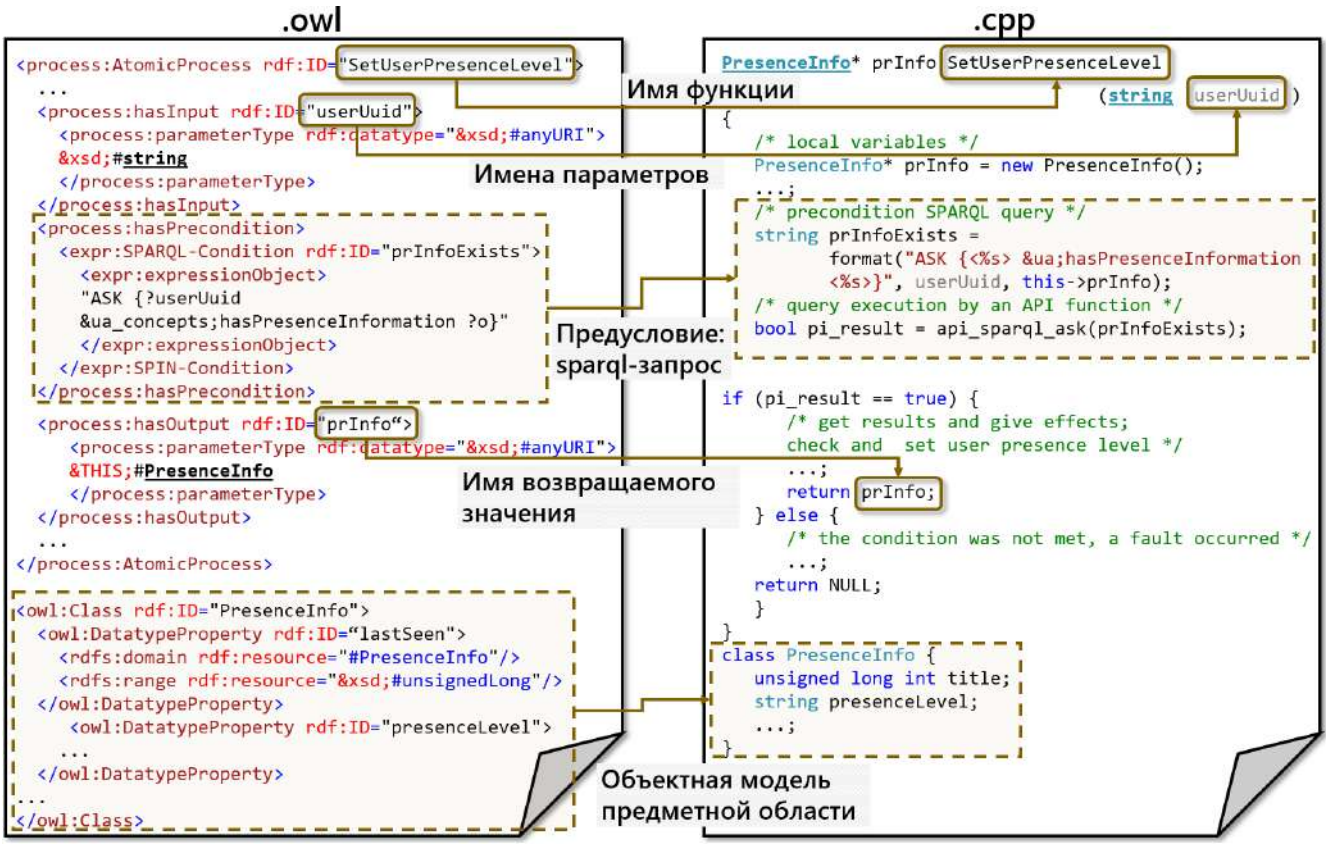


Рисунок 4.3 — Пример генерации программного кода функции SetUserPresence для агента-агрегатора присутствия

содержимым использована низкоуровневая библиотека M3-Python KPI, позволяющая работать с онтологической информацией на языке Python.

Сервис сопровождения и визуализации плана деятельности людей

В СИИО при проведении мероприятий реализуются несколько сценариев извлечения информации из общего информационного содержимого. Основными сценариями являются извлечение информации о текущей секции и программе мероприятия, а также отображение текущего докладчика.

Получение и отображения программы мероприятия происходит на мобильном устройстве пользователя и на общем на экране с программой мероприятия. Агент-визуализатор плана деятельности извлекает из общего информационного содержимого индивид секции. Индивид секции содержит в себе список индивидов временных слотов, каждый из которых описывает отдельный доклад.

Агент-визуализатор в цикле запрашивает каждый индивид временного слота из списка и отображает полученную информацию на экране.

Другим сценарием является получение информации о текущем докладчике и переключение текущего докладчика. После того, как агент-визуализатор плана деятельности получил текущую секцию со списком временных слотов и отобразил программу мероприятия, агент подписывается на свойство секции, содержащее текущий временной слот (т.е. текущий доклад). После завершения доклада агент-контроллер управления планом деятельности меняет текущий временной слот секции, агент-визуализатор плана деятельности получает по подписке обновленную информацию о текущем докладчике и отображает соответствующие изменения на экране.

Агент-визуализатор плана деятельности людей реализован на языке C++ при помощи кросс-платформенного инструментария разработки Qt. Для дизайна графического интерфейса использован язык QML. Для взаимодействия с общем информационным содержимым использована библиотека SmartSlog [17], позволяющая работать с онтологической информацией на языке C.

Сервис совместного пополнения информационного содержимого историческими знаниями

На основе разработанной онтологии для сервиса S_{enr} генерируется код программных агентов, участвующих в реализации данного сервиса. В соответствии с рисунком 4.4 представлен пример генерации программного кода на языке C++, реализующего составной процесс DetectHiddenRelations для агента-контроллера обнаружения исторических связей. По аналогии с атомарным процессом, составной процесс также реализуется программной функцией: ее название, входные, выходные параметры и другие составные элементы определяются на основе соответствующих понятий и отношений онтологии.

Для этой цели используются представленные процедуры алгоритма кодогенерации. Особенностью генерации в данном случае является преобразование управляющей конструкции Repeat-While в оператор цикла while, условие выполнения которого задается запросом SPARQL-ASK. Составляющие атомарные процессы преобразуются в вызовы соответствующих им функций с заданны-

ми параметрами и возвращаемым значением. На основе заданного в онтологии эффекта выполнения процесса генерируется код, реализующий выполнение соответствующего запроса SPARQL-INSERT.

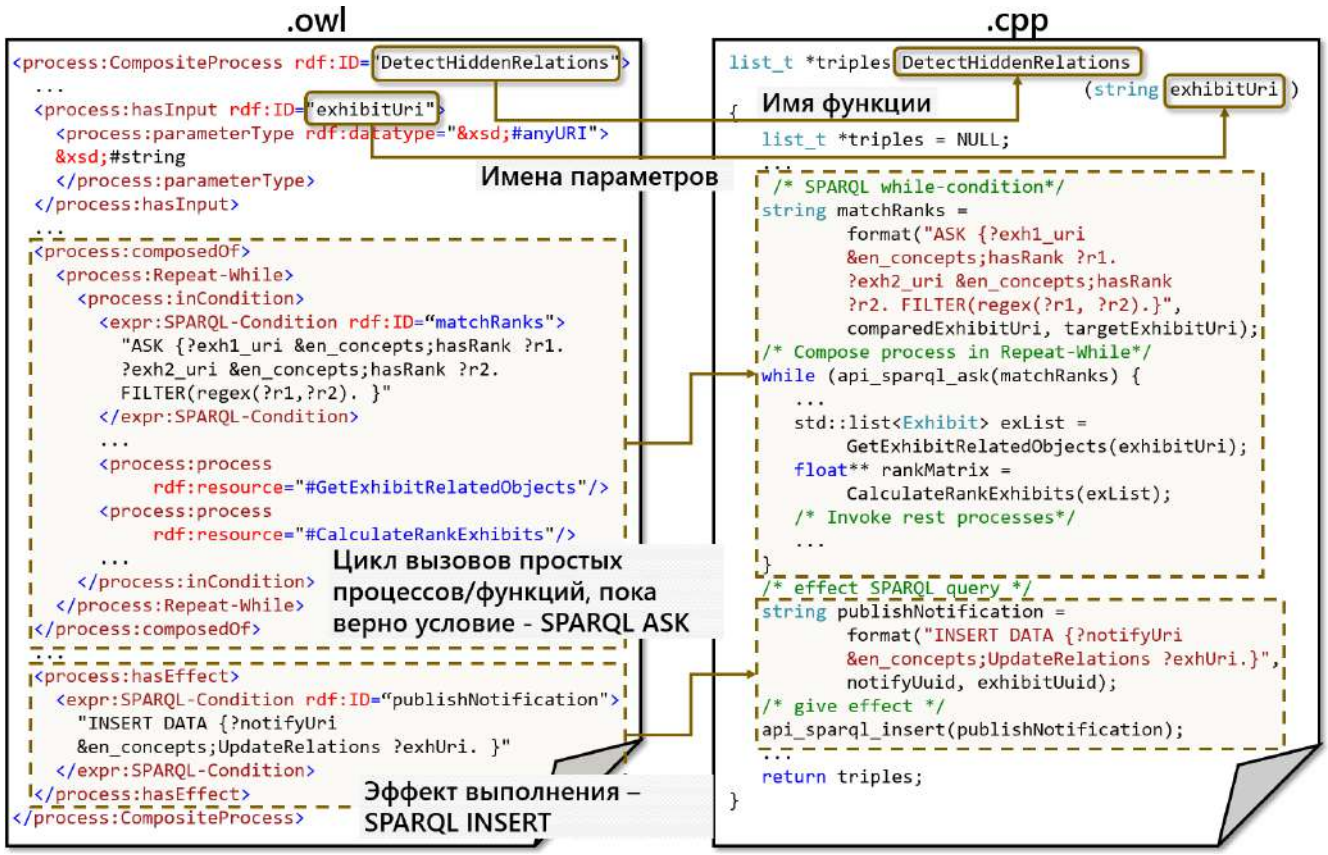


Рисунок 4.4 — Пример генерации программного кода функции DetectHiddenRelations для агента-контроллера обнаружения исторических связей

Метрики программной реализации сервисов

Эффективное использование онтологий сервисов СИИО зависит от того, насколько эффективно могут выполняться операции промежуточного ПО по мере увеличения количества компонентов онтологии и соответствующего описания семантики в RDF. Таблица 8 отображает основные метрики онтологии реализованных экспериментальных образцов сервисов. Онтологическое представление сервиса включает в среднем 1124 RDF-троек (на основе представленных четырех сервисов).

Учитывая производительность промежуточного ПО с большим набором данных, в крупномасштабных развертываниях процесс построения и достав-

Таблица 8 — Метрики разработанных онтологии сервисов

Сервис	Часть онтологии	Кол-во свойств данных	Кол-во объектных свойств	Количество классов и их индивидов	Кол-во RDF троек
S_{prs}	Профиль	19	6	29	162
	Модель процесса	52	119	136	921
	Итого	71	125	165	1083
S_{cws}	Профиль	21	10	32	189
	Модель процесса	60	150	172	1146
	Итого	81	160	204	1335
S_{enr}	Профиль	13	5	19	111
	Модель процесса	83	184	152	1257
	Итого	96	189	171	1368
S_{mnt}	Профиль	9	2	19	90
	Модель процесса	42	82	83	621
	Итого	51	84	102	711

ки сервиса может оказаться трудоемким процессом для ресурсно-ограниченных устройств с ограниченными возможностям. Тем не менее, хранение всех семантических данных, запуск промежуточного ПО на мощной серверной машине и использование различных технологий для поддержки эффективной передачи данных по маломощным и низкоскоростным устройствам и сетям позволяет решить эту проблему [57].

