

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

**Федеральное государственное автономное образовательное учреждение высшего
образования «Национальный исследовательский технологический университет
«МИСИС»**

На правах рукописи

Дерябин Сергей Андреевич

Модели и методы оценки и анализа архитектур для построения автономно
функционирующих горнопромышленных систем

2.3.1 – «Системный анализ, управление и обработка информации, статистика»

Диссертация на соискание ученой степени кандидата технических наук

Научный руководитель: доктор технических наук, заведующий кафедрой
автоматизированных систем управления федерального государственного автономного
образовательного учреждения высшего образования «Национальный исследовательский
технологический университет «МИСИС» Темкин Игорь Олегович

Москва – 2026

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	4
ГЛАВА 1. АНАЛИЗ И ОБОСНОВАНИЕ АРХИТЕКТУРНЫХ ПРОБЛЕМ ГОРНОПРОМЫШЛЕННЫХ СИСТЕМ.....	13
1.1 Анализ развития и современного состояния горнопромышленных систем	13
1.2 Модели представления и задачи управления архитектурами систем	20
1.2.1 Модели архитектур систем Индустрии 4.0.....	22
1.2.2. Модели архитектур программных систем	27
1.2.3. Задачи управления архитектурой	33
1.3 Методы оценки и анализа архитектур систем	36
1.3.1 Методы оценки качества архитектур предприятий	36
1.3.2 Методы оценки качества архитектур программных систем	38
1.3.3 Методы анализа сложных систем.....	41
1.4 Постановка задачи исследования	44
1.5 Выводы по Главе 1	51
ГЛАВА 2. РАЗРАБОТКА МОДЕЛЕЙ ПРЕДСТАВЛЕНИЯ АРХИТЕКТУР ИНДУСТРИАЛЬНЫХ СИСТЕМ.....	54
2.1 Формализация критериев и задач цифровой трансформации промышленных систем	54
2.2 Формализация задачи онтологического моделирования архитектуры промышленной системы.....	62
2.3 Разработка онтологической модели архитектуры промышленной системы... ..	67
2.3.1 Разработка онтологии общего назначения	67
2.3.2 Разработка онтологии специального назначения.....	77
2.4 Разработка алгоритмических механизмов формирования метамодели архитектуры	81
2.4.1 Разработка процедур логико-семантического вывода из онтологии	81
2.4.2 Алгоритм формирования метамодели архитектуры.....	86
2.5 Выводы по Главе 2.....	91
ГЛАВА 3. РАЗРАБОТКА МОДЕЛЕЙ ОЦЕНКИ И АЛГОРИТМОВ КОНФИГУРИРОВАНИЯ АРХИТЕКТУР	95
3.1 Разработка моделей представления и критериев оценки архитектуры.....	95
3.2 Разработка операций и правил модификации логической конфигурации архитектур	104
3.3 Разработка критериев и механизмов решения задач цифровой трансформации архитектур	109
3.4 Разработка алгоритмов конфигурирования и анализа архитектур	112
3.4.1 Алгоритм процесса конфигурирования и анализа архитектуры «as is»	112
3.4.2 Алгоритм процесса конфигурирования и анализа архитектуры «to be»	115
3.5 Выводы по Главе 3.....	118

ГЛАВА 4. ЭКСПЕРИМЕНТАЛЬНОЕ МОДЕЛИРОВАНИЕ И АНАЛИЗ РЕЗУЛЬТАТОВ	121
4.1 Результаты моделирования работы алгоритма построения метамодели архитектуры	122
4.1.1 Результаты онтологического моделирования	122
4.1.2 Результаты моделирования работы логико-семантического вывода.....	126
4.2 Результаты моделирования работы алгоритмов конфигурирования и анализа архитектур	133
4.2.1 Результаты моделирования работы алгоритма конфигурирования и анализа архитектуры «as is»	133
4.2.2 Результаты моделирования работы алгоритма конфигурирования и анализа архитектуры «to be»	142
4.3 Анализ результатов.....	154
ЗАКЛЮЧЕНИЕ.....	157
СПИСОК ИСТОЧНИКОВ	160
ПРИЛОЖЕНИЯ	172
Приложение А.	172
Листинг исходного кода алгоритма формирования метамодели архитектуры автономно функционирующей кибер-физической геопространственной индустриальной системы	172
Приложение Б.....	180
Листинг исходного кода алгоритма конфигурирования и анализа архитектуры «as is»	180
Приложение В.	184
Листинг исходного кода алгоритма конфигурирования и анализа архитектуры «to be»	184
Приложение Г.....	189
Акт внедрения результатов диссертационной работы в ООО «ПРАЙМ ГРУП».....	189
Приложение Д	190
Акт внедрения результатов диссертационной работы в ООО «Брайт Софт».....	190
Приложение Е	191
Акт внедрения результатов диссертационной работы в ООО «Экспасофт»	191

ВВЕДЕНИЕ

Актуальность работы. В последние годы в мире решается нетривиальная задача проведения Цифровой Трансформации (ЦТ) промышленных предприятий в парадигме концепции «Индустрия 4.0». Как правило, под цифровой трансформацией подразумевается переход к качественно иному способу организации и управления технологическими процессами с целью повышения их интегральной эффективности посредством комплексного применения современных информационных технологий, включая роботизацию оборудования, Индустриальный Интернет Вещей, интеллектуальный анализ данных, Цифровые Двойники и др. При этом в научной и практических сферах в настоящий момент не существует консенсуса по вопросу формализации данного процесса, в частности: выбора инвариантных критериев, разработки методов оценки и анализа результатов подобной деятельности, а также формирования требований к конечным виду и форме структурно-функциональной организации производств. Тем не менее, при аналитическом обобщении известных стандартов и фреймворков в данной области, таких как, например, TOGAF 10 и RAMI4.0, можно сделать вывод, что разработчики этих стандартов видят суть ЦТ в реорганизации архитектур предприятий для построения автономно функционирующих индустриальных систем. В контексте настоящего исследования под автономно функционирующими индустриальными системами понимаются системы, реализующие все технологические процессы без участия человека и обладающие способностями к самоорганизации и саморегуляции. Таким образом суть решения задачи ЦТ может быть сведена к процессу замещения в архитектуре предприятия персонала на программные системы, интегрированные в единую кибер-физическую производственную среду с четко формализованной моделью структурно-функциональной организации деятельности на базе онтологического представления априорных знаний.

Цифровая трансформация горнопромышленных предприятий, на фоне других индустриальных сфер, представляется одной из наиболее сложных задач, что связано, с особенностями горного производства: пространственной распределённостью и нестационарностью геотехнической производственной среды; наличием множества разнотипных специализированных автоматизированных систем с низкой степенью интеграции; отсутствием подходов к унифицированному модельному представлению гетерогенных компонент архитектуры предприятия. Учитывая подобные специфические условия, определено, что существующие подходы цифровой трансформации являются малоприспособленными для горнопромышленных предприятий, характеризующихся в настоящем исследовании как сложные крупномасштабные пространственно-распределенные

геотехнические системы с открытым типом производственных сред. В ещё большей степени задача эффективной трансформации систем подобного типа осложнена отмеченным ранее отсутствием формальных критериев и методов количественной оценки состояний архитектур, инвариантных различным этапам ЦТ. Существующие подходы имеют преимущественно экспертно-аналитический характер и не обеспечивают строгой формализации, воспроизводимости и количественного сопоставления различных вариантов архитектурных решений. Таким образом, разработка формальных моделей представления, количественных критериев оценки качества, алгоритмов конфигурирования и анализа архитектур автономно функционирующих кибер-физических пространственно-распределенных геотехнических промышленных систем является актуальной научно-исследовательской и научно-практической задачей.

Степень разработанности темы исследования. Фундаментальные теоретические основы в области автоматизации производств заложены в работах Я.З. Цыпкина, А.А. Красовского, Л.С. Понтягина и В.М. Глушкова, а современные проблемы создания автономно функционирующих кибер-физических промышленных систем рассматривались L. Monostori, J. Lee, P. Leitão. Применительно к горнодобывающей отрасли большой вклад внес Н. Durrant-Whyte, реализовавший концепцию автономных горнотранспортных систем, а также отечественные учёные: Л.А. Пучков, Н.И. Федунец, Д.К. Потресов, внесшие вклад в развитие теории построения АСУ в горной отрасли, и И.О. Темкин, формализовавший и систематизировавший методологические аспекты применения элементов теории Искусственного Интеллекта для комплексного решения широкого спектра задач управления горнодобывающими производствами.

Понятие архитектуры предприятия как системного описания структуры впервые формализовано J.A. Zachman, что получило дальнейшее развитие в методологии стандарта TOGAF одноименного международного консорциума и стандарте RAMI 4.0 для промышленных систем Индустрии 4.0 консорциума Plattform Industrie 4.0. Среди отечественных исследователей вопросы архитектурного проектирования предприятий в контексте цифровой трансформации рассматривались в работах Г.Б. Клейнера, А.А. Ковалёва, С.Д. Бодрунова. Теоретические основы архитектур программного обеспечения заложены в работах D. Parnas, систематизированы M. Shaw, D. Garlan, L. Bass, P. Clements, R. Kazman и развиты в работах S. Newman, M. Fowler и R. Richards. Однако сегодня сохраняется разрыв между понятиями архитектуры предприятия и архитектуры программного обеспечения, а все предложенные модели ориентированы преимущественно на визуально-аналитическое описание и не предоставляют формального аппарата для количественного сопоставления архитектурных решений.