Метрики полученной программной реализации для сервисов S_{prs}, S_{cws}, S_{enr}, S_{mnt} представлены в таблице 9 в следующем разделе вместе с долей сгенерированного для каждого агента кода. Основными технологиями являются: Qt/C++, QML, Python, HTML, JS, CSS, XML.

4.3 Оценка трудозатрат на разработку программной инфраструктуры

Полученные программные реализации участвуют в экспериментальных исследованиях, направленных на оценку эффективности предложенного в работе метода разработки программной инфраструктуры для организации СИИО, кото-

рый основан на применении в нем других полученных результатов. В качестве определения эффективности используется формулировка из стандартов ГОСТ Р ИСО: «соотношение между достигнутым результатом и использованными ресурсами» [8]. Процесс определения эффективности следует методике, которая направлена на оценку трудозатрат на разработку сервисов за счет использования унифицированной онтологии сервиса и генератора программного кода взаимодействия агентов. Полученные оценки сравниваются с трудозатратами на разработку программной инфраструктуры без использования предложенного метода. Трудозатраты выражаются в человеко-часах, причем вместо часов могут использоваться другие единицы измерения продолжительности времени. Трудозатраты с использованием метода (E) раскладываются на составляющие: трудозатраты на проектирование (e_d), накладные затраты на проектирование, трудозатраты на программирование (e_c), накладные затраты на программирование. Каждая из составляющих оценивается в сравнении с трудозатратами на разработку без использования метода (E', $e_d{}'$, $e_c{}'$).

Ключевыми этапами (учитывая трудозатраты) проектирования многоагентных программных систем являются: (а) определение ролей агентов и их функциональное описание, (б) концептуальное моделирование межролевого взаимодействия на основе выбранного протокола, (в) моделирование взаимодействия пользователя и системы, определение интерфейса доступа. Кроме того, немаловажным является этап создания структуры кода для каждого агента и системы в целом. Использование разработчиком предложенного в работе метода позволяет фиксировать полученные проектные решения (роли агентов, протокол и модель взаимодействия агентов, интерфейс сервисов) при непосредственном создании онтологического описания сервисов в унифицированных терминах.

Известно, что использование онтологий на этапе проектирования увеличивает трудозатраты разработчика для создания проектных решений. Однако, одни дополнительные затраты можно минимизировать, а другие дают дополнительные возможности на следующих этапах разработки (например, автоматизация программирования, самоорганизация агентов). Одним из способов минимизации затрат является использование существующих инструментальных средств автоматизированного проектирования (например, Protégé), которые предоставляют программную среду для быстрого прототипирования, в которой разработчик онтологий может быстро создавать индивидов, экспериментировать с семантическими ограничениями, визуализировать онтологические описания. Более того,

метод разработки использует предложенные типовые модели взаимодействия агентов и предметно-ориентированные модели сервисов, упрощая проектирование сервисов за счет предоставления разработчику архитектурных и поведенческих абстракций агентов. Таким образом, можно утверждать, что трудозатраты на проектирование сервисов, учитывая накладные затраты, с использованием предложенного метода разработки программной инфраструктуры сопоставимы с трудозатратами на проектирование многоагентных систем без использования метода: $e_d = c * e_d{}'$, где $1 < c < 2$ – фактор учета накладных затрат.

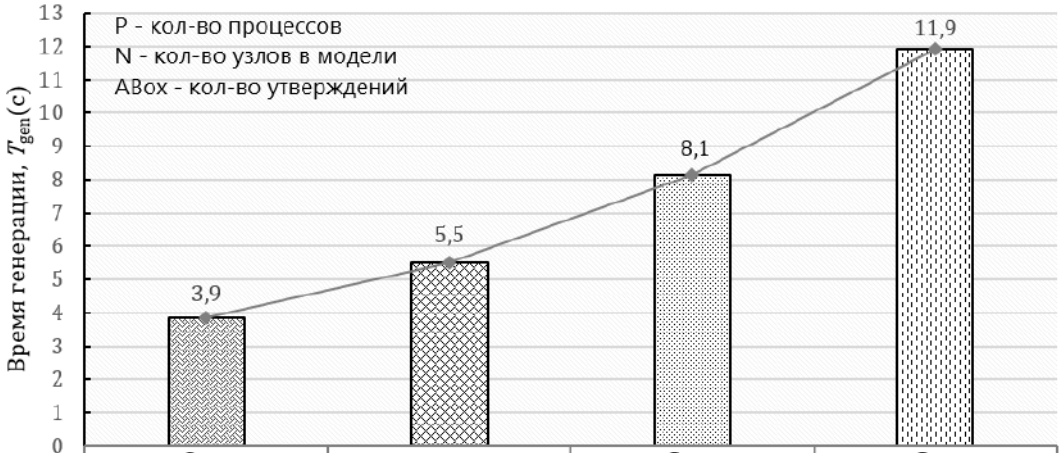
Дополнительные трудозатраты на проектирование позволяют получать единообразные онтологические модели сервисов, которые определяют интерфейс взаимодействия и описывают семантику выполнения. Благодаря такому единообразному способу проектирования, сервисы различных предметных областей приобретают возможность взаимодействовать друг с другом независимо от вычислительной среды, в которой разворачиваются СИИО. Использование метода не ограничивается этапом проектирования. Полученные проектные решения в виде онтологий сервисов используются для автоматизации дальнейших процессов программирования сервисов (создание объектной модели и структур данных, методы, функции кода).

Трудозатраты на программирование исследуются на основе отношения общего количества строк исходного кода агентов к автоматически созданному с помощью полученной реализации генератора программного кода. В соответствии с таблицей 9 представлена доля сгенерированного программного кода для агентов, участвующих в реализации сервисов. Сервисы разработаны на основе предложенных предметно-ориентированных моделей. Программный код в независимости от роли программного агента состоит из следующих блоков: (1) структуры объектной модели данных и методы для работы с ними, (2) информационно-управляемое взаимодействие на основе поддерживаемых операций для промежуточного программного обеспечения, (3) внутренняя логика, включающая локальную обработку общей информации. Средняя доля сгенерированного программного кода составила 23,4%, причем наибольшие результаты пришлись на блок «информационно-управляемое взаимодействие». Для объектной модели данных генератор также показывает высокие показатели, средний процент покрытия соответствующего исходного кода сгенерированным составляет 68%.

Таблица 9 — Доля сгенерированного программного кода для реализаций сервисов

Сервис	Агенты и их роли		Объектная модель данных		Информационно-управляемое взаимодействие		Внутренняя логика		Итого	
			SLOC	%	SLOC	%	SLOC	%	SLOC	%
Сервис распознавания присутствия и анализа активности	Агент-*адаптер* сенсора	общ.	26	8,8	78	26,4	191	64,8	295	100
		сген.	21	7,1	40	13,6	9	3	**70**	**23,7**
	Агент-*агрегатор* присутствия	общ.	18	11,2	44	27,5	98	61,2	160	100
		сген.	14	8,7	18	11,3	5	2,8	**37**	**23,1**
	Агент-*монитор* активности	общ.	74	13,2	88	15,7	392	71,1	560	100
		сген.	52	9,3	75	13,4	13	2,3	**140**	**25**
Сервис сопровождения и визуализации плана деятельности	Агент-*визуализатор* программы СД	общ.	314	14,9	519	24,6	1274	60,5	2107	100
		сген.	135	6,4	280	13,3	27	1,3	**442**	**20,9**
	Агент-*контроллер* управления программой СД	общ.	316	12,8	916	37,1	1236	50,1	2468	100
		сген.	210	8,5	540	21,9	22	0,9	**772**	**31,3**
	Агент-*интерпретатор* веб-сервиса управления содержимым	общ.	92	8,2	217	19,3	817	72,5	1126	100
		сген.	60	5,3	106	9,4	8	0,7	**174**	**15,4**
	Агент-*воспроизводитель* мультимедийной информации	общ.	283	9,3	613	20,1	2153	70,6	3049	100
		сген.	186	6,1	253	8,3	37	1,2	**476**	**15,6**
Сервис совместного пополнения содержимого историческими данными	Агент-*искатель* исторической информации	общ.	65	9,6	162	23,9	451	66,5	678	100
		сген.	50	7,4	103	15,2	7	1	**160**	**23,6**
	Агент-*контроллер* исторических связей	общ.	222	12,6	515	29,2	1028	58,2	1765	100
		сген.	148	8,4	339	19,2	23	1,3	**510**	**28,9**
	Агент-*агрегатор* обогащения информации	общ.	391	11,9	898	27,3	2001	60,8	3290	100
		сген.	296	9	513	15,6	26	0,8	**835**	**25,4**
Сервис мониторинга объектов физической среды	Агент-*интерпретатор* базы данных сенсорной сети	общ.	82	11,6	144	20,3	484	68,1	710	100
		сген.	53	7,5	106	14,9	9	1,3	**168**	**23,7**
	Агент-*агрегатор* измерений сенсорной сети	общ.	82	9,8	194	23,2	562	67	838	100
		сген.	52	6,2	106	12,6	5	0,6	**163**	**19,4**
	Агент-*контроллер* узлов сенсорной сети	общ.	40	12,7	89	28,5	183	58,8	312	100
		сген.	29	9,3	54	17,4	6	1,9	**89**	**28,6**

На процесс программирования сервисов с использованием метода накладываются дополнительные трудозатраты, связанные с временем, затраченным на процесс генерации программного кода. Генератор принимает в качестве входных



	S_{enr}	S_{mnt}	S_{cws}	S_{prs}
P	12	14	21	24
N	534	612	887	1010
ABox	254	285	735	733

Рисунок 4.5 — Время генерации программного кода для реализаций сервисов

параметров разработанные онтологии сервисов и онтологию предметной области. Время генерации программного кода зависит от онтологических метрик (цикломатическая сложность, размер словаря), характеризующих сложность обхода метамодели онтологии (онтологический граф). Эксперименты показали (см. рисунок 4.5), что при обработке онтологии для сервиса сопровождения и визуализации программы СД, которая имеет самые высокие показатели метрик из разработанных онтологий, время обхода метамодели и генерации кода не превышает 12 с.

Качество сгенерированного кода исследуется на основе: (1) исследования производительности полученных программных реализаций сервисов, (2) вычисления и оценки метрик качеств сгенерированного программного кода. Так, например, экспериментальные исследования сервиса сопровождения и визуализации плана деятельности выполнялись на основе оценки производительности возможностей (прикладных функций) хранения и получения программы совместной деятельности, обработки программы по ходу деятельности, показа медиа- и другой визуальной информации. В качестве иллюстрирующего примера совместной деятельности рассмотрена организация проведения конференций.

Основные процессы выполнения сервисов выполняются программными агентами, запущенными локально, на оборудовании, установленном в вычислительной среде для информационного окружения проведения конференций. В частности, компьютер, на котором запущен агент-воспроизводитель мультимедийной информации связан с общественным экраном для отображения медиа-

информации по время докладов (например, слайд-шоу). Основная схема сценария представлена в соответствии с рисунком 4.6.