Фундаментальные основы современной инженерии онтологий, как способа формального представления знаний о системах, разработаны Т. Gruber, N. Guarino и Т. Т. Berners-Lee. В отечественной научной школе вопросы инженерии знаний и онтологического моделирования разрабатывались А.С. Клещёвым, Т.А. Гавриловой, В.Ф. Хорошевским, Д.А. Пospelовым. Теоретические основы дескрипционной логики систематизированы F. Baader, D. Calvanese, D. McGuinness, D. Nardi и P. Patel-Schneider. Существенный вклад в развитие языка дескрипционной логики OWL внесли I. Horrocks, B. Motik, U. Sattler. Применение онтологического моделирования, в том числе на языке OWL, для описания отдельных технологических процессов и объектов горнопромышленных систем известно по работам B. Zhang, R.M. Stanković, L. Kolonja, Л. Власенко и др. Примеры комплексного применения онтологий и дескрипционной логики для создания моделей архитектур горнопромышленных систем в настоящий момент в научной литературе не представлены.

Широко известны методы экспертно-аналитической оценки качества архитектур программных систем (Architecture Tradeoff Analysis Method), разработанные R. Kazman и развитые M. Klein, P. Clements, С.В. Зыковым; а также количественные метрики качества программного кода, включая: цикломатическую сложность Маккейба (Т.Т. McCabe), метрики Халстеда (М.Н. Halstead) и объектно-ориентированные метрики Чидамбера и Кемерера (S.R. Chidamber, C.F. Kemerer). Однако несмотря на разработанность и относительную развитость данных методов, их применение для количественной оценки архитектур сложных промышленных систем практически невозможно.

В области теоретико-информационного анализа можно отметить работы N. Rashevsky, A. Mowshowitz и M. Dehmer, предложивших использовать энтропийные критерии оценки для анализа графовых структур сложных систем; а также работы H. Bunke и K. Riesen, определяющих концепцию расстояния редактирования графов как меры различия двух структур. Совмещение энтропийной оценки сложности графа с оценкой стоимости операций редактирования создаёт теоретическую основу для формирования комплексного критерия управления жизненным циклом цифровой трансформации архитектуры, однако перенос данных подходов с уровня абстрактных графовых структур на уровень архитектуры промышленной системы, включая горнопромышленные предприятия, остается нерешенной задачей, что подтверждается отсутствием соответствующих публикаций.

Целью настоящего исследования является обеспечение процессов цифровой трансформации горнопромышленных предприятий посредством разработки формальных критериев, моделей и алгоритмов оценки и анализа архитектур на всех этапах построения

автономно функционирующих кибер-физических пространственно-распределенных геотехнических промышленных систем.

Для достижения поставленной цели в работе решаются следующие **задачи**:

1. Сформулировать критерии и формализовать задачи управления цифровой трансформацией архитектур сложных крупномасштабных пространственно-распределенных геотехнических предприятий для их приведения к виду автономно функционирующих кибер-физических промышленных систем.
2. Разработать модели представления знаний о структурно-функциональной организации пространственно-распределенных геотехнических промышленных систем, обеспечивающих необходимую вычислительную инструментальную основу самоорганизации и саморегуляции на базе современных средств онтологического моделирования.
3. Разработать унифицированные критерии количественной оценки и механизмы анализа архитектур инвариантно их состояниям на отдельных этапах жизненного цикла цифровой трансформации.
4. Разработать формальные правила и алгоритмические процедуры модификации структурно-функциональной организации архитектур, обеспечивающих возможности определения оптимальных планов конфигурационных изменений в целях построения автономно функционирующих кибер-физических промышленных систем.
5. Осуществить моделирование для апробации и верификации разработанных критериев, моделей и алгоритмов на примере реальных технологических процессов и архитектур горнопромышленных предприятий по открытой добыче полезных ископаемых.

Основная идея работы заключается в комплексном применении современных средств логико-семантического моделирования на базе дескрипционной логики с использованием языка OWL и методов алгебры множеств в сочетании с теоретико-графовым подходом к оценке информационной сложности архитектур горнопромышленных предприятий для формализации и разработки критериев, моделей и алгоритмов, обеспечивающих возможность решения задач цифровой трансформации для построения автономно функционирующих кибер-физических систем.

Научная новизна исследования состоит в следующем:

1. В комплексном применении современных средств онтологического моделирования и дескрипционной логики, а также механизмов логико-семантического вывода в виде строгого комбинаторного перебора допустимых

предикативных высказываний, обеспечивающих непротиворечивость, полноту и точность вывода априорных экспертных знаний для формирования метамодели архитектуры индустриальной системы.

2. В формализации представления компонент индустриальной системы, которая позволяет описать взаимосвязь атомарных функциональных задач управления, агентов управления и физических узлов размещения и согласовать понятия архитектуры предприятия с архитектурами программных систем.
3. В предложении универсального энтропийного критерия оценки сложности логической конфигурации архитектуры индустриальной системы, обеспечивающего возможность количественного сопоставления состояний системы, независимо от этапов жизненного цикла.
4. В разработке набора формальных операций модификации логической конфигурации архитектуры с эвристической метрикой стоимости структурных изменений графа в виде петель модификации и правила допустимости таких операций, позволяющих осуществлять количественное сопоставление вариантов архитектуры для выбора оптимального на каждом следующем этапе трансформации.
5. В разработке и экспериментальном подтверждении инструментальной основы решения задач по управлению жизненным циклом цифровой трансформации горнопромышленных предприятий и созданию автономно функционирующих кибер-физических пространственно-распределенных геотехнических индустриальных систем.

Основные **научные положения**, выносимые на защиту:

1. Алгоритм формирования метамодели архитектуры на базе онтологического моделирования и комбинаторных механизмов логико-семантического вывода априорных знаний, обеспечивающий точность, полноту и непротиворечивость результатов формального построения структурно-функциональной конфигурации пространственно-распределенной геотехнической индустриальной системы.
2. Теоретико-множественная модель представления архитектуры, включающая графоаналитическое описание компонент и критерии оценки их структурной сложности на базе информационных индексов, реализующих унифицированный подход к анализу логической конфигурации сложных крупномасштабных пространственно-распределенных геотехнических индустриальных систем для приведения их к автономно функционирующему виду.

3. Метод конфигурирования и модификации архитектуры с использованием механизмов количественной оценки структурных преобразований графа на основе петель модификации, обеспечивающий количественное сопоставление состояний индустриальной системы инвариантно этапам жизненного цикла и возможность планирования оптимальных стратегий её цифровой трансформации.

Теоретическая значимость работы заключается в развитии основ системного анализа архитектур сложных крупномасштабных пространственно-распределенных геотехнических индустриальных систем в части: формализации понятия архитектуры как графовой структуры; установления фундаментального компромисса между минимизацией структурной сложности архитектуры и минимизацией сложности её модификации; формализации связи между онтологическим моделированием знаний о системе и количественной оценкой качества её архитектуры.

Практическая ценность результатов определяется тем, что разработан набор программных инструментов, которые:

- реализуют алгоритмические процедуры формализации знаний о структурно-функциональной организации систем в ходе составления модели и планировании цифровой трансформации;
- позволяют количественно оценивать и сопоставлять варианты архитектурных решений в целях выбора оптимальных стратегий в соответствии с конкретными этапами жизненного цикла систем;
- обеспечивают возможность выявлять критические элементы архитектуры в целях глубинного анализа качества функционирования пространственно-распределенных геотехнических индустриальных систем;
- предоставляют информацию для обоснования выбора путей модификации архитектуры на основе баланса структурной сложности и стоимости процедур модификации.

Результаты работы востребованы, в первую очередь, крупными ИТ-компаниями, участвующими в цифровой трансформации горнопромышленных и других пространственно-распределенных геотехнических индустриальных систем, а также непосредственно горнодобывающими предприятиями.

Обоснованность и достоверность результатов обеспечивается: корректным применением теории и методов системного анализа, графовых моделей, а также теоретико-информационных подходов к оценке сложности систем для математической формализации представления архитектур горнопромышленных систем. Верификацией консистентности и

когерентности разработанной онтологической модели инструментальными средствами среды Protégé для доказательства истинности и непротиворечивости заложенных аксиом с использованием специализированных механизмов – «рассуждателей» (HermiT) с доказанной вычислительной разрешимостью. Соответствием результатов моделирования работы механизмов логико-семантического вывода с теоретическими оценками предела комбинаторного перебора. Экспериментальным моделированием работы алгоритмов конфигурирования и анализа архитектуры и получением воспроизводимых количественных результатов.

Апробация работы.

Основные результаты диссертационного исследования докладывались и обсуждались на следующих международных и отечественных научных конференциях:

- XXXIV Международный научный симпозиум «Неделя горняка – 2026». г. Москва, НИТУ МИСИС, 2-6 февраля 2026 г.
- XXX Международный научный симпозиум «Неделя горняка – 2022». г. Москва, НИТУ МИСИС, 1-4 февраля 2022 г.
- The 24-th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems «KES 2020». онлайн, 16–18 сентября 2020 г.
- XXVIII Международный научный симпозиум «Неделя горняка – 2020». г. Москва, НИТУ МИСИС, 27 января – 31 января 2020 г.
- The 5-th International Conference on Fuzzy Systems and Data Mining «FSDM 2019». г. Китакусси, Япония, 18-21 октября 2019 г.
- The 23-rd International Conference on Knowledge-Based and Intelligent Information & Engineering Systems «KES 2019». г. Будапешт, Венгрия, 4-6 сентября 2019 г.
- The 2019 International Conference on Cooperation and Integration of Industry, Education, Research and Application. г. Харбин, КНР, 14-16 июня 2019 г.
- 1-ая Международная научно-практическая конференция «Шаг в будущее: Искусственный Интеллект и Цифровая Экономика». г. Москва, ГУУ, 4-5 декабря 2017 г.