Рисунок 4.6 — Сценарий выполнения экспериментальных исследований для сервиса сопровождения и визуализации программы совместной деятельности

Агент-воспроизводитель мультимедийной информации обрабатывает различный мультимедийный контент (например, файлы презентации), полученный в результате взаимодействия с агентом-интерпретатором веб-сервиса управления содержимым. Каждый PDF-файл в результате обработки агентом раскладывается на отдельные JPG-файлы, каждый из которых представляет отдельный слайд. Веб-сервис управления содержимым, реализуется с помощью сервера, обрабатывающего различные запросы, в том числе и на загрузку сохраненных файлов. Ссылка на каждый файл размещается в общем информационном содержимом окружения, что делает файлы доступными для других программных агентов. Агент-контроллер управления программой СД связывает ссылки на файлы слайдов презентаций с соответствующими докладами.

Агент-воспроизводитель мультимедийной информации отвечает за визуализацию показа слайдов текущего выступающего на общем экране. Агент хранит информацию о номере текущего слайда и об общем количестве слайдов презентации. Это решение позволяет агенту загрузить текущий слайд, используя веб-ссылку на файл, а затем отобразить слайд на общем экране. Рассмотренные операции обеспечивают небольшую нагрузку на программную реализацию агентов; нагрузка зависит только от размера файла слайда (от 50 Кбайт до 250 Кбайт).

Программа конференции состоит из докладов, которые будут представлены участниками конференции в ходе совместной деятельности. Полное описание программы конференции хранится в общем информационном содержимом. Следовательно, одними из ресурсоемких операций для агента-контроллера управления программой СД являются получение программы из общего информационного содержимого с помощью сетевого взаимодействия (время T_{ntw}) и локальная обработка программы конференции (время T_{loc}). Агент выполняет данные операции каждый раз, когда программа конференции обновляется в общем информационном содержимом. Например, организаторы добавили новый доклад в программу или изменили информацию о выступлении (например, время начала доклада). На стороне агента получение и обработка программы реализуются функцией преобразования программы из онтологического представления в локальные структуры объектной модели предметной области, которые используются в реализации внутренней логики программного агента. Таким образом, программа мероприятия хранится в локальной памяти агента и динамически изменяется в зависимости от текущего состояния образа программы в общем информационном содержимом (реализуется подписка на изменения).

Сущность программы конференции описывается с помощью соответствующего онтологического представления (для совместного использования в общем информационном содержимом) и набора утверждений. Программа конференции представляется как индивид класса Section, значения свойств которого описывают характеристики докладов (название, время начала) и их последовательности (список временных слотов). Можно оценить количество RDF-троек, необходимых для представления программы конференции: $3 + 4n$, где n – количество докладов в секции. Так, например, при $n = 20$ для передачи цифрового образа программы (набор RDF-троек, представленных в виде XML-документа) промежуточному ПО необходимо передать по сети около 2 Кбит информации агенту-контроллеру управления программой СД.

Полученные экспериментальные оценки для $T_{\text{loc}}(n)$ и $T_{\text{ntw}}(n)$ представлены на рисунках 4.7 и 4.8, соответственно. Для анализа масштабируемости программы мероприятия при разных конфигурациях количество докладов менялось в интервале $2 \leqslant n \leqslant 50$. В целом, провелось 100 измерений $T_{\text{loc}}(n)$ и $T_{\text{ntw}}(n)$. Основной график строится на основе вычисления стандартного отклонения полученных значений измерений и позволяет оценить усредненное поведение. Кроме того, каждый график содержит линейные и полиномиальные регрессии, необходимые

для оценки характера поведения измеряемых величин: рост $T_{\text{loc}}(n)$ схож с линейным; поведение $T_{\text{ntw}}(n)$ близко к полиномиальному.

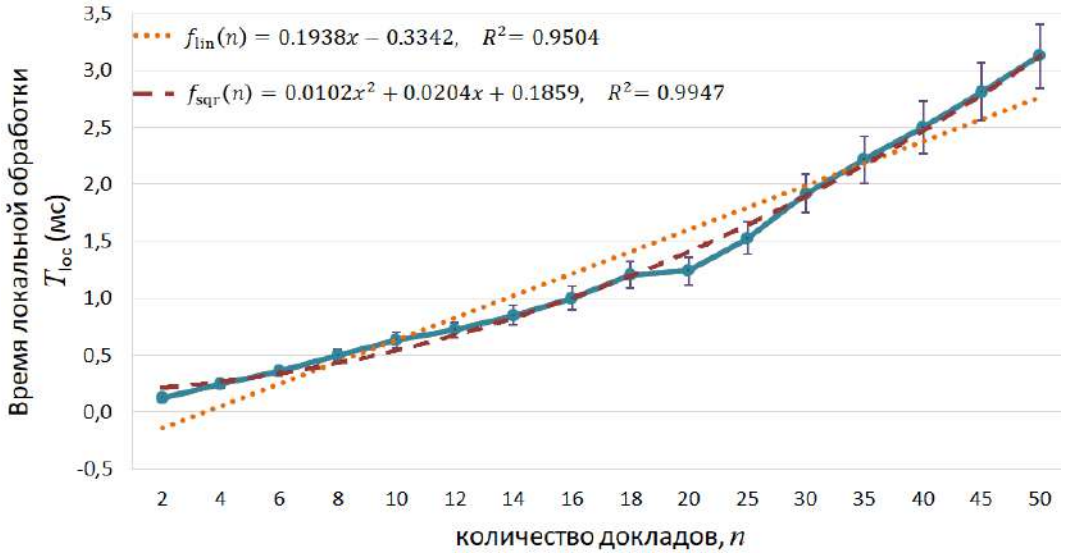


Рисунок 4.7 — Экспериментальное время $T_{\text{loc}}(n)$ локальной обработки программы конференции, состоящей из n докладов

При обновлении секции сервис S_{cws} из общего информационного содержимого извлекает полное RDF-дерево, представляющее доклады секции, однако обновления могут содержатся только в какой-либо его части. Использование в реализации сервиса S_{cws} языка запросов SPARQL для механизма подписки позволит получать только необходимые элементы цифрового образа программы, уменьшая при этом средние показатели измерений для T_{ntw}. При оценке T_{ntw} также следует учесть заданную пропускную способность беспроводной локальной сети, значения которой позволят оценить верхние границы поведения.

В ходе проведения экспериментальных исследований установлено, что для организации конференции, состоящей из 8-16 докладов (типичный размер секции), результат суммирования средних показателей для $T_{\text{loc}}(n)$ и T_{ntw} остается в рамках 2 с. Тогда как 8 с для локальной обработки и передачи по сети 50 цифровых образов докладов является довольно существенным, учитывая нелинейный рост $T_{\text{ntw}}(n)$. На основе вычисления стандартного отклонения для T_{ntw}, которое составляет в среднем 0,58 с можно сделать вывод о наличии высоких нагрузок на беспроводную локальную сеть. Увеличение пропускной способности сети с сохранением устойчивого соединения с промежуточным ПО позволяет обеспечить высокую производительность программных агентов, уменьшив время T_{ntw}. Визуализация показа слайдов текущего выступающего на общем экране эффективно решается за счет хранения в информационном содержимом информации о номере

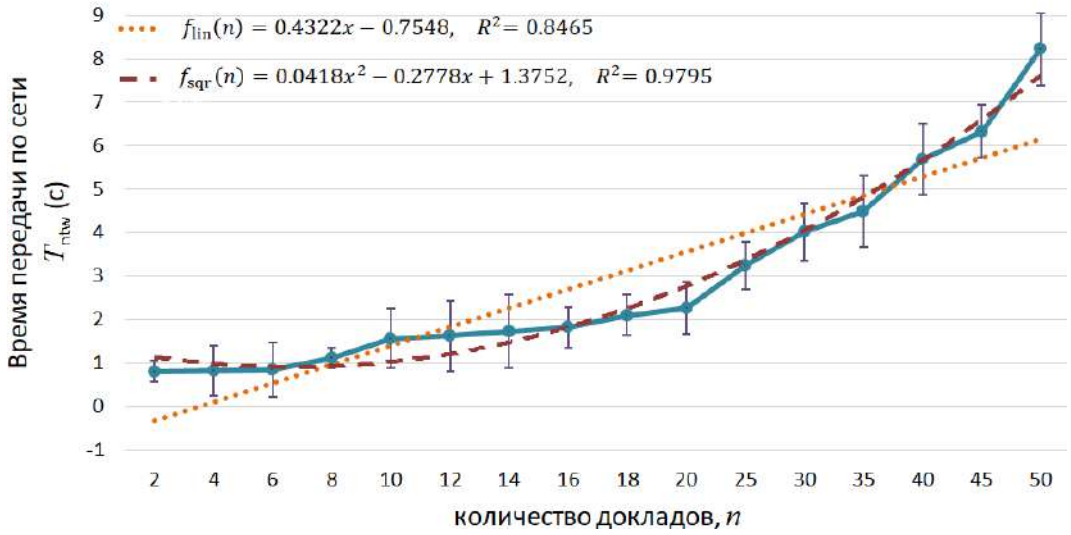


Рисунок 4.8 — Экспериментальное время $T_{\text{ntw}}(n)$ получения программы конференции (n докладов) из информационного содержимого

текущего слайда и об общем количестве слайдов. Извлечение такой информации агентом-воспроизводителем мультимедийной информации не требует много производительных ресурсов, так как возникающая нагрузка обеспечивается только сетевой загрузкой текущего изображения для слайда презентации посредством размещенной в общем информационном содержимом веб-ссылки, а, следовательно, зависит только от размера файла и производительности сети.

Экспериментальные исследования сервиса распознавания присутствия и анализа активностей пользователей выполнялись для оценки производительности приведенных раннее сценариев S_1–S_3 («Участники приходят на мероприятие», «Участники приходят в помещение и выходят из него во время мероприятия», «Анализ активности участников», соответственно). Для сценариев S_1 и S_2 оценивается время перехода между состояниями представленными в описанных раннее выражениях 3.1, 3.2 и 3.3.

Шаг 1. Сенсор определяет ближайшие устройства, выполняющие передачу данных, и отправляет информацию о присутствии агенту-адаптеру сенсора.

Шаг 2. При первом обнаружении MAC-адреса устройства, агент-агрегатор присутствия публикует информацию о присутствии в онтологической форме: один OWL-индивид класса Presence с двумя свойствами данных: lastSeen и enter.

Шаг 3. Агент-агрегатор присутствия следит за добавлением индивидов класса Presence и изменением свойств данных для существующих индивидов этого класса. Такое постоянное обнаружение использует операцию подписки. При

обнаружении изменений агент вычисляет уровень присутствия и обновляет значение соответствующего свойства данных.

Шаг 4. Сервисы, которым необходима информация о присутствии пользователей (напр., сервис сопровождения и визуализации плана деятельности) подписываются на обновления свойства данных presenceLevel.

Общее время выполнения данных шагов является временем определения присутствия пользователя. Оно пропорционально количеству пользователей и зависит от производительности беспроводной сети. Основная нагрузка связана с выполнением операций публикации и подписки.

На рис. 4.9 представлено измеренное распределение времени выполнения шагов 1–4 для одного пользователя. Размер выборки — 100 элементов. Среднее время составляет 677 мс. Характерны частые выбросы, которые, как правило, зависят от работы сети, т.к. основная нагрузка связана с передачей данных.

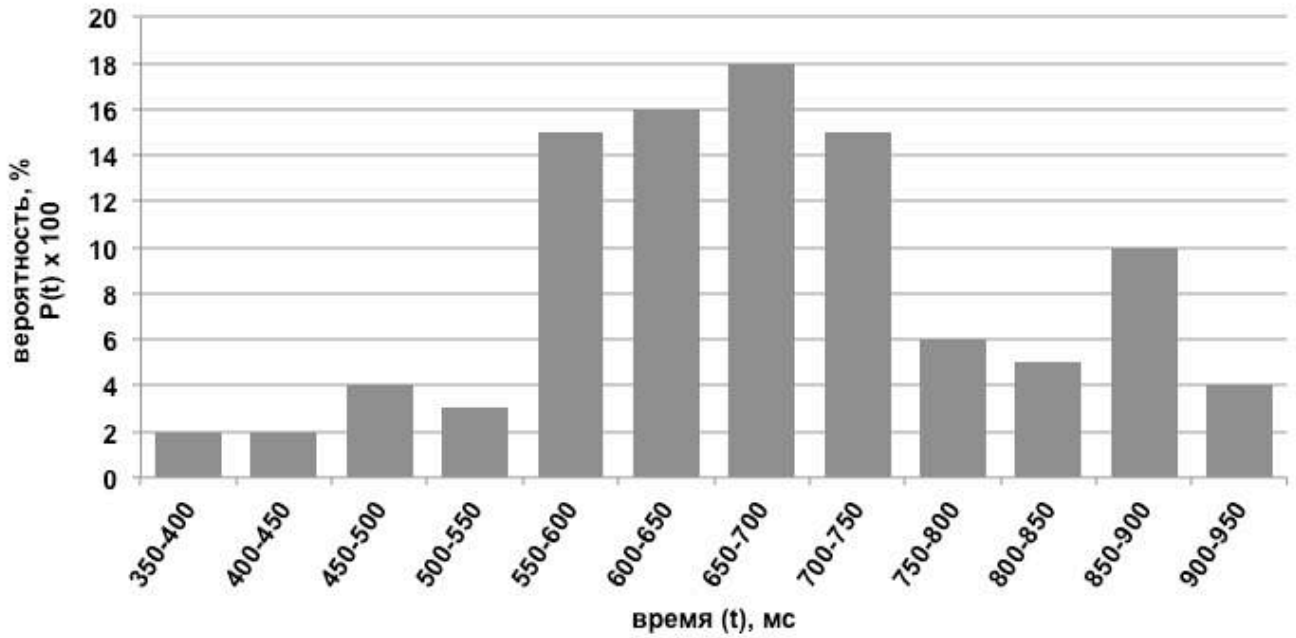


Рисунок 4.9 — Распределение времени определения присутствия пользователя

Для группы сценариев S_3 важной задачей является оценка порогового временного значения последней активности пользователя. Необходимо установить пороговое значение последней активности, в зависимости от вида мобильного устройства.

Рассмотрим следующие мобильные устройства (смартфоны): iPhone 8 и Samsung Galaxy S8. Каждый из них периодически отправляет пакеты (кадры) запросов на зондирование для определения ближайших точек доступа (если беспроводная сеть активна и не используется). Сенсор присутствия отслеживает

такие пакеты, определяя присутствие мобильных устройств даже при неактивной передаче данных. Частота зондирования зависит от внутренних алгоритмов, реализуемых производителем на устройстве. Пакеты запросов последовательно отправляются по всем каналам передачи данных.

Частота отправки была измерена с помощью мониторинга каналов беспроводной сети. Измерялся период времени, по истечению которого сенсор получает пакеты запросов на зондирование беспроводной локальной сети для конкретных смартфонов. Для каждого из устройств наблюдались регулярные временные промежутки, в течении которых приходил пакет запроса на зондирование. Для устройства iPhone 8 данный период определялся диапазоном $[40, 50]$ секунд, для устройства Galaxy S8 — диапазоном $[27, 34]$. Для каждого устройства определены наиболее часто обнаруживаемые периоды: iPhone 8 — период 45 с, Galaxy S8 — период 30 с. В экспериментах устройства были расположены примерно в 5 метрах от сенсора присутствия. Полученные результаты дают представления о периоде отправки запросов типичными представителями устройств производителей Apple и Samsung.

В качестве порогового значения последней активности следует выбирать верхнюю границу времени для используемых устройств. В проведенных экспериментах данное значение составляет 50 с. Далее будем использовать более консервативное значение, равное 60 с.

Агент-агрегатор присутствия постоянно отслеживает свойство lastSeen и реагирует, когда значение данного свойства превышает пороговое временное значение последней активности. Таким образом, пользователь считается покинувшим помещение. Затем агент-агрегатор присутствия обновляет значение свойства presenceLevel для соответствующего пользователя. Когда устройство снова обнаруживается в помещении, то пользователь рассматривается как вновь прибывший, и выполняются шаги, описанные для сценариев группы S_1.

Анализ степени активности позволяет определить, когда пользователь покидал помещение во время мероприятия. Пусть пороговое значение последней активности равно 60 с (т.е. степень активности рассчитывается каждые 60 с во время проведения мероприятия). Эксперименты показывают, что если степень активности становится меньше $0{,}017$, то пользователь покинул помещение. Таким образом, данное значение можно использовать как пороговое. Существует также зависимость степени активности от типа пользователя, например, «очень активны» и «слабо активный».

На производительность сценариев группы S_3 влияет объем памяти, занимаемый регистрационными файлами на сервисе управления содержимым, и время их обработки для вычисления метрик сетевой активности. В измерительном эксперименте проведено мероприятие с 10 докладчиками, каждое выступление с запланированной продолжительностью 15 минут. Пользователи использовали мобильные устройства для управления презентацией и для доступа к презентациям других пользователей. Сервис анализа активности запущен на выделенной ЭВМ (CPU 2.50GHz, RAM 8Gb, Windows 10) и вычисляет уровень сетевой активности, степень активности и среднее значение RSSI для каждого участника. Среднее время обработки файла составило 0,72 с. Средний размер файла составляет около 346 КБ. Потребовалось около 3500 КБ дискового пространства для хранения регистрационных файлов пользователей на сервисе управления содержимым.

4.4 Выводы

В главе описана программная реализация комплекса программных средств: а) реализация генератора программного кода взаимодействия агентов на основе полученного алгоритма автоматизации программирования и концептуальной модели сервиса; б) экспериментальные образцы предметно-ориентированных сервисов на основе метода разработки интероперабельной программной инфраструктуры для организации СИИО для вычислительных сред интеллектуального зала, умного музея и промышленного предприятия. На основе реализованного комплекса программных средств выполнены экспериментальные исследования с целью оценки эффективности метода разработки программной инфраструктуры на этапах проектирования и программирования сервисов.

Накладные трудозатраты разработчика при использовании предложенной унифицированной онтологии на этапе проектирования сервисов минимизируются за счет применения программных сред для автоматизированной разработки онтологий (например, Protégé). Более того, использование предложенных типовых моделей взаимодействия агентов и предметно-ориентированных моделей сервисов упрощает проектирование сервисов за счет предоставления разработчику архитектурных и поведенческих абстракций агентов. Установлено, что

трудозатраты на проектирование сервисов, учитывая накладные затраты, с использованием предложенного метода разработки программной инфраструктуры сопоставимы с трудозатратами на проектирование многоагентных систем без использования метода: $e_d = c * e_d{}'$, где $1 < c < 2$.

Полученные проектные решения в виде онтологий сервисов используются для автоматизации дальнейших процессов программирования сервисов. Оценка трудозатрат на программирование выполнена на основе отношения общего количества строк исходного кода агентов для экспериментальных образцов сервисов к автоматически созданному с помощью полученной на основе предложенного алгоритма автоматизации программирования взаимодействия агентов реализации генератора программного кода. В результате, средняя доля сгенерированного программного кода для агента составила 23,4%, где 7,6% пришлось на объектную модель данных предметной области, а 15,8% на информационно-управляемое взаимодействие.

Заключение

В диссертационной работе предложено решение актуальной научно-технической задачи по повышению эффективности разработки интероперабельной программной инфраструктуры совместно используемого информационного интернет-окружения за счет унифицированного моделирования сервиса как системы взаимодействующих агентов и автоматизированного программирования взаимодействия на основе кодогенерации. Получены следующие результаты:

1. Сформирован метод разработки интероперабельной программной инфраструктуры, обеспечивающей информационно-управляемое взаимодействие агентов для интеграции динамических и неоднородных ресурсов при построении сервиса. Метод позволяет снизить трудозатраты на разработку программной инфраструктуры для сервисов СИИО за счет унифицированной и автоматизированной разработки сервиса как многоагентной системы.
2. Предложена концептуальная модель сервиса, позволяющая описывать с помощью онтологии варианты информационно-управляемого взаимодействия агентов для построения контекстных сервисов и их композиции на основе технологий Семантического веба. За счет такого унифицированного онтологического описания процессов построения сервиса, сервисы различных предметных областей приобретают возможность взаимодействовать друг с другом для решения совместных задач независимо от вычислительной среды, в которой разворачиваются СИИО.
3. Разработан алгоритм автоматизации программирования взаимодействия агентов, обеспечивающий кодогенерацию программных механизмов информационно-управляемого взаимодействия по заданной онтологии информационного сервиса. Средняя доля автоматически сгенерированного программного кода для реализации интероперабельных структур модели данных и поведения агентов составляет 23,4%.
4. Предложен набор предметно-ориентированных моделей проектирования сервисов СИИО для востребованных приложений, что позволяет снизить трудозатраты на проектирование за счет предоставления разработчику архитектурных и поведенческих абстракций взаимодействия агентов при использовании предложенного метода.

5. Реализован комплекс программных средств в составе: а) генератор программного кода взаимодействия агентов; б) экспериментальные образцы предметно-ориентированных сервисов для вычислительных сред интеллектуального зала, умного музея и промышленного предприятия. На основе реализованного комплекса программных средств проведены экспериментальные исследования с целью оценки эффективности сформированного метода разработки программной инфраструктуры.

В качестве рекомендаций и перспектив дальнейшей разработки темы можно указать научное направление исследования самоорганизации агентов путем определения их функциональных ролей, моделей взаимодействия и операций, благодаря пониманию семантики процессов построения сервиса, а также сведений о доступных ресурсах.

Тема диссертации и отмеченные 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 научной специальности 05.13.11 — «Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей».

Список сокращений и условных обозначений

СИИО	совместно используемое информационное интернет-окружение
ИКТ	информационно-коммуникационные технологии
ПО	программное обеспечение
ИИ	искусственный интеллект
ИП	интеллектуальное пространство
ОИС	общее информационное содержимое
IoT	концепция «Интернет вещей» (англ. Internet of Things)
WoT	Веб вещей (англ. Web of Things)
SWoT	Семантический веб вещей (англ. Semantic Web of Things)
RDF	модель Семантического веба для представления данных и метаданных (Resource Description Framework)
RDF-тройка	утверждение о ресурсе вида: «субъект - предикат - объект»
SPARQL	язык запросов к данным, представленным с использованием модели RDF

Словарь терминов

В настоящей диссертационной работе применяют следующие термины с соответствующими определениями.

Ресурс — средство, обеспечивающее решение задач, возникающих при деятельности человека или группы людей.