Научные положения и ряд основных научно-практических результатов, полученные автором, использованы при выполнении в Федеральном государственном автономном образовательном учреждении высшего образования «Национальный исследовательский технологический университет «МИСИС» следующих грантовых проектов и хозяйственно-договорных работ:

1. Грант РФФИ №23-11-00197 «Гибридные подходы для анализа деятельности и визуализации функционирования сложных многомерных систем», 2023-2025 гг., исполнитель.
2. Грант РФФИ № 17-11-01353 «Разработка моделей и методов для анализа деятельности и визуализации поведения сложных многомерных объектов», 2021-2022 гг., исполнитель.
3. Грант РФФИ №19-17-00184 «Разработка методов построения и моделей функционирования цифровой платформы управления транспортно-технологическими процессами при добыче минерального сырья», 2019-2021 гг., основной исполнитель.
4. Грант РФФИ №16-07-01197 «Разработка и исследование вычислительных моделей для построения механизма управления роботизированными объектами горно-транспортного комплекса», 2016-2018 гг., исполнитель.
5. Договор оказания услуг с АО «МХК «ЕвроХим» от 10.01.2022 г. №8000433216 «на проведение работ по изучению конъюнктуры рынка программно-технических комплексов для построения систем позиционирования, оповещения и поиска, а также других смежных систем в условиях подземных рудников», основной исполнитель.
6. Договор оказания услуг с АО «Электронная Москва» от 01.09.2020 г. №32009357453 «по апробации технологии цифрового дублирования операционной деятельности работников на объектах Комплекса градостроительной политики и строительства г. Москвы», основной исполнитель.

Реализация и внедрение результатов работы.

Результаты диссертационного исследования внедрены и используются в деятельности ряда отечественных ИТ-компаний, специализирующихся на разработке и интеграции программных продуктов для автоматизации технологических процессов в различных секторах производства:

1. Общество с ограниченной ответственностью «ПРАЙМ ГРУП».
2. Общество с ограниченной ответственностью «Экспасофт».
3. Общество с ограниченной ответственностью «Брайт Софт».

Публикации

По теме диссертации опубликовано 10 научных работ, в том числе 8 в периодических научных изданиях, индексируемых в международных базах цитирования Scopus и Web of Science, 2 из которых включены в «белые списки журналов» РЦНИ, 1 в

изданиях, рекомендованных ВАК, 1 в сборниках трудов отечественных научных конференций. По результатам работ в составе коллектива авторов получено 4 свидетельства о государственной регистрации программ для ЭВМ.

Структура и объем работы. Диссертация состоит из введения, четырёх глав, заключения и списка использованных источников. Работа изложена на 191 странице машинописного текста, содержит 60 рисунков, 25 таблиц. Список использованных источников включает 142 наименования. Работа содержит 6 приложений.

ГЛАВА 1. АНАЛИЗ И ОБОСНОВАНИЕ АРХИТЕКТУРНЫХ ПРОБЛЕМ ГОРНОПРОМЫШЛЕННЫХ СИСТЕМ

1.1 Анализ развития и современного состояния горнопромышленных систем

Сегодня в горнодобывающей отрасли, как и во многих других областях экономико-технологических сфер деятельности, стоит нетривиальная задача по организации цифровой трансформации предприятий в рамках концепции «Индустрия 4.0» [1-5]. При этом конкретного универсального определения для термина «цифровая трансформация», как применительно для сферы добычи полезного ископаемого, так и для любой иной отрасли, в настоящий момент не существует [6-8]. Однако сама суть, закладываемая в данное понятие, во многом проистекает из предпосылок появления термина – радикальных изменений способов организации производства, произошедших на отдельных исторических этапах, посредством применения передовых современных технологий для повышения эффективности и объемов производственной деятельности, которым дано условное название промышленные революции. В этой связи, сегодня существенным и важным фактором определения вектора цифровой трансформации является как выявление конкретных технологий Индустрии 4.0, обеспечивающих качественно-количественные изменения производства, так и ретроспективный анализ организации управления предприятиями в горной отрасли.

Исторически горнодобывающую отрасль можно охарактеризовать как одну из наиболее структурированных, обладающую четким регламентом производственных работ и хорошо подверженной декомпозиции [9-10]. Так характерные принципы организации процессов в горной отрасли не зависимо от исторического этапа в терминологии жизненного цикла производства можно выразить в виде обобщенной последовательности отдельных мероприятий:

- поиск и разведка запасов месторождения;
- подготовительные работы по долгосрочному планированию производства и предварительному вскрытию месторождения (т.е. удалению пустой горной породы);
- непосредственно комплекс добычных работ, направленных на извлечение полезных ископаемых и их доставку до потребителей;
- в отдельных случаях, обогащение полезных ископаемых, т.е. повышение их концентрации в отгружаемой потребителям горной массе;
- закрытие месторождения и рекультивация нарушенных земель, под которой понимается выполнение действий по восстановлению природных компонентов в области производственных работ.

Однако, с учётом детального анализа непрерывного развития горной отрасли, можно отметить, что каждое из перечисленных мероприятий обрастало большим количеством внутренних функциональных задач и элементов и, как следствие, усложнялись их структурно-функциональные схемы взаимодействия с целью повышения качества реализации производственных процессов [4, 5, 11]. Вследствие чего, сегодня представить унифицированную комплексную схему типового горнопромышленного предприятия с использованием традиционных подходов проектирования систем достаточно сложно, но можно выделить её следующие ключевые особенности:

1. Пространственная распределенность геотехнологической среды с непрерывно изменяющимися границами и состояниями типовых инфраструктурных объектов.
2. Высокая механизация труда и применение широкого спектра мобильного и стационарного технологического оборудования.
3. Применение большого числа разнотипных специализированных программных систем для автоматизации ряда технологических процессов и операций.

При этом в настоящее время считается, что горнодобывающая отрасль прошла третью промышленную революцию – автоматизацию технологических процессов и производств [4, 5, 11]. Суть данного этапа трансформации производств заключалась в комплексном применении вычислительных технологий в части внедрения средств сбора оперативных данных о ходе реализации технологических процессов с последующей обработкой информации и принятием решений по изменению условий функционирования производства информационно-аналитическими системами под контролем или с прямым участием человека [12, 13].

В первую очередь автоматизация затронула горнотранспортный комплекс, для которого были разработаны и внедрены бортовые системы мониторинга технико-технологических состояний оборудования и системы централизованного диспетчерского управления – автоматизированные системы управления горнотранспортным комплексом (АСУ ГТК) [5, 12, 14]. Рассматривая современные АСУ ГТК с традиционного взгляда пирамиды иерархии автоматизации (Рисунок 1), достаточно сложно отнести их к конкретному уровню управления – технологического процесса или производства, т.к. они обладают функциями решения задач обоих уровней.

На ранних этапах (начиная с конца 1980-х годов и до начала 2010-х) АСУ ГТК представляли собой в большей степени автоматизированные системы управления технологическими процессами или SCADA (Supervisory Control And Data Acquisition) [10, 12]. Основной задачей таких систем являлось получение информации о местоположении объектов горнотранспортного комплекса в производственной среде, контроль загрузки

кузова самосвала и уровня топлива, а также радиосообщение операторов техники с диспетчером для двухсторонней передачи оперативных данных о ходе выполнения процессов и изменении режимов работы. Информация, собираемая с бортовых сенсорных устройств, передавалась по радиоканалам на сервер системы и визуализировалась в виде линейных мнемосхем связки «забой-отвал» или «экскаватор-самосвал», графиков технико-технологических показателей оборудования и спутникового местоположения объектов на карте местности с целью принятия решений по управлению процессом диспетчерским персоналом. При этом реализация большого числа задач, связанных с выработкой управляющих решений, таких как оперативное переназначение маршрутов/заданий, оптимизация производственных показателей процесса, оценка эксплуатационного состояния оборудования и др., оставались в виде не автоматизированной («ручной») интеллектуальной деятельности диспетчерского персонала и операторов техники. Среди наиболее известных и широко распространенных систем в международной и отечественной практике можно назвать такие как MineStar (Caterpillar, США), DISPATCH (Modular Mining Systems, США), Wenco (Wenco International Mining Systems, Канада), Карьер (ВИСТ групп, Россия) [15, 16].

В дальнейшем (с 2010-х годов) с развитием информационных технологий и во многом благодаря наличию базы SCADA-функциональности, АСУ ГТК начали свою поэтапную трансформацию до уровня систем управления производственными процессами или MES (Manufacturing Execution System). Помимо автоматизации задач комплексного оперативного диспетчерского анализа, планирования и управления добычей полезных ископаемых, в контур АСУ ГТК легли и задачи процессов обеспечения и поддержки, таких как, например, технические осмотры и ремонты оборудования, включая предиктивное обслуживание [17-18], контроль качества дорожного полотна в карьере [19-21] и др. Для математико-алгоритмического решения ряда задач использовались как традиционные методы линейного программирования, эвристические и инженерно-технические подходы [22,23], так и методы интеллектуального анализа данных и машинного обучения [24-25]. Применению последних методов поспособствовало фактическое наличие больших объемов исторических данных, накапливаемых в течение продолжительного периода эксплуатации диспетчерских систем [13], таким образом, открыв дальнейшее направление по разработке средств «безлюдного» выполнения технологических процессов.