Информационный ресурс — любой вид активного кода или программного обеспечения, а также любой элемент данных или цифровое представление чего-либо, использующиеся в информационных системах, которые важны с точки зрения удовлетворения информационных потребностей людей при решении задач определенной предметной области.

Технический ресурс — совокупность программных и аппаратных средств и сетевых коммуникаций для реализации определенной концепции вычислений в некоторой пространственно-ограниченной области.

Вычислительная среда — совокупность программных и аппаратных средств и сетевых коммуникаций для реализации определенной концепции в некоторой пространственно-ограниченной области.

Информационное окружение — вычислительная среда, оснащенная развитой информационно-коммуникационной инфраструктурой, предназначенной для автоматизации происходящих в ней процессов, направленных на обеспечение информационной поддержки людей во время решения задач определенной предметной области.

Совместно используемое информационное интернет-окружение — технологически оснащенное информационное окружение, предоставляющее множеству пользователей набор цифровых сервисов с доступом к сети Интернет для обеспечения возможности взаимодействовать друг с другом посредством совместного использования доступных ресурсов на основе информационного обмена при решении общей задачи.

Совместная деятельность на базе ИКТ — междисциплинарная область исследования, ориентированная на изучение характеристик и особенностей совместной деятельности людей с целью разработки продвинутых информационно-коммуникационных технологий для поддержки такой деятельности.

Интеллектуальность — способность вычислительной среды действовать так, как будто ее восприятие окружающего контекста дублирует восприятие

пользователя, а выполняемые средой действия при этом происходят незаметно (неотличимы от самой среды) для пользователя.

Агент — автономная программа, которая способна воспринимать информацию об этой среде, интерпретировать ее изменения и выполнять действия, влияющие на среду.

Информационно-управляемое взаимодействие агентов — взаимодействие агентов, выполняемое в зависимости от информации, доступной в общем информационном содержимом.

Интеллектуальное пространство — цифровая информационно-вычислительная среда, которая способна обеспечивать находящееся в ней динамическое множество участников контекстно-зависимыми сервисами тогда, когда они требуются и, по возможности, упреждающим (проактивным) образом.

Промежуточное программное обеспечение — программный слой, расположенный между системным программным обеспечением и прикладными системами, который направлен на поддержку требований для организации интеллектуальных пространств. Программное обеспечение, предназначенное для интеграции разнообразных вычислительных устройств и программных компонентов и организации их сетевого взаимодействия.

Сервис — система взаимодействующих программных агентов (распределенная динамическая неоднородная система) для реализации предписанной функции с привлечением доступных ресурсов. Программная система, доступ к которой предоставляется посредством заданного интерфейса и осуществляется в соответствии с указанными в описании сервиса ограничениями и политиками, а выполнение следует заданному набору процессов (совокупность действий, повторяемых во времени).

Программная инфраструктура — средства, которые делают возможным функционирование сервисов. Включает в себя программно-аппаратное обеспечение, вычислительную среду, а также следующие компоненты: а) промежуточное программное обеспечение для обеспечения информационно-управляемого взаимодействия агентов; б) агенты для совместного построения и предоставления сервисов; в) системное программное обеспечение для запуска и обеспечения функционирования других компонент.

Список литературы

1. *Ариарский*, *М. А.* Теория социально-культурной деятельности отвечает на вызовы времени / М. А. Ариарский // Вестник Санкт-Петербургского государственного института культуры. — 2013. — Т. 2, № 15. — С. 38—43.
2. *Беляева*, *А. В.* Компьютерно-опосредствованная совместная деятельность и проблема психического развития / А. В. Беляева, М. Коул // Психологический журнал. — 1991. — Т. 12, № 2. — С. 145—152.
3. *Воронина*, *Т.* Управление инновациями в сфере образования / Т. Воронина, О. Молчанова, А. Абрамешин // Высшее образование в России. — 2001. — № 6. — С. 3—12.
4. *Герасимова*, *И. А.* Тенденции развития науки и образования в аспекте глобализации / И. А. Герасимова, Е. В. Грибова // Культура, личность, общество в современном мире: методология, опыт эмпирического исследования. — 2013. — С. 903—913.
5. *Городецкий*, *В. И.* Самоорганизация и многоагентные системы. I. Модели многоагентной самоорганизации / В. И. Городецкий // Известия Российской академии наук. Теория и системы управления. — 2012. — № 2. — С. 92—92.
6. *Городецкий*, *В. И.* Базовая онтология коллективного поведения автономных агентов и ее расширения / В. И. Городецкий, В. В. Самойлов, Д. В. Троцкий // Известия Российской академии наук. Теория и системы управления. — 2015. — № 5. — С. 102—102.
7. *Городецкий*, *В. И.* Многоагентные технологии для индустриальных приложений: реальность и перспектива / В. И. Городецкий, П. О. Скобелев // Труды СПИИРАН. — 2017. — Т. 55, № 0. — С. 11—45.
8. *Горчакова*, *Е. Н.* Качество, результативность, эффективность, квалитативность: терминологические аспекты / Е. Н. Горчакова, Ф. Е. Поклонский // Экономика промышленности. — 2009. — Т. 1, № 44.
9. *Денисюк*, *А. В.* Определение адекватной модели для представления знаний в современных информационных системах / А. В. Денисюк, В. С. Кавицкая, В. В. Любченко. — 2014.

10. Интеллектуальная система автоматизированного проведения конференций / А. М. Кашевник [и др.] // Труды СПИИРАН. — 2010. — Т. 14. — С. 228—243.

11. Интеллектуальная система тематического исследования научно-технической информации (ИСТИНА) / В. А. Васенин [и др.] // Информационное общество. — 2013. — № 1/2. — С. 21—36.

12. *Кадиров*, *Н. Т.* Краткая история и классификация программного обеспечения совместной работы / Н. Т. Кадиров // Молодой ученый. — 2017. — № 6. — С. 20—23.

13. *Калюжный*, *К. А.* Информационная среда и информационная среда науки: сущность и назначение / К. А. Калюжный // Наука. Инновации. Образование. — 2015. — № 18. — С. 7—23.

14. *Камнева*, *В. В.* Цифровая экономика в образовании / В. В. Камнева, Е. А. Коняева // Вопросы студенческой науки. — 2018. — Т. 3, № 19. — С. 101—105.

15. *Корзун*, *Д. Ж.* Формализм сервисов и архитектурные абстракции для программных приложений интеллектуальных пространств / Д. Ж. Корзун // Программная инженерия. — 2015. — № 2. — С. 3—12.

16. *Корзун*, *Д. Ж.* Операция подписки для приложений в интеллектуальных пространствах платформы Smart-M3 / Д. Ж. Корзун, А. А. Ломов // Труды СПИИРАН. — 2012. — Т. 23. — С. 439—458.

17. *Корзун*, *Д. Ж.* Автоматизированная модельно-ориентированная разработка программных агентов для интеллектуальных пространств на платформе Smart-M3 / Д. Ж. Корзун, А. А. Ломов, П. И. Ванаг // Программная инженерия. — 2012. — № 5. — С. 6—14.

18. *Корзун*, *Д. Ж.* Сервис построения культурной программы для участников конференции в интеллектуальном зале / Д. Ж. Корзун, С. А. Марченков, А. С. Вдовенко // Материалы научно-практической конференции студентов, аспирантов и молодых учёных Современные технологии в теории и практике программирования. — СПб : ИД "Политехнический университет", 04.2015. — С. 30—31.

19. *Кринкин*, *К. В.* Использование географического контекста в интеллектуальных пространствах (Smart spaces) / К. В. Кринкин, К. Г. Юденок // Известия Санкт-Петербургского государственного электротехнического университета ЛЭТИ. — 2014. — № 2. — С. 6—11.

20. *Куприянов*, *А. А.* Аспекты интероперабельности автоматизированных систем / А. А. Куприянов // Автоматизация процессов управления. — 2009. — № 4. — С. 40—49.

21. *Марченков*, *С. А.* Автоматизация процессов программирования агентов на основе кодогенерации при построении семантических сервисов интеллектуальных пространств. Часть 1 / С. А. Марченков // Программная инженерия. — 2019. — Т. 10, № 6. — С. 257—264.

22. *Марченков*, *С. А.* Автоматизация процессов программирования агентов на основе кодогенерации при построении семантических сервисов интеллектуальных пространств. Часть 2 / С. А. Марченков // Программная инженерия. — 2019. — Т. 10. — Принято к печати.

23. *Марченков*, *С. А.* Расширение возможностей совместной деятельности в интеллектуальном зале на основе сервисов электронного туризма / С. А. Марченков, А. С. Вдовенко, Д. Ж. Корзун // Труды СПИИРАН. — 2017. — Т. 1, № 50. — С. 165—189.

24. *Марченков*, *С. А.* Определение присутствия пользователей в интеллектуальном зале на основе отслеживания активности в беспроводной сети / С. А. Марченков, Д. Ж. Корзун // Ученые записки Петрозаводского государственного университета. Сер.: Физико-математические науки. — 2015. — № 2. — С. 114—119.

25. *Марченков*, *С. А.* Разработка инфраструктурных сервисов для системы интеллектуального зала / С. А. Марченков, Д. Ж. Корзун // Научно-исследовательская работа обучающихся и молодых ученых : материалы 67-й Всероссийской (с международным участием) научной конференции обучающихся и молодых ученых [Электронный ресурс]. — Петрозаводск : Издательство ПетрГУ, 04.2015. — С. 239—242.

26. *Марченков, С. А.* Разработка программной инфраструктуры для сервисно-ориентированных систем совместной деятельности / С. А. Марченков, Д. Ж. Корзун // Материалы научно-практической конференции студентов, аспирантов и молодых учёных Современные технологии в теории и практике программирования. — СПб. : ИД ”Политехнический университет”, 04.2016. — С. 21—23.

27. Методы и средства автоматизированного проектирования прикладной онтологии / Б. В. Добров [и др.] // Известия Российской академии наук. Теория и системы управления. — 2004. — Т. 2. — С. 58—68.

28. *Намиот, Д. Е.* Метаданные в REST-модели / Д. Е. Намиот // Программная инженерия. — 2015. — № 5. — С. 20—25.

29. *Намиот, Д. Е.* О стандартах Умного Города / Д. Е. Намиот // Информационное общество. — 2017. — № 2. — С. 45—52.

30. *Паньшин, Б.* Цифровая экономика: особенности и тенденции развития / Б. Паньшин // Наука и инновации. — 2016. — Т. 3, № 157.

31. Правительство, промышленность, логистика, инновации и интеллектуальная мобильность в цифровой экономике / В. П. Куприяновский [и др.] // Современные информационные технологии и ИТ-образование. — 2017. — Т. 13, № 1. — С. 74—96.

32. *Разумовский, А. Г.* Валидация объектно-ориентированных программ с использованием онтологии / А. Г. Разумовский, М. Г. Пантелеев // Программная инженерия. — 2012. — Т. 7. — С. 7.

33. *Розина, И. Н.* Педагогическая компьютерно-опосредованная коммуникация как прикладная область коммуникативных исследований / И. Н. Розина // Образовательные технологии и общество. — 2005. — Т. 8, № 2. — С. 257—265.

34. *Тарасов, В. Б.* От многоагентных систем к интеллектуальным организациям: философия, психология, информатика / В. Б. Тарасов. — Москва : Эдиториал УРСС, 2002.