В виду неминуемо устойчиво высоких рисков в области безопасности, непрерывной интенсификации объемов и масштабирования производственной деятельности, а также развитием средств информатики и вычислительной техники, очевидным трендом стало применение элементов роботизированных технологий [26-29], направленных, в свою

очередь, на повышение непрерывности процессов посредством исключения участия людей из реализации наиболее опасных и трудоемких операций по извлечению и транспортировке горной массы [30-32]. Ключевые инжиниринговые компании-разработчики АСУ ГТК совместно с производителями горнодобывающей техники (Caterpillar, Komatsu, Hitachi, БелАЗ) вели активные работы в области создания автономных транспортных систем (Autonomous Haulage Systems) [33-35].



Рисунок 1. Пирамида иерархии АСУ в контексте горнодобывающих предприятий открытого типа

Наиболее масштабным и значимым можно назвать пилотный проект в австралийском карьере региона Пилбара компании Rio Tinto, на котором в 2015 году было осуществлено внедрение 150 роботизированных самосвалов Komatsu, выполняющих задачи транспортирования горной породы из забоя в отвалы без участия человека (при удаленном наблюдении диспетчерами) [36]. Исходные особенности проекта не позволили развить успех и осуществить перенос и масштабирование результатов на другие карьеры. Большая часть компаний-производителей и горнодобывающих предприятий столкнулись с

рядом технических трудностей [37], однако работы в данном направлении продолжают в рамках опытно-экспериментальных запусков отдельных участков с небольшим числом роботизированных самосвалов. В ходе таких экспериментов, а также с учётом научно-исследовательских изысканий, фокус дальнейшей автоматизации горнодобывающих производств сместился в область создания кибер-физических систем [38-39], полно, адекватно и точно описывающих динамику сложной технологической среды в виде её цифрового аналога (двойника) с целью выработки надежных, безопасных и высокоэффективных управляющих решений. Как предполагается, Цифровые Двойники предприятий должны обеспечить автономное выполнение цикла производственных задач посредством интеграции большего числа аппаратных средств сбора данных и переноса всех функциональных элементов в цифровое представление, что позволит не только повысить показатели эффективности, но и открыть возможности к изменению стратегий управления предприятиями, в том числе обеспечит необходимый уровень безопасности управления беспилотной техникой [40-44].

Таким образом, сегодня наиболее важной проблемой на пути цифровой трансформации горнодобывающих производств в рамках концепции Индустрии 4.0 можно назвать поиск способов представления и интеграции всех функциональных компонентов предприятия в виде единой информационно-управляющей среды [45-47]. На пути же решения данной проблемы стоят сложности, связанные с отсутствием единого подхода к пониманию структурно-функционального представления подобной системы [48-51], необходимостью реструктуризации действующих информационно-управляющих сред предприятий [52-56], а также отсутствие инструментальной базы, обеспечивающей переход от автоматизированного управления к полностью автономному функционированию производства [54, 57]. В качестве такой инструментальной базы следует понимать как конкретные вычислительные модели управления, так и механизмы программного взаимодействия функциональных компонентов – начиная от роботизированных объектов горнотранспортного комплекса и заканчивая отдельными программными приложениями, так что реализация цифровой трансформации производств во многом лежит в области задач организации эффективной работы с данными [48, 52, 55] посредством соответствующих протоколов [58-60], алгоритмов и способов архитектурного исполнения программ [47, 54, 61, 62].

Эксплуатируемые в настоящий момент системы диспетчеризации с точки зрения программно-технической реализации обладают проприетарными клиент-серверными исполнениями архитектуры и относительно единообразной функциональностью, что обеспечивает их взаимозаменяемость и легкое понимание способов интеграции в бизнес-

логику предприятий [12, 16, 24]. С другой стороны, стоит отметить, что данные системы изначально выступали в качестве каркасо-образующих сред построения информационно-вычислительных структур предприятий, но при этом не покрывали все потребности в области автоматизации управления добычей полезных ископаемых.

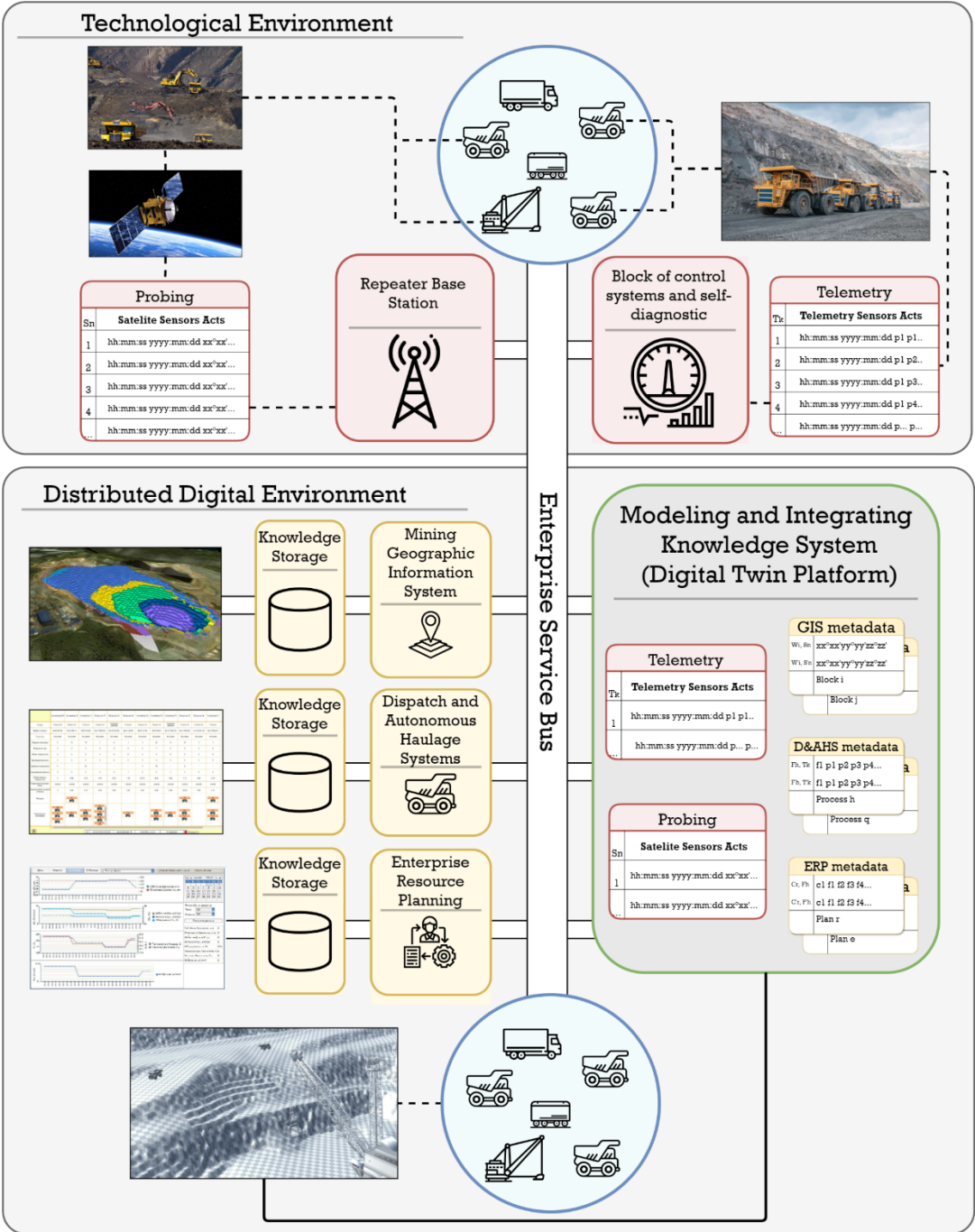


Рисунок 2. Концептуальная структурно-функциональная схема горнодобывающего предприятия

Задачи, связанные с горно-геологическими и маркшейдерскими измерениями и расчётами, не входили в контуры оперативного управления технологическими процессами и требовали специализированных инструментов вида CAD/CAM. В качестве таких

элементов выступают горно-геологические информационные системы (ГГИС), обеспечивающие задачи автоматизированного проектирования плана развития месторождения, предоставления картографических маркшейдерских и геологических схем, а также формирования плана добычи посредством декомпозиции месторождения до единичных атомарных элементов первичной продукции – блоков горной массы [63, 64].

Другой, не менее важной частью выступают системы управления ресурсами предприятия – ERP-системы (Enterprise Resource Planning), обеспечивающие задачи прогнозирования и планирования поставок продукции и ресурсов, необходимых для реализации производственных задач. При этом сегодня все перечисленные системы – АСУ ГТК, ГГИС и ERP – образуют единую сложно сконфигурированную организационно-функциональную среду управления [65], включающую элементы межсистемного информационного взаимодействия (Рисунок 2).

Ввиду исходной клиент-серверной архитектуры систем процессы межсистемного взаимодействия реализовывались на каждом предприятии в индивидуальном порядке без каких-либо единых шаблонов или подходов к структурированию информационного пространства и представляли из себя создание буферных хранилищ информации и отдельных уникальных интерфейсов взаимодействия. Это привело к получению слабоструктурированных хранилищ информации («Озер данных»), содержащих большие объемы технологических данных, но не позволяющих использовать их для построения каких-либо вычислений без участия специалиста, способного идентифицировать назначение такой информации для извлечения знаний. Данный подход не вызывал критических проблем при автоматизации отдельных задач управления процессами в рамках модернизации эксплуатируемых систем. Однако внесение каких-либо изменений в состав компонентной базы информационно-управляющей среды и соответственно структуры продуцируемых данных, требует реорганизации всех взаимосвязанных компонент. При этом для внесения таких изменений требуется отключение ряда систем, что приводит к частичной или полной парализации производственных процессов, а часть продуцируемых данных полностью утрачивается.

В условиях же цифровой трансформации горнодобывающих производств, подразумевающей интеграцию новых программных компонентов в действующие процессы предприятий, такой подход к организации межсистемного взаимодействия очевидно не соответствует ни бизнес-требованиям, ни требованиям Индустрии 4.0. В этой связи одной из основных задач цифровой трансформации можно назвать реорганизацию способов взаимодействия функциональных компонентов горнопромышленного предприятия, обеспечивающей возможность автономной реализации производственных процессов.

1.2 Модели представления и задачи управления архитектурами систем

Детальный анализ опыта отечественной и зарубежной практики решения задач в области цифровой трансформации предприятий [1, 2, 15], в настоящий момент, показывает отсутствие единого формального представления к комплексным результатам такой деятельности. В весьма общем смысле под термином цифровой трансформации понимается переход к качественно новому уровню реализации бизнес-процессов, что дает лишь абстрактное представление конечного результата.

Имея в виду тот факт, что предшествующим (и формально завершенным) этапом трансформации предприятий являлась их автоматизация, следует предполагать, что цифровизация заключается не только в применении современных методов и средств информационных технологий. В контексте известных научных исследований [46, 47] и практических решений говорится о применении новых программно-аппаратных средств и реорганизации структурно-функциональных и информационных схем взаимодействия компонентов предприятия, участвующих в реализации производственных процессов. Схемы взаимодействия, как правило, принято называть архитектурой предприятия [48, 52, 66-72], что в условиях понятия «цифровая трансформация» тесно переплетается с понятием архитектуры исполнения программных систем [54, 73-76]. Однако данные понятия не являются тождественными, что приводит к рассогласованности в терминологии и в понимании целей, задач и результатов цифровой трансформации.

Более того, существующие подходы плохо объясняют и не дают каких-либо критериев и способов оценки различий между предприятием с «высоким» уровнем автоматизации и предприятием, прошедшим цифровую трансформацию. Для выявления четкой границы между «автоматизацией» и «цифровизацией» производств, в рамках настоящего исследования под цифровой трансформацией будет подразумеваться минимизация или полное исключение (там, где это возможно) участия человека в бизнес-процессах. Это означает, что реализуемые в настоящее время процессы с непосредственным участием человека, в том числе с помощью информационных и автоматизированных систем, должны быть приведены к автономно исполняемой программно-аппаратной форме.

Введем следующие определения:

Архитектура предприятия – это действующая форма структурно-функциональной организации технологических процессов и сопряженных с ними процедур информационного взаимодействия между акторами реализации таких процессов.

Модель архитектуры предприятия – это форма абстрактного представления объективных знаний об архитектуре предприятия.

Цифровая трансформация – это процесс модификации архитектуры предприятия, направленный на построение автономно функционирующей производственной системы.

В современной практике управления IT-проектами отмечается [77], что эффективность разработки, поддержки функционирования и модернизации системы зависит от наличия хорошо задокументированной модели архитектуры [66-71]. Такая модель должна обладать свойствами лаконичного представления её компонентов и их формальной структурно-функциональной взаимосвязи, обеспечивая тем самым необходимыми знаниями всех лиц, принимающих решения по управлению [48-50, 72, 76]. Имея же в виду что результатом цифровой трансформации предполагается построение автономно функционирующего предприятия, следует предположить, что система управления такого предприятия, с одной стороны должна являться частью действующей архитектуры, а с другой, она должна использовать ее модельное представление при принятии решений.

В соответствии с изложенным введем следующее определение:

Автономно функционирующее (или цифровое) предприятие – это индустриальная система, реализующая технологические процессы изготовления продукции без участия человека, обладающая способностями к самоорганизации и саморегуляции.

Исходя из введенного определения, можно заключить о необходимости обеспечения механизмов взаимосвязи модели и действующей архитектуры для управления автономно функционирующим предприятием. Тогда для реализации таких масштабных проектов, как цифровая трансформация промышленных предприятий, вопросы разработки моделей архитектуры играют значительную роль и требует поиска решений как в области формирования эталонной компонентной базы, так и к способам её адекватного описания, представления и управления. Однако составление моделей архитектур является не тривиальным процессом в виде особенностей формулирования исходных требований, а именно:

- необходимость поиска компромисса при составлении и чтении модели между специалистами предприятий, обладающими компетенциями в понимании бизнес-процессов и не разбирающимися в программной реализации систем, и разработчиками, не обладающими знаниями тонкостей бизнес-процессов, но требующими четкого понимания поставленных перед ними проектных задач;
- сложность формализации, а в некоторых случаях полное непонимание конечных функциональных и нефункциональных требований к цифровому предприятию как к

единой системе, а также способов качественно-количественной оценки их достижения;

- необходимость поэтапного изменения архитектуры предприятия, которая, помимо прочего, может включать выполнение отдельных бизнес-процессов в «ручном режиме»;
- включение в модель архитектуры будущей системы многих слабо связанных новых и перспективных технологий, которые еще не были внедрены на предприятии на момент начала проектирования цифровой трансформации, а также отсутствие реальных примеров успешной интеграции подобных решений в той же или смежных отраслях.

1.2.1 Модели архитектур систем Индустрии 4.0

В практике организации управления предприятиями различных секторов экономической деятельности широко известны стандарты и фреймворки построения архитектуры автоматизации и/или цифровизации бизнес-процессов, такие как eTOM (Enhanced Telecom Operations Map), COBIT (Control Objectives for Information and Related Technologies), ITIL (Information Technology Infrastructure Library), PCF (Process Classification Framework) и многие другие [78, 79]. Как правило данные стандарты представляют из себя набор рекомендаций общего характера для формирования проектно-рабочей документации без каких-либо конкретных примеров исполнения, руководств и инструментов реализации для управления архитектурой. Фреймворки же, являющиеся по сути своей инструментальным подходом исполнения архитектуры, упрощают вопросы разработки проектов по реорганизации действующих бизнес-процессов, однако их применение не всегда является универсальным и эффективным. В рамках цифровой трансформации предприятий сегодня ориентируются на ключевой международный стандарт «**IEC PAS 63088:2017 Smart manufacturing – Reference architecture model Industry 4.0 (RAMI4.0)**» («ГОСТ Р 59799-2021 Умное производство. Модель эталонной архитектуры Индустрии 4.0») [67, 68, 71] и «**TOGAF 10**» (The Open Group Architecture Framework) – стандарт и фреймворк построения корпоративных архитектур предприятий [69, 70].

TOGAF 10 является результатом деятельности международной инжиниринговой группы разработчиков программного обеспечения «The Open Group Architecture Forum», работа которых берет своё начало с 1995 года. Целью данного фреймворка является упрощение способов проектирования современных архитектур сложных крупномасштабных корпоративных сред на базе гибкой методологии разработки ПО (Agile) и унифицированного подхода к формированию проектной документации для учета всех

заинтересованных сторон проекта цифровой трансформации компании. Основными элементами фреймворка является Метод Разработки Архитектуры ADM (Architecture Development Method), регламентирующий этапы, терминологию и принципы создания архитектур, и Базовая Архитектура (Foundation Architecture. Рисунок 3), включающую в себя Техническую Эталонную Модель (Technical reference model – TRM), набор базовых Строительных Блоков (Building Blocks) и Базу Данных Стандартов (Standards Information Base).