35. *Тесля, Н. Н.* Принципы построения интеллектуальных транспортных систем для обеспечения инфомобильности / Н. Н. Тесля // Труды СПИИРАН. — 2014. — Т. 37, № 0. — С. 21—36.

36. *Хорошевский, В. Ф.* Пространства знаний в сети Интернет и Semantic Web (Часть 1) / В. Ф. Хорошевский // Искусственный интеллект и принятие решений. — 2008. — № 1. — С. 80—97.

37. *Черняк, Л.* Долгий путь к социальному программному обеспечению / Л. Черняк // Computerworld Россия. — 2007.

38. *Шлыкова, О. В.* Влияние информационно-коммуникационных технологий на социокультурную среду региона (по материалам исследования) / О. В. Шлыкова // Вестник Московского государственного университета культуры и искусств. — 2014. — Т. 2, № 58. — С. 181—188.

39. *Юсупов, Р.* От умных приборов к интеллектуальному пространству / Р. Юсупов, А. Л. Ронжин // Вестник Российской академии наук. — 2010. — Т. 80, № 1. — С. 45—51.

40. *Юсупов, Р.* От умных приборов к интеллектуальному пространству / Р. Юсупов, А. Л. Ронжин // Вестник Российской академии наук. — 2010. — Т. 80, № 1. — С. 45—51.

41. A comprehensive framework for modeling requirements of CSCW systems / M. A. Teruel [et al.] // Journal of Software: Evolution and Process. — 2017. — Vol. 29, no. 5. — e1858.

42. A free and open source Java framework for building Semantic Web and Linked Data applications [Electronic Resource]. — URL: https://jena.apache.org/ (visited on 09/10/2019).

43. A Semantic Approach to Designing Information Services for Smart Museums / D. Korzun [et al.] // International Journal of Embedded and Real-Time Communication Systems (IJERTCS). — 2016. — Vol. 7, no. 2. — P. 15—34.

44. A Smart Space-Based Design of Semantic Layer for Advancing Museum Information Services / S. Marchenkov [et al.] // Proc. 19th Conf. of Open Innovations Association FRUCT / ed. by S. Balandin, T. Tyutina. — Jyvaskyla, Finland, 11/2016. — P. 159—166.

45. A web of things approach for indoor position monitoring of elderly and impaired people / F. Antoniazzi [et al.] // 21st Conference of Open Innovations Association (FRUCT). — 11/2017. — P. 51—56.

46. *Ahmed*, *T.* Security policies in distributed CSCW and workflow systems / T. Ahmed, A. R. Tripathi // IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans. — 2010. — Vol. 40, no. 6. — P. 1220—1231.

47. Anonymous agent coordination in smart spaces: State-of-the-art / A. Smirnov [et al.] // Smart Spaces and Next Generation Wired/Wireless Networking. — Springer, 2009. — P. 42—51.

48. Application layer protocols for the Internet of Things: A survey / M. B. Yassein, M. Q. Shatnawi, [et al.] // 2016 International Conference on Engineering & MIS (ICEMIS). — IEEE. 2016. — P. 1—4.

49. Applying the 3C model to groupware development / H. Fuks [et al.] // International Journal of Cooperative Information Systems. — 2005. — Vol. 14, 02n03. — P. 299—328.

50. *Assel*, *M.* A security framework for dynamic collaborative working environments / M. Assel, S. Wesner, A. Kipp // Identity in the Information Society. — 2009. — Dec. — Vol. 2, no. 2. — P. 171—187.

51. *Batanov*, *D. N.* Using Ontologies to Create Object Model for Object-Oriented Software Engineering / D. N. Batanov, W. Vongdoiwang // Ontologies: A Handbook of Principles, Concepts and Applications in Information Systems / ed. by R. Sharman, R. Kishore, R. Ramesh. — Boston, MA : Springer US, 2007. — P. 461—487.

52. *Chen*, *E. T.* The internet of things: Opportunities, issues, and challenges / E. T. Chen // The Internet of Things in the Modern Business Environment. — 2017. — P. 167—187.

53. *Clay*, *A.* An Overview of Smartrooms and Collaborative Work Environments. From Research Issues to User Acceptance in the Oil and Gas Industry / A. Clay // Proceeding of ERGO'IA 2018. — Bidart, France, 10/2018.

54. Creating context-aware collaborative working environments / M. A. Martinez-Carreras [et al.] // International Journal on Artificial Intelligence Tools. — 2011. — Vol. 20, no. 1. — P. 195—207.

55. CSCW tools: concepts and architectures / W. Reinhard [et al.] // Computer. — 1994. — May. — Vol. 27, no. 5. — P. 28—36.

56. CSCW—What Does It Mean? (Panel Session) / L. Bannon [et al.] // Proceedings of the 1988 ACM Conference on Computer-supported Cooperative Work. — Portland, Oregon, USA : ACM, 1988. — P. 191—192. — (CSCW ’88).

57. CuteSIB Demo for Raspberry Pi / S. Marchenkov [et al.] // Proc. 18th Conf. Open Innovations Framework Program FRUCT. — S.-Petersburg, Russia : ITMO Univeristy, 04/2016. — P. 545—545.

58. DBpedia: Online Access [Electronic Resource]. — URL: https://wiki.dbpedia.org/OnlineAccess (visited on 11/09/2019).

59. Designing a Generic Collaborative Working Environment / M. A. Martinez-Carreras [et al.] // IEEE International Conference on Web Services (ICWS 2007). — 07/2007. — P. 1080—1087.

60. *Dissanaike*, *S.* Utilizing XML-RPC or SOAP on an embedded system / S. Dissanaike, P. Wijkman, M. Wijkman // 24th International Conference on Distributed Computing Systems Workshops, 2004. Proceedings. — IEEE. 2004. — P. 438—440.

61. *Eddine*, *M. M. C.* An agent based approach for modeling a groupware / M. M. C. Eddine, K. Okba // Multiagent and Grid Systems. — 2016. — Vol. 12, no. 3. — P. 199—215.

62. *Evermann*, *J.* Ontology based object-oriented domain modelling: fundamental concepts / J. Evermann, Y. Wand // Requirements engineering. — 2005. — Vol. 10, no. 2. — P. 146—160.

63. From the Internet of Things to the Web of Things: Resource-oriented Architecture and Best Practices / D. Guinard [et al.] // Architecting the Internet of Things / ed. by D. Uckelmann, M. Harrison, F. Michahelles. — Berlin, Heidelberg : Springer Berlin Heidelberg, 2011. — P. 97—129.

64. *Galov*, *I.* Design of Semantic Information Broker for Localized Computing Environments in the Internet of Things / I. Galov, A. Lomov, D. Korzun // Proc. 17th Conf. of Open Innovations Association FRUCT. — Yaroslavl, Russia : IEEE, 04/2015. — P. 36—43.

65. *Gazis*, *V.* A Survey of Standards for Machine-to-Machine and the Internet of Things / V. Gazis // IEEE Communications Surveys Tutorials. — 2017. — Vol. 19, no. 1. — P. 482—511.

66. *Grudin*, *J.* Computer-supported cooperative work: history and focus / J. Grudin // Computer. — 1994. — May. — Vol. 27, no. 5. — P. 19—26.

67. *Gutwin*, *C.* A framework of awareness for small groups in shared-workspace groupware : tech. rep. / C. Gutwin, S. Greenberg ; Department of Computer Science, University of Saskatchewan. — 1999. — No. 99—1.

68. *Gyrard*, *A.* Semantic web methodologies, best practices and ontology engineering applied to Internet of Things / A. Gyrard, M. Serrano, G. A. Atemezing // 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT). — 2015. — P. 412—417.

69. IoT Middleware: A Survey on Issues and Enabling Technologies / A. H. Ngu [et al.] // IEEE Internet of Things Journal. — 2017. — Feb. — Vol. 4, no. 1. — P. 1—20.

70. IoT-Lite: a lightweight semantic model for the internet of things and its use with dynamic semantics / M. Bermudez-Edo [et al.] // Personal and Ubiquitous Computing. — 2017. — June. — Vol. 21, no. 3. — P. 475—487.

71. *Iqbal*, *R.* A collaborative platform for heterogeneous CSCW systems: case study of academic applications / R. Iqbal, A. James, R. Gatward // Second IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 2003. Proceedings. — 09/2003. — P. 462—467.

72. *Järvinen*, *H.* Semantic Interoperability for Web Services based Smart Home Systems / H. Järvinen, P. Vuorimaa // Proceedings of the International Conference on Agents and Artificial Intelligence-Volume 1. — SCITEPRESS-Science, Technology Publications, Lda. 2015. — P. 141—148.

73. *Jeners*, *N.* A meta-model for cooperation systems / N. Jeners, W. Prinz, S. Franken // Working Conference on Virtual Enterprises. — Springer. 2013. — P. 239—246.

74. *Johansen*, *R.* Teams for tomorrow (groupware) / R. Johansen // Proceedings of the Twenty-Fourth Annual Hawaii International Conference on System Sciences. Vol. 3. — 01/1991. — P. 521—534.

75. *Johansen*, *R.* GroupWare: Computer Support for Business Teams / R. Johansen. — New York, NY, USA : The Free Press, 1988.

76. *Kling*, *R.* Cooperation, Coordination and Control in Computer-supported Work / R. Kling // Commun. ACM. — New York, NY, USA, 1991. — Dec. — Vol. 34, no. 12. — P. 83—88.

77. *Knublauch*, *H.* Ontology-driven software development in the context of the semantic web: An example scenario with Protege/OWL / H. Knublauch // 1st International workshop on the model-driven semantic web (MDSW2004). — Monterey, California, USA, 2004. — P. 381—401.

78. *Koch*, *M.* Computer-Supported Cooperative Work - Concepts and Trends / M. Koch, T. Gross // In: Proc. Conf. of the Association Information And Management (AIM), Lecture Notes in Informatics (LNI) P-92. — Koellen Verlag, 2006.

79. *Korzun*, *D. G.* Deployment of Smart Spaces in Internet of Things: Overview of the design challenges / D. G. Korzun, S. I. Balandin, A. V. Gurtov // Internet of Things, Smart Spaces, and Next Generation Networking. — Springer, 2013. — P. 48—59.

80. *Křemen*, *P.* Ontology-Driven Information System Design / P. Křemen, Z. Kouba // IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews). — 2012. — May. — Vol. 42, no. 3. — P. 334—344.

81. *Kujur*, *P.* Evolution of World Wide Web: Journey From Web 1.0 to Web 4.0 / P. Kujur, B. Chhetri // International Journal of Computer Science and Technology. — 2015. — Jan. — Vol. 6, issue 1. — P. 134—138.

82. *Kulakov*, *K.* An Approach to Generating Ontology-Based Object Model for Smart-M3 / K. Kulakov, S. Marchenkov, S. Tishkov // Proceedings of the 24th Conference of Open Innovations Association FRUCT. — Moscow, Russia : FRUCT Oy, 04/2019. — P. 670—676.

83. *Lee*, *C. P.* From The Matrix to a Model of Coordinated Action (MoCA): A Conceptual Framework of and for CSCW / C. P. Lee, D. Paine // Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing. — Vancouver, BC, Canada : ACM, 2015. — P. 179—194. — (CSCW ’15).