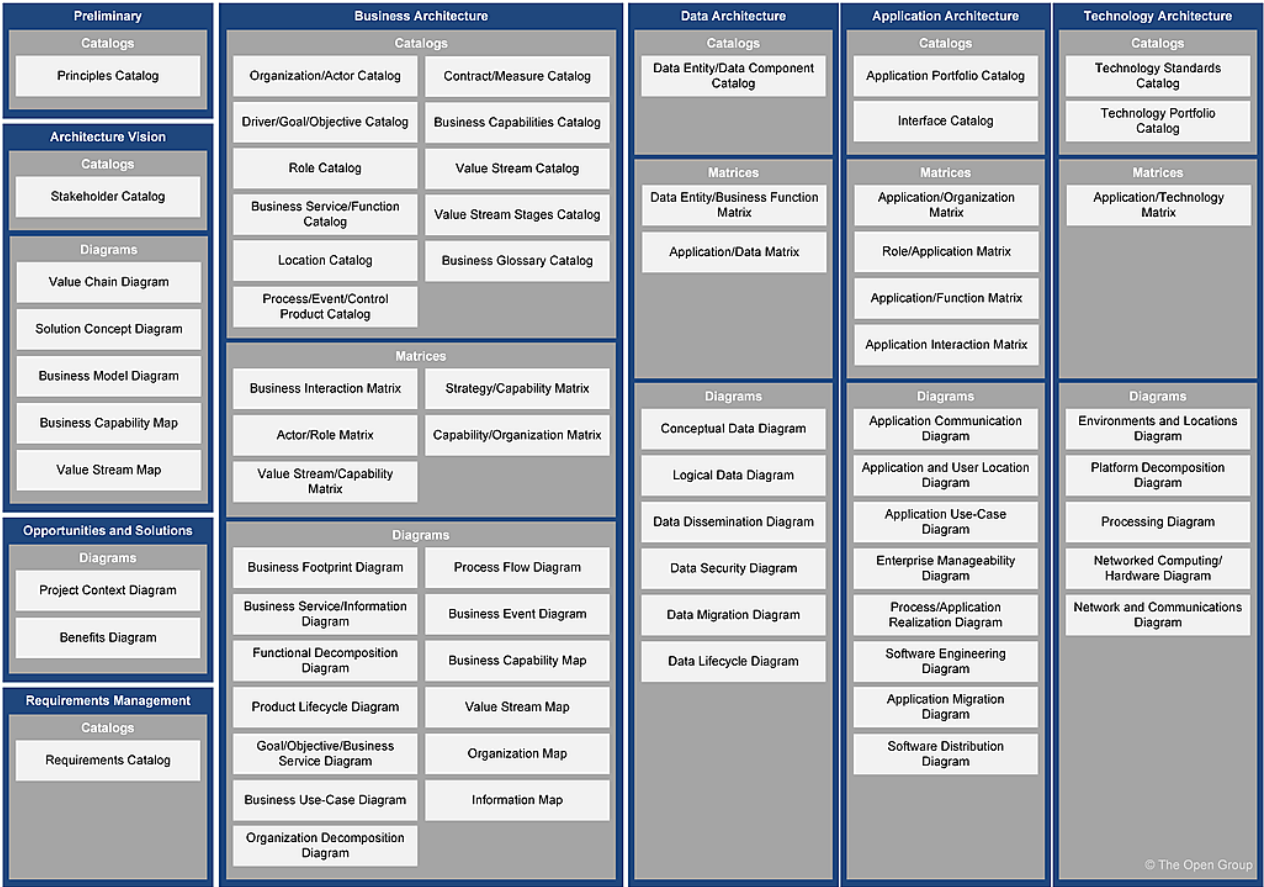


Рисунок 3. Онтологическая модель Базовой Архитектуры TOGAF 10

Базовая Архитектура, предписываемая стандартом, делится на четыре основных домена (слоя):

1. Бизнес-архитектура, определяющая цели, стратегии, формы организации бизнес-процессов и методы управления.
2. Архитектура данных, описывающая структуру данных и потоков информации, продуцируемых и используемых в рамках деятельности предприятия.
3. Архитектура приложений, представляющая собой схему отдельных программных компонентов, их взаимосвязь с бизнес-процессами предприятия и информацией, продуцируемой в рамках выполнения функций.

4. Технологическая архитектура, описывающая реализацию инфраструктуры (стандарты, библиотеки, технические возможности аппаратного и программного обеспечений) для организации взаимодействия бизнес-процессов, данных и приложений.

Описание доменов архитектуры регламентируется через формирование строительных блоков, которые в свою очередь должны быть представлены в виде каталогов сущностей, матриц связей и различного рода структурно-функциональных диаграмм, наиболее полно отражающих перечень и последовательность взаимодействия компонентов. Выбор необходимых диаграмм во многом определяется исходя из базы стандартов, представляющей на начальном этапе концептуальное описание компании через те компоненты (базовые строительные блоки), которые наилучшим образом характеризуют предметную область – перечень требований, внутренних и внешних услуг-сервисов компании. В качестве архитектурного стиля программной реализации стандартом рекомендуется использовать микросервисный подход – деление компонентов до атомарных функциональных блоков.

В целом, стандарт представляет собой всеобъемлющий и универсальный подход к проектированию архитектур бизнес-процессов, что, однако, нельзя считать в полной мере релевантным подходом для каждого отдельно взятого случая. Безусловно составление наиболее полного представления модели архитектуры с различных видов является необходимым условием успешной модернизации предприятия и в особенности его цифровой трансформации. Однако данный стандарт не учитывает особенности проектирования систем Индустрии 4.0. В первую очередь стандарт не задает явного целеполагания о построении автономно функционирующих предприятий, что, очевидно, упускает из виду специфические особенности создания систем подобного рода. TOGAF 10 рекомендует использование большого числа описательных элементов представления строительных блоков архитектуры, однако не предлагает каких-либо способов их согласования, что в особенности важно для поддержания корректной связи между весьма разнотипными диаграммами. Более того использование традиционных нотаций графического моделирования, хоть и может быть применимо для описания архитектуры, остается трудоемким процессом и задает высокий порог вхождения узкоспециализированным сотрудникам предприятий, чье участие в составлении модели архитектуры является критически необходимым. Концептуальная терминология и соответствующие способы формального представления архитектуры в стандарте ориентированы на широкий спектр деятельности организаций, что также может усложнять его адаптацию и применение для производственных предприятий и, в частности,

горнодобывающих производств. Как итог, стандарт не предписывает каких-либо обязательных специфических требований к назначению, составу функциональных и инфраструктурных компонентов и показателям качества системы в парадигме концепции Индустрии 4.0, что даёт слабое представление отличия архитектур автоматизации от архитектур цифровой трансформации.

Стандарт «IEC PAS 63088:2017 Smart manufacturing – Reference architecture model industry 4.0 (RAMI4.0)» и его отечественный перевод «ГОСТ Р 59799-2021 Умное производство. Модель эталонной архитектуры Индустрии 4.0» на сегодняшний день признан ведущим руководством в области построения архитектур промышленных предприятий в парадигме Индустрии 4.0. Данный стандарт регламентирует основную терминологию, которой необходимо придерживаться при проектировании цифровой трансформации предприятий, а также две ключевых модели – непосредственно модель эталонной архитектуры Индустрии 4.0 (RAMI4.0. Рисунок 4) в виде структурированного описания взаимосвязи компонентов предприятия, и эталонную модель компонента Индустрии 4.0, представляющую из себя цифровое онтологическое описание функционального блока, задействованного в реализации бизнес-процессов.

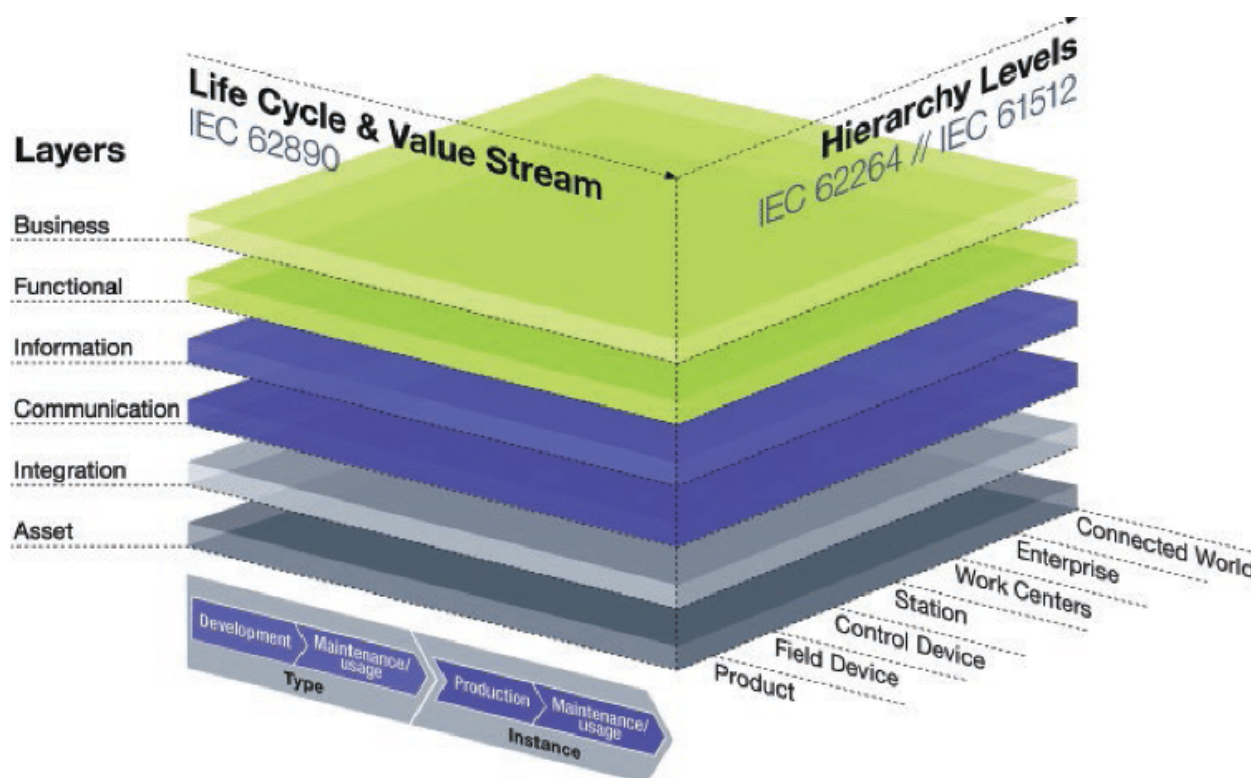


Рисунок 4. Эталонная модель архитектуры Индустрии 4.0 – RAMI4.0

Эталонная модель архитектуры представляет из себя три оси измерений предприятия:

1. Уровни иерархической организации, описывающий промышленное предприятие в контексте объектов физического мира: производимой продукции, элементарных сенсорных и исполнительных устройств, устройств управления, технологического оборудования, центров управления процессами, предприятия в целом и взаимодействия с внешним «подключенным» миром.
2. Жизненный цикл продукции, описывающий цепочку формирования добавочной стоимости, включая разработку форм типовой продукции, производство конкретного экземпляра, их непрерывные поддержку и обслуживание.
3. Архитектурные слои, описывающие:
 - физические объекты – ресурсы, используемые в ходе выполнения работ;
 - интеграцию физических объектов в цифровое пространство;
 - способы информационного взаимодействия;
 - продуцируемые данные;
 - набор функциональных задач;
 - перечень и организацию бизнес-процессов.

Основополагающими компонентами для описания архитектуры являются физические объекты реального мира, в качестве которых могут выступать различные сенсорные и вычислительные устройства, технологическое оборудование, программные информационно-аналитические системы и т.д., выполняющие определенные услуги в рамках сервис-ориентированной парадигмы. При проектировании предполагается иерархическая вложенность компонентов и их рекомбинация для формирования новых. Описание архитектуры компонента предполагает разработку оболочки администрирования в виде манифеста – метаданных, описывающей целевые, функциональные, информационные и иные требуемые параметры существования компонента в цифровом мире для регуляции его работы посредством диспетчера компонентов. Для описания элементов архитектур в виде каких-либо доменов рекомендуется использовать их модельное представление со сквозной взаимосвязью между всеми осями, однако стандартом не предлагается каких-то конкретных способов описания такой архитектуры вообще. В целом же стандарт предлагает использование широко известных (хорошо зарекомендовавших) методик, технологий и инструментов, включаемых в Индустрию 4.0, но не формализует какую-либо их фактическую реализацию и конечные требования к виду всей системы, в том числе, её исполнение в автономно функционирующем виде. Иными словами, данный стандарт также даёт слишком общее представление об архитектуре систем Индустрии 4.0, оставляя процесс проектирования, выбор инструментария, формулирование

требований и назначения на самостоятельное усмотрение каждого участника цифровой трансформации.

Исходя из анализа стандартов следует отметить близкую взаимосвязь базовых принципов архитектурирования цифровой трансформации предприятий с основными позициями требований к разработке и модернизации автоматизированных систем в соответствии с ГОСТ 34.602-2020 «Информационные технологии. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы». Основным отличием стандартов от классических подходов АСУ выступает введение современной терминологии жизненного цикла программных продуктов для описания элементов системы, что, в целом, нельзя считать существенно полезным свойством с позиций практической реализации построения автономно функционирующих производств. Предложения или обязательные требования по составу и назначению функциональных, информационных или технико-инфраструктурных компонентов, предписываемых Индустрией 4.0, отсутствуют или носят общеописательный характер, что усложняет понимание по их применению, инструментам проектирования и разработки для сложной крупномасштабной производственной системы. Однако некоторый синтез данных стандартов может позволить использовать отдельные подходы к формализации и представлению модели архитектуры при проектировании цифровой трансформации горнопромышленных предприятий.

В качестве ещё одного важного обобщающего вывода, следует отметить направленность стандартов на формирование онтологического представления предприятия, обеспечивающего универсальность и согласованность знаний на любом этапе жизненного цикла, что, в целом, можно считать оправданным и обоснованным подходом. В то же время, онтологическое проектирование архитектур, нацеленных на представление знаний автономно функционирующей системе о самой системе, должно предполагать возможность манипулирования такими знаниями. Предлагаемые же способы проектирования моделей архитектур в рассматриваемых стандартах не имеют прямых способов проведения вычислительных операций над понятиями и отношениями между ними, что при определенных условиях можно считать недостатком.

1.2.2. Модели архитектур программных систем

В довольно общем смысле понятие архитектуры программного обеспечения представляет из себя n-звенную структуру, где под звеном понимаются абстрагированные слои взаимодействующих частей программы, отвечающих за представление, обработку, хранение данных и управление бизнес-логикой процедур [80, 81]. При этом следует

отметить, что в настоящий момент отсутствует унифицированная и строго регламентированная терминология для описания архитектур программного обеспечения [82]. Главным образом это можно объяснить высокой прикладной направленностью и диверсифицированностью технологий и языков программирования, индивидуальные методологические аспекты развития которых не ориентировались на создание общей фундаментальной научно-терминологической базы. Таким образом сегодня общие термины, такие как, например функция или класс, могут иметь отличные определения для отдельных подходов в программировании, а некоторые термины не имеют ясных и четких определений как таковых. В этой связи в рамках настоящего исследования определим следующее:

Архитектура программного обеспечения (программной системы, приложения) – это структура изолирования и связи слоев и классов программного кода, реализующих полный набор функций.

Слой программного обеспечения (программной системы, приложения) – это программно-технически и/или физически изолированная часть кода, реализующая набор классов по представлению, обработке, хранению данных или управлению бизнес-логикой.

Класс программного обеспечения (программной системы, приложения) – это изолированная структурная единица программного кода, реализующая процедуру представления, обработки, хранения данных или управления бизнес-логикой.

Функция программного обеспечения (программной системы, приложения) – это логически связанный уникальный набор классов из разных слоев архитектуры.

Таким образом фактически конкретная архитектура ПО регламентирует число слоев, распределение и взаимосвязь классов в этих слоях, а набор классов логически формируют функцию. Так же необходимо отметить, что и слой, и функция, и класс могли бы называться компонентом архитектуры, однако, очевидно, что это привело бы к терминологической путанице.

Рассмотрим существующие наиболее устойчивые подходы к исполнению архитектур ПО с позиции распределения слоев, классов и функций [81, 82], а именно:

1. **Монолитная** (Рисунок 5). Представляет из себя наиболее устаревшую архитектуру со строгим отсутствием физического разделения классов и слоев, и низкой программной делимостью классов, а также отсутствием средств внешних интеграций. Преимуществом данной архитектуры является её высокая производительность, а недостатком является высокая трудоемкость действий по обеспечению требований к модифицируемости, масштабируемости и расширяемости, т.к. любые изменения монолитной архитектуры требуют внесения

изменений в исходный код всех связанных классов. Также следует иметь в виду, что монолитная архитектура предполагает её целостное развертывание на одном вычислительном ресурсе, что невозможно для децентрализованных многокомпонентных производственных систем, действующих сразу в нескольких исполнительных средах.

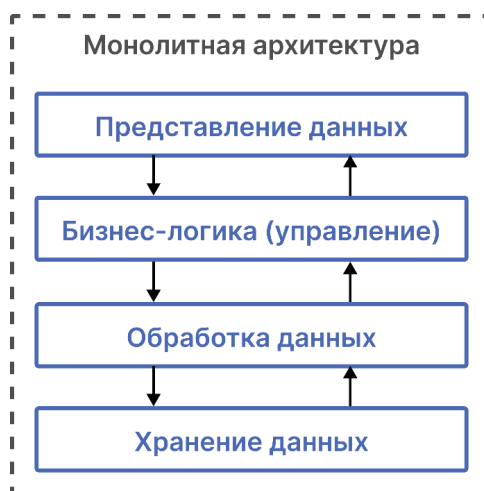


Рисунок 5. Схематичное представление монолитной архитектуры

2. **Клиент-серверная** (Рисунок 6). Наиболее распространенный тип архитектуры программных систем, предполагающей физическое разделения слоев на сервер (выполняющего ключевые процедуры по обработке и хранению данных и управлению бизнес-логикой) и тонкий (только представление) или толстый (с возможностями локальной реализации всех классов) клиенты, а также отсутствием прямых средств интеграции внешних систем. Данный тип архитектур применим для распределенных систем, действующих в двух и более физических средах и относительно эффективен с точки зрения производительности из-за отсутствия избыточного дублирования классов на сервере. Однако простота модифицируемости, масштабируемости и расширяемости для крупномасштабных систем достигается только в клиентской части. Любые внесения изменений в архитектуру требуют сложных и ресурсоемких методов поддержки жизненного цикла для обеспечения непрерывности работы системы.