84. *Marchenkov*, *S. A.* Smart-M3 CuteSIB Demo for a Wireless Router with OpenWrt-Based Firmware / S. A. Marchenkov, D. E. Baganov, D. G. Korzun // Proc. 20th Conf. of Open Innovations Association FRUCT / ed. by S. Balandin, A. Levina, T. Tyutina. — Saint-Petersburg, Russia, 04/2017. — P. 634—638.

85. *Marchenkov*, *S.* User Presence Detection Based on Tracking Network Activity in SmartRoom / S. Marchenkov, D. Korzun // Proc. 16th Conf. of Open Innovations Association FRUCT. — Oulu, Finland, 10/2014. — P. 45—50.

86. *Marchenkov*, *S.* Enhancing the opportunities of collaborative work in an intelligent room using e-tourism services / S. Marchenkov, A. Vdovenko, D. Korzun // Trudy SPIIRAN. — 2017. — Vol. 1, no. 50. — P. 165—189.

87. Middleware for Internet of Things: A Survey / M. A. Razzaque [et al.] // IEEE Internet of Things Journal. — 2016. — Feb. — Vol. 3, no. 1. — P. 70—95.

88. *Mohammed*, *F. H.* Survey on IoT Services: Classifications and Applications / F. H. Mohammed, R. Esmail // International Journal of Science and Research (IJSR). — 2015. — Vol. 4, no. 1. — P. 2124—2127.

89. *Niemantsverdriet*, *K.* A Perspective on Multi-user Interaction Design Based on an Understanding of Domestic Lighting Conflicts / K. Niemantsverdriet, H. Essen, B. Eggen // Personal Ubiquitous Comput. — 2017. — Apr. — Vol. 21, no. 2. — P. 371—389.

90. *Odell*, *J. J.* Representing Agent Interaction Protocols in UML / J. J. Odell, H. Van Dyke Parunak, B. Bauer // Agent-Oriented Software Engineering / ed. by P. Ciancarini, M. J. Wooldridge. — Berlin, Heidelberg" : Springer Berlin Heidelberg, 2001. — P. 121—140.

91. On applicability of wireless routers to deployment of smart spaces in Internet of Things environments / S. Marchenkov [et al.] // Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS). Vol. 2. — Bucharest, Romania : IEEE, 09/2017. — P. 1000—1005.

92. On IoT-services: Survey, Classification and Enterprise Integration / M. Thoma [et al.] // Green Computing and Communications (GreenCom), 2012 IEEE International Conference on (Besancon, France). — IEEE, 11/2012. — P. 257—260.

93. OpenIoT: Open Source Internet-of-Things in the Cloud / J. Soldatos [et al.] // Interoperability and Open-Source Solutions for the Internet of Things / ed. by I. Podnar Žarko, K. Pripužić, M. Serrano. — Cham : Springer International Publishing, 2015. — P. 13—25.

94. OWL-S: Semantic Markup for Web Services [Electronic Resource]. — 2004. — URL: https://www.w3.org/Submission/OWL-S/ (visited on 02/07/2019).

95. Pellet: A practical OWL-DL reasoner / E. Sirin [et al.] // Journal of Web Semantics. — 2007. — Vol. 5, no. 2. — P. 51—53.

96. Performance Evaluation of Smart-M3 Applications: A SmartRoom Case Study / D. Korzun [et al.] // Proc. 18th Conf. of Open Innovations Association FRUCT / ed. by S. Balandin, T. Tyutina, A. Levina. — St. Petersburg, Russia : ITMO University, 04/2016. — P. 138—144.

97. *Pinelle*, *D.* Task analysis for groupware usability evaluation: Modeling shared-workspace tasks with the mechanics of collaboration / D. Pinelle, C. Gutwin, S. Greenberg // ACM Transactions on Computer-Human Interaction (TOCHI). — 2003. — Vol. 10, no. 4. — P. 281—311.

98. *Rich*, *C.* Approaches to automatic programming / C. Rich, R. C. Waters // Advances in computers. — 1993. — Vol. 37. — P. 1—57.

99. *Rodden*, *T.* A survey of CSCW systems / T. Rodden // Interacting with computers. — 1991. — Vol. 3, no. 3. — P. 319—353.

100. *Rong*, *W.* A Multi-agent Architecture for CSCW Systems: From Organizational Semiotics Perspective / W. Rong, K. Liu // Agent Computing and Multi-Agent Systems / ed. by Z.-Z. Shi, R. Sadananda. — Berlin, Heidelberg : Springer Berlin Heidelberg, 2006. — P. 766—772.

101. *Ryu*, *M.* Integrated semantics service platform for the Internet of Things: A case study of a smart office / M. Ryu, J. Kim, J. Yun // Sensors. — 2015. — Vol. 15, no. 1. — P. 2137—2160.

102. Sawsdl: Semantic annotations for wsdl and xml schema / J. Kopecký [et al.] // IEEE Internet Computing. — 2007. — Vol. 11, no. 6. — P. 60—67.

103. *Schmidt*, *K.* Taking CSCW seriously / K. Schmidt, L. Bannon // Computer Supported Cooperative Work (CSCW). — 1992. — Mar. — Vol. 1, no. 1. — P. 7—40.

104. Semantic web of things: an analysis of the application semantics for the iot moving towards the iot convergence / A. J. Jara [et al.] // International Journal of Web and Grid Services. — 2014. — Vol. 10, no. 2/3. — P. 244—272.

105. Semantics for the Internet of Things: early progress and back to the future / P. Barnaghi [et al.] // International Journal on Semantic Web and Information Systems (IJSWIS). — 2012. — Vol. 8, no. 1. — P. 1—121.

106. Service modelling for the Internet of Things / S. De [et al.] // 2011 Federated Conference on Computer Science and Information Systems (FedCSIS) (Szczecin, Poland). — IEEE, 09/2011. — P. 949—955.

107. SIOC project: an ontology of terms that can be used to describe online communities on the Web of Data [Electronic Resource]. — URL: http://sioc-project.org/ (visited on 09/23/2019).

108. *Siricharoen*, *W. V.* Ontologies and object models in object oriented software engineering / W. V. Siricharoen // IAENG International Journal of Computer Science. — 2007. — Vol. 33, no. 1. — P. 19—24.

109. Smart Museum of Everyday Life History in Petrozavodsk State University: Software Design and Implementation of the Semantic Layer / S. Marchenkov [et al.] // Proc. 21st Conf. of Open Innovations Association FRUCT. — Helsinki, Finland, 11/2017. — P. 224—230.

110. Smart Spaces-Based Application Development: M3 Architecture, Design Principles, Use Cases, and Evaluation / D. G. Korzun [et al.] // International Journal of Embedded and Real-Time Communication Systems (IJERTCS). — 2017. — Vol. 8, no. 2. — P. 66—100.

111. Smart-M3 Information Sharing Platform / J. Honkola [et al.] // Proc. IEEE Symp. Computers and Communications (ISCC'10). — Riccione, Italy : IEEE Computer Society, 06/2010. — P. 1041—1046.

112. Soupa: Standard ontology for ubiquitous and pervasive applications / H. Chen [et al.] // The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004. — IEEE. 2004. — P. 258—267.

113. *Stankovic*, *M.* Modeling online presence / M. Stankovic // J. Breslin, U. Bojars, A. Passant and S. Fernandez, editeurs, Proceedings of the First Social Data on the Web Workshop, Karlsruhe, Germany. Vol. 405. — 2008. — P. 58.

114. The Friend of a Friend (FOAF) project [Electronic Resource]. — 09/2011. — URL: http://www.foaf-project.org/ (visited on 09/23/2019).

115. The M3 Architecture for Smart Spaces: Overview of Semantic Information Broker Implementations / F. Viola [et al.] // Proc. of the 19th Conference of Open Innovations Association FRUCT / ed. by S. Balandin, T. Tyutina. — Jyvaskyla, Finland : IEEE, 11/2016. — P. 264—272.

116. The OWL-S Editor – A Development Tool for Semantic Web Services / D. Elenius [et al.] // The Semantic Web: Research and Applications / ed. by A. Gómez-Pérez, J. Euzenat. — Berlin, Heidelberg : Springer Berlin Heidelberg, 2005. — P. 78—92.

117. Towards a semantic-aware collaborative working environment / M. A. Martinez Carreras [et al.] // Computing and Informatics. — 2013. — Vol. 30, no. 1. — P. 7—30.

118. Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI / F. Curbera [et al.] // IEEE Internet computing. — 2002. — Vol. 6, no. 2. — P. 86—93.

119. *Vdovenko*, *A.* Enhancing the SmartRoom System with e-Tourism Services / A. Vdovenko, S. Marchenkov, D. Korzun // Proc. 17th Conf. Open Innovations Framework Program FRUCT. — Yaroslavl, Finland, 04/2015. — P. 237—246.

120. *Vdovenko*, *A.* Mobile Multi-Service Smart Room Client: Initial Study for Multi-Platform Development / A. Vdovenko, S. Marchenkov, 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, U. Trifonova. — Petrozavodsk, Russia : SUAI, 04/2013. — P. 143—152.

121. Virtual Shared Workspace for Smart Spaces and M3-based Case Study / D. Korzun [et al.] // Proc. 15th Conf. of Open Innovations Association FRUCT / ed. by S. Balandin, U. Trifonova. — St. Petersburg, Russia : ITMO University, 04/2014. — P. 60—68.

122. *Yang*, *C. W.* Ontology driven approach to generate distributed automation control from substation automation design / C. W. Yang, V. Dubinin, V. Vyatkin // IEEE Transactions on Industrial Informatics. — 2017. — Feb. — Vol. 13, no. 2. — P. 668—679.

123. *Yu*, *H.* From internet of things to internet of agents / H. Yu, Z. Shen, C. Leung // 2013 IEEE international conference on green computing and communications and IEEE Internet of Things and IEEE cyber, physical and social computing. — IEEE. 2013. — P. 1054—1057.

124. *Zeng*, *D.* The web of things: A survey / D. Zeng, S. Guo, Z. Cheng // JCM. — 2011. — Vol. 6, no. 6. — P. 424—438.

125. *Zhang*, *W.* Exploring Semantic Web technologies for ontology-based modeling in collaborative engineering design / W. Zhang, J. Yin // The International Journal of Advanced Manufacturing Technology. — 2008. — Vol. 36, no. 9/10. — P. 833—843.

Приложение А

Акты внедрения

1) Акт о внедрении результатов диссертационной работы в учебный процесс кафедры информатики и математического обеспечения, ПетрГУ.

Министерство науки и высшего образования Российской Федерации
федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕТРОЗАВОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ПетрГУ)

Ленина пр., д. 33, Петрозаводск, Республика Карелия, 185910
тел. (814 2) 78-51-40, 71-10-29, факс: (814 2) 71-10-00
E-mail: rectorat@petrsu.ru, office@petrsu.ru, https:// petrsu.ru

ОКПО 02069533, ОГРН 1021000519935, ИНН/КПП 1021/0487874/101001

_____ № _____

На № _____ от _____

"УТВЕРЖДАЮ"
проректор по учебной работе,
_____ К.Г. Тарасов
_____ сентября 2019 г.

АКТ
о внедрении результатов диссертационной работы
на соискание ученой степени
кандидата технических наук
Марченкова Сергея Александровича

Комиссия в составе: председателя Н.Ю. Световой, директора института математики и информационных технологий, и членов Е.Е. Семеновой, председателя учебно-методической комиссии института математики и информационных технологий, Ю.А. Богоявленского, заведующего кафедрой информатики и математического обеспечения, и Д.Ж. Корзуна, доцента кафедры информатики и математического обеспечения,
составила настоящий акт, подтверждающий, что результаты диссертационной работы С.А Марченкова «**Автоматизированная разработка интероперабельной программной инфраструктуры для организации совместно используемого информационного интернет-окружения**» используются в учебном процессе кафедры информатики и математического обеспечения федерального государственного бюджетного образовательного учреждения высшего образования «Петрозаводский государственный университет».

Основные результаты диссертационного исследования используются в дисциплине «Интеллектуальные сетевые пространства», начиная с 2016/2017 уч. года. Дисциплина входит в вариативную часть учебного плана основной образовательной программы магистратуры по направлениям подготовки «Прикладная математика и информатика» и «Информационные системы и технологии», являясь дисциплиной по выбору.

Следующие результаты исследования используются при решении обучающимися задач в рамках данной дисциплины.
1. Предлагаемый метод разработки программной инфраструктуры для организации совместно используемого информационного интернет-окружения используется обучающимися для унификации процесса разработки сервис-ориентированных приложений интеллектуальных пространств с поддержкой автоматизации программирования. Совместно используемые информационные интернет-окружения определяют рекомендуемые классы приложений для реализации полученных в рамках

выбранного проекта теоретических решений. Метод предлагает набор этапов разработки, следуя которым, обучающиеся создают архитектурные решения и онтологические модели представления знаний для реализации программного прототипа своего проекта.
2. Полученная программная реализация (с открытым исходным кодом) генератора программного кода агентов позволяет увеличить объем генерируемого кода для реализации сервисов и освободить обучающихся от создания вручную значительного объема рутинного кода. Это позволяет обучающимся при создании своего программного прототипа сосредоточиться на творческих, интеллектуальных заданиях, направленных на разработку архитектурных решений, моделей данных и сценариев поведения агентов.

Использование указанных результатов в учебном процессе позволило обеспечить студентов современными знаниями о моделях проектирования сервисов и способах автоматизации программирования на основе онтологических моделях в области программной разработки совместно используемых информационных интернет-окружений.

Председатель:
директор института математики
и информационных технологий,
доцент, к.ф.-м.н. Н.Ю. Светова

Члены комиссии:

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

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

доцент каф. информатики
и математического обеспечения
к.ф.-м.н. Д.Ж. Корзун

2) Акт о внедрении результатов кандидатской диссертационной работы в рамках коммерциализации результатов научно-технической деятельности проекта № 14.574.21.0060, выполняемой ООО «Опти-Софт».

ОБЩЕСТВО С ОГРАНИЧЕННОЙ ОТВЕТСТВЕННОСТЬЮ "ОПТИ-СОФТ"
185035, Республика Карелия, г. Петрозаводск, проспект Ленина (Центр Р-Н), д. 31, каб. 304
тел. +7 (814 2) 71-32-10, +7 (814 2) 71-32-22, факс: +7 (814 2) 71-32-16
e-mail: ashabaev@opti-soft.ru
ОКПО 6568922, ОГРН 1101001003641, ИНН/КПП 1001232729/100101001

А К Т

о внедрении результатов диссертационной работы
на соискание ученой степени
кандидата технических наук
Марченкова Сергея Александровича

Настоящий акт составлен о том, что результаты диссертационной работы **«Автоматизированная разработка интероперабельной программной инфраструктуры для организации совместно используемого информационного интернет-окружения»** использованы в рамках коммерциализации результатов выполнения проекта от 30 июня 2014 г № 14.574.21.0060 «Разработка технологии интеллектуализации локализованных вычислительных сред Интернета физических устройств для персонализированного построения и упреждающей доставки сервисов».

Результаты интеллектуальной деятельности проекта, полученные на основе представленного С. А. Марченковым *метода разработки программной инфраструктуры для организации совместно используемого интернет-окружения* (в частности, «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»: свидетельство государственной регистрации программы для ЭВМ № 2015615091), используются для производства и развития высокотехнологической продукции в виде цифровых сервисов при разработке следующего программного обеспечения: 1) программное обеспечение информационно-управляемого взаимодействия в условиях Интернета вещей и больших данных, 2) программные модули информационного поиска и вывода знаний на основе данных мониторинга параметров здоровья человека и контекста ситуации.

Использование указанного метода позволило:
а) обеспечить семантическую интероперабельность программных агентов при реализации программного обеспечения и развитии цифровых сервисов для привлечения разнообразных ресурсов;
б) унифицировать процесс проектирования цифровых сервисов как систем взаимодействующих программных агентов;
в) снизить трудозатраты на разработку цифровых сервисов за счет предоставления разработчику архитектурных и поведенческих абстракций взаимодействия агентов;
г) увеличить объем автоматически генерируемого программного кода для реализации структур модели данных и поведения агентов при программировании цифровых сервисов.

Директор ООО «Опти-Софт», к.т.н. А. И. Шабаев
"22" июля 2019 г. М.П.

3) Акт об использовании результатов кандидатской диссертационной работы в исследованиях, выполняемых ООО «ЦМИТ» при разработке ПАК для мониторинга роботизированного производственного оборудования.

ООО «ЦЕНТР МОЛОДЕЖНОГО ИННОВАЦИОННОГО ТВОРЧЕСТВА»
ИНН 1001315164, КПП 100101001
185035, Петрозаводск, пр. Ленина, 31, каб. 105

А К Т

об использовании результатов диссертационной работы
на соискание ученой степени
кандидата технических наук
Марченкова Сергея Александровича

Настоящим актом удостоверяется, что результаты диссертационной работы Марченкова С.А., выполненной по теме «**Автоматизированная разработка интероперабельной программной инфраструктуры для организации совместно используемого информационного интернет-окружения**», используются ООО «ЦМИТ» (Центр молодежного инновационного творчества, Петрозаводск, республика Карелия) в исследованиях, направленных на разработку программно-аппаратного комплекса многопараметрического мониторинга роботизированного производственного оборудования в рамках выполнения НИОКР проекта № 44683 (2018-2020 гг.).

Использование следующих результатов позволило повысить эффективность разработки многоагентной программной инфраструктуры для обеспечения мониторинга производственного оборудования и информационного сопровождения сотрудников:
1. Концептуальная модель информационного сервиса позволяет унифицировано проектировать сервисы как системы взаимодействующих агентов с привлечением производственных ресурсов, сотрудников и их контекстной информации.
2. Алгоритм автоматизации программирования взаимодействия агентов позволяет уменьшить количество создаваемого вручную рутинного программного кода для реализации модели данных и информационно-управляемого взаимодействия агентов при программировании сервисов мониторинга физического оборудования.
3. Предметно-ориентированная модель проектирования для сервисов мониторинга физической среды позволяет снизить трудозатраты при разработке за счет использования готовых архитектурных и поведенческих абстракций взаимодействия агентов.

Полученные модели проектирования и алгоритм позволяют создавать на их основе научно-технические решения, обеспечивающие активное вовлечение информационных и технических ресурсов, а также ресурсов людей-сотрудников при решении задач диагностики состояния и условий эксплуатации, предупреждения сбоев и интеллектуального управления обслуживанием производственного оборудования. Описание использования результатов представлено в научно-техническом отчете по итогам работ выполнения НИОКР за этап 1 (24.12.2018 – 23.06.2019).

Директор ООО «ЦМИТ» Мелентьев В.В.
"05" августа 2019 г. М.П.

Приложение Б

Регистрация программ для ЭВМ

Свидетельства о государственной регистрации программы для ЭВМ.

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2018664673

Программный модуль локального ранжирования данных для составления персональных рекомендаций на основе онтологической базы знаний истории повседневности Петрозаводского государственного университета

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

Авторы: ***Петрина Оксана Борисовна (RU), Марченков Сергей Александрович (RU), Корзун Дмитрий Жоржевич (RU)***

Заявка № **2018661923**
Дата поступления **29 октября 2018 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***20 ноября 2018 г.***

Руководитель Федеральной службы по интеллектуальной собственности

Г.П. Ивлиев

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2018610213

Мобильный клиент для семантического аннотирования и персонализированного доступа к корпусу источников по истории повседневности в музее (версия для операционной системы Android)

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

Авторы: ***Вдовенко Андрей Сергеевич (RU), Марченков Сергей Александрович (RU), Петрина Оксана Борисовна (RU), Корзун Дмитрий Жоржевич (RU)***

Заявка № **2017661363**

Дата поступления **07 ноября 2017 г.**

Дата государственной регистрации в Реестре программ для ЭВМ ***09 января 2018 г.***

Руководитель Федеральной службы по интеллектуальной собственности

Г.П. Ивлиев

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации базы данных

№ 2017621001

Онтологическая база знаний истории повседневности Петрозаводского государственного университета

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

Авторы: ***Петрина Оксана Борисовна (RU), Волохова Валентина Владимировна (RU), Яловицына Светлана Эркиевна (RU), Каюмова Мадина Расуловна (RU), Семьина Дарья Андреевна (RU), Марченков Сергей Александрович (RU), Вдовенко Андрей Сергеевич (RU), Варфоломеев Алексей Геннадьевич (RU), Корзун Дмитрий Жоржевич (RU)***

Заявка № **2017620690**

Дата поступления **03 июля 2017 г.**

Дата государственной регистрации в Реестре баз данных ***01 сентября 2017 г.***

Руководитель Федеральной службы по интеллектуальной собственности

Г.П. Ивлиев

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2016662271

Инструментальное средство C_KP1 платформы Smart-M3 для реализации на языке программирования ANSI C доступа на уровне модели RDF к информационному содержимому интеллектуального пространства

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

Авторы: ***Ломов Александр Андреевич (RU), Марченков Сергей Александрович (RU), Корзун Дмитрий Жоржевич (RU)***

Заявка № **2016619737**

Дата поступления **15 сентября 2016 г.**

Дата государственной регистрации в Реестре программ для ЭВМ ***07 ноября 2016 г.***

Руководитель Федеральной службы по интеллектуальной собственности

Г.П. Ивлиев

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2016614670

Сервис динамического формирования контекстно-ориентированных веб-страниц в интеллектуальном зале SmartRoom

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

Авторы: ***Марченков Сергей Александрович (RU), Вдовенко Андрей Сергеевич (RU), Корзун Дмитрий Жоржевич (RU)***

Заявка № **2015660988**

Дата поступления **13 ноября 2015 г.**

Дата государственной регистрации в Реестре программ для ЭВМ ***27 апреля 2016 г.***

Руководитель Федеральной службы по интеллектуальной собственности

Г.П. Ивлиев

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2016611636

Программный модуль поиска и извлечения историко-культурной информации из базы знаний DBpedia для представления в интеллектуальном пространстве

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

Авторы: ***см. на обороте***

Заявка № **2015662330**
Дата поступления **15 декабря 2015 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***08 февраля 2016 г.***

Руководитель Федеральной службы по интеллектуальной собственности

Г.П. Ивлиев

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2016610808

Сервис историко-культурного сопровождения коллективной работы в интеллектуальном зале SmartRoom

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

Авторы: ***Вдовенко Андрей Сергеевич (RU), Марченков Сергей Александрович (RU), Корзун Дмитрий Жоржевич (RU)***

Заявка № **2015660772**

Дата поступления **10 ноября 2015 г.**

Дата государственной регистрации в Реестре программ для ЭВМ ***19 января 2016 г.***

Руководитель Федеральной службы по интеллектуальной собственности

Г.П. Ивлиев

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