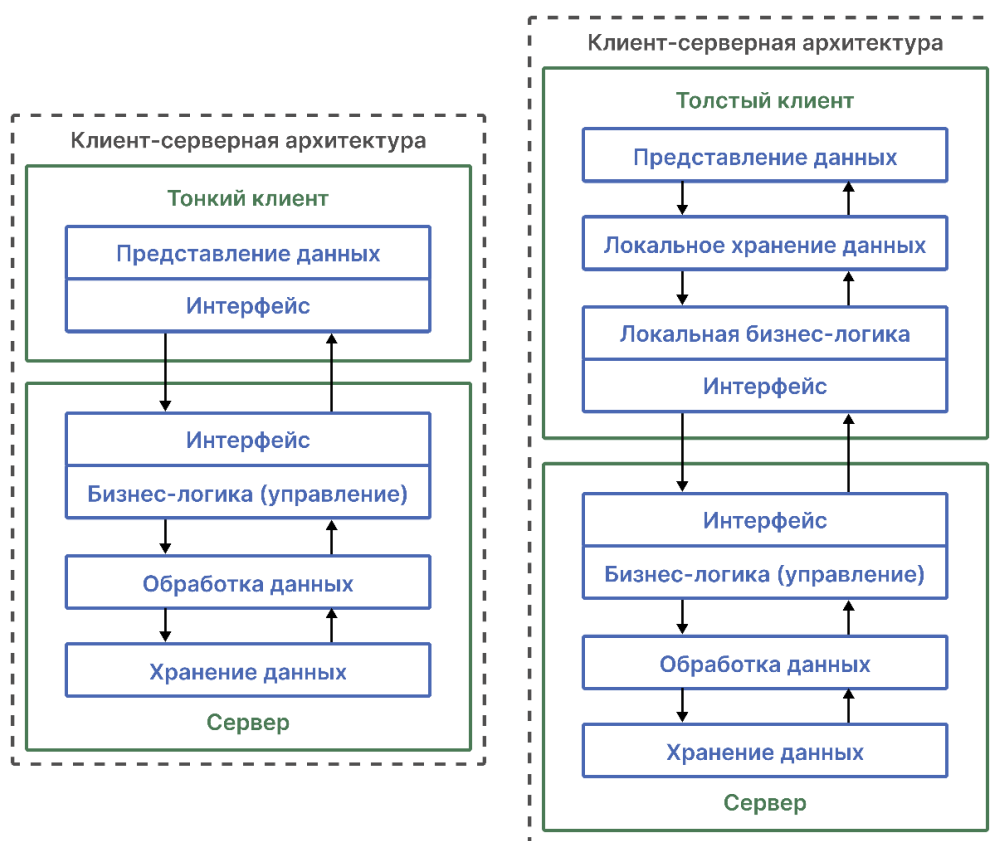


Рисунок 6. Схематичное представление клиент-серверной архитектуры

3. **Многослойная или многоуровневая** (Рисунок 7). Представляет из себя более сложно сконфигурированную модификацию клиент-серверной архитектуры. В данной архитектуре все классы слоев программно разделены в рамках отдельных функций. В целом, данная архитектура лучше поддерживает все необходимые требования для крупномасштабных систем, однако также не имеет средств прямой интеграции с внешними системами. Внесение каких-либо изменений в архитектуру является трудозатратным процессом.

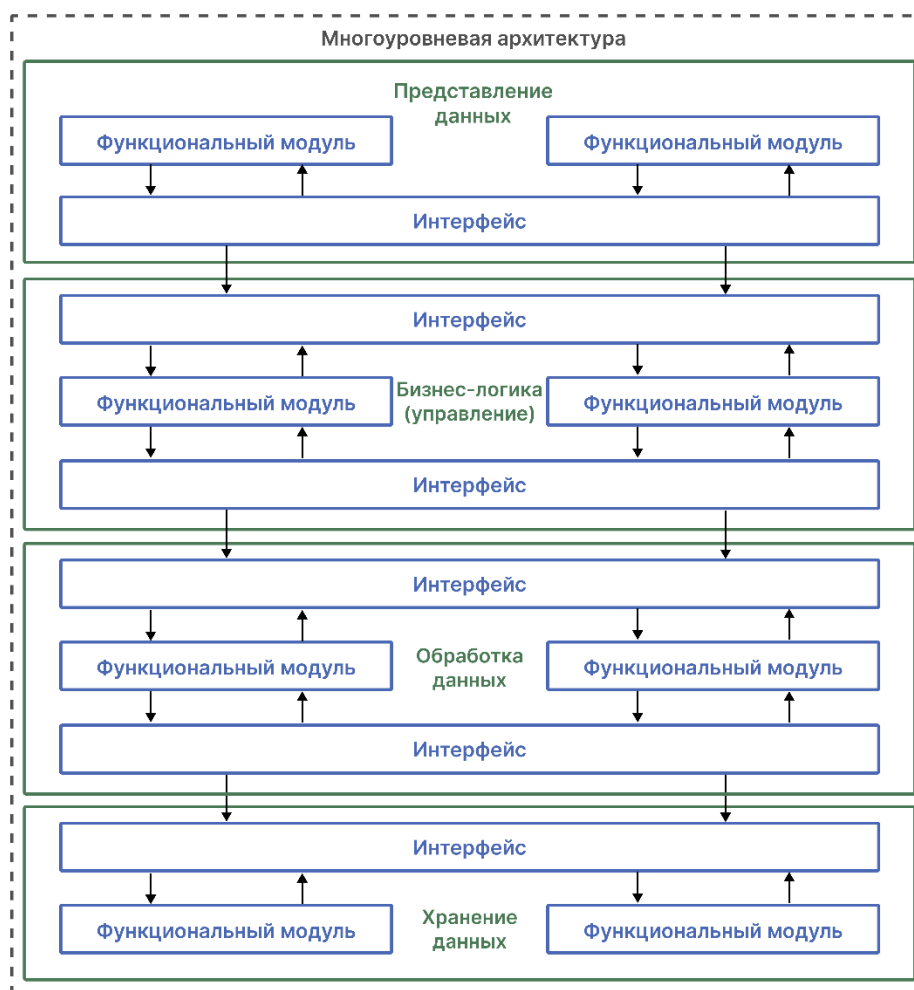


Рисунок 7. Схематичное представление многослойной архитектуры

4. **Сервис-ориентированная** (Рисунок 8). Одна из более современных и, в то же время, менее структурно формализованных архитектур, ориентированная на представление компонентов через понятие «услуг» (сервисов). С точки зрения программной реализации, основным отличием сервис-ориентированной архитектуры является интеграция программных приложений в событийно-ориентированном режиме через унифицированные программные интерфейсы и протокол – интеграционную сервисную шины предприятия (Enterprise Service Bus), а также сложных программных механизмов оркестрации ресурсов. В данном типе архитектуры предполагается большая гибкость и независимость межсистемного взаимодействия, что, в целом относительно хорошо поддерживает модифицируемость, масштабируемость, расширяемость и интероперабельность всей производственной системы, ограничиваемых только структурой самих интегрированных программных приложений. Однако, ввиду не контролируемой изолированности слоев и классов,

повышаются риски дублирования функций и однотипных вспомогательных служб, что существенным образом сказывается на производительности системы.

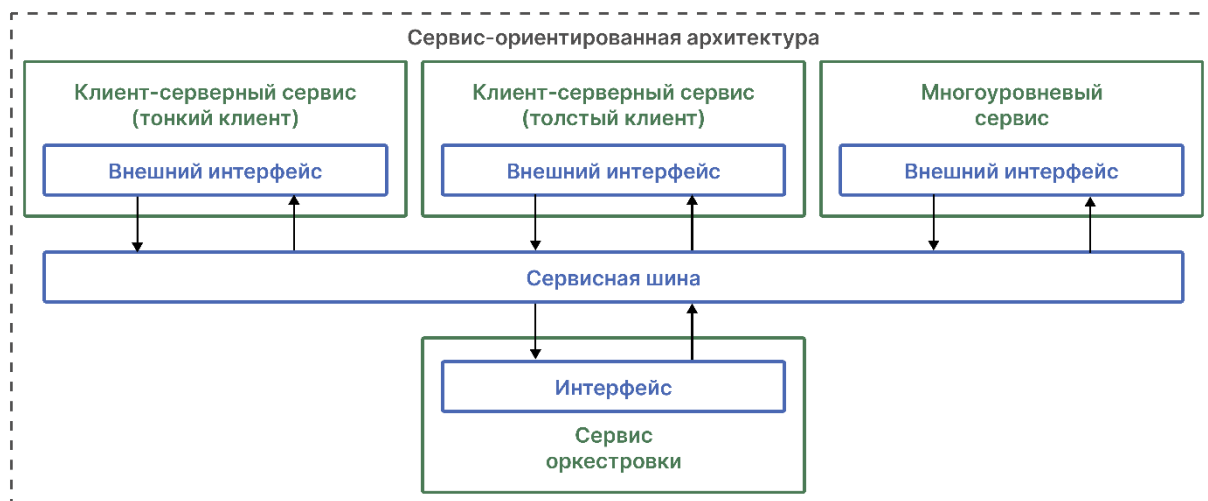


Рисунок 8. Схематичное представление сервис-ориентированной архитектуры

5. **Микросервисная** (Рисунок 9). Наиболее новый и относительно перспективный вид архитектуры, подразумевающий под собой «многомонолитную» структуру с горизонтальной интеграцией, где каждый монолит представляет из себя независимое программное приложение, выполняющее небольшую (атомарную) и узкоспециализированную задачу в событийно-ориентированном режиме. Взаимодействие осуществляется на уровне функций и происходит на основе унифицированных протоколов с использованием посредников брокеров сообщений (Message Broker). В такой архитектуре предполагается высокая изолированность (принцип «черных ящиков») микросервисов, что наилучшим образом поддерживает требования к модифицируемости, масштабируемости и расширяемости всей системы. Наличие посредников, обеспечивающих простоту взаимодействия (и соответственно замену и интеграцию новых) функций, существенно повышает интероперабельность всей системы, но повышает риски в области информационной безопасности. Необходимость включения в каждый микросервис компонентов, отвечающих за его самостоятельное функционирование, создает серьёзные проблемы в области производительности архитектуры. Внесение же каких-либо изменений в такую архитектуру может происходить наименее трудозатратным способом и с минимальными потерями в части нарушения непрерывности процессов.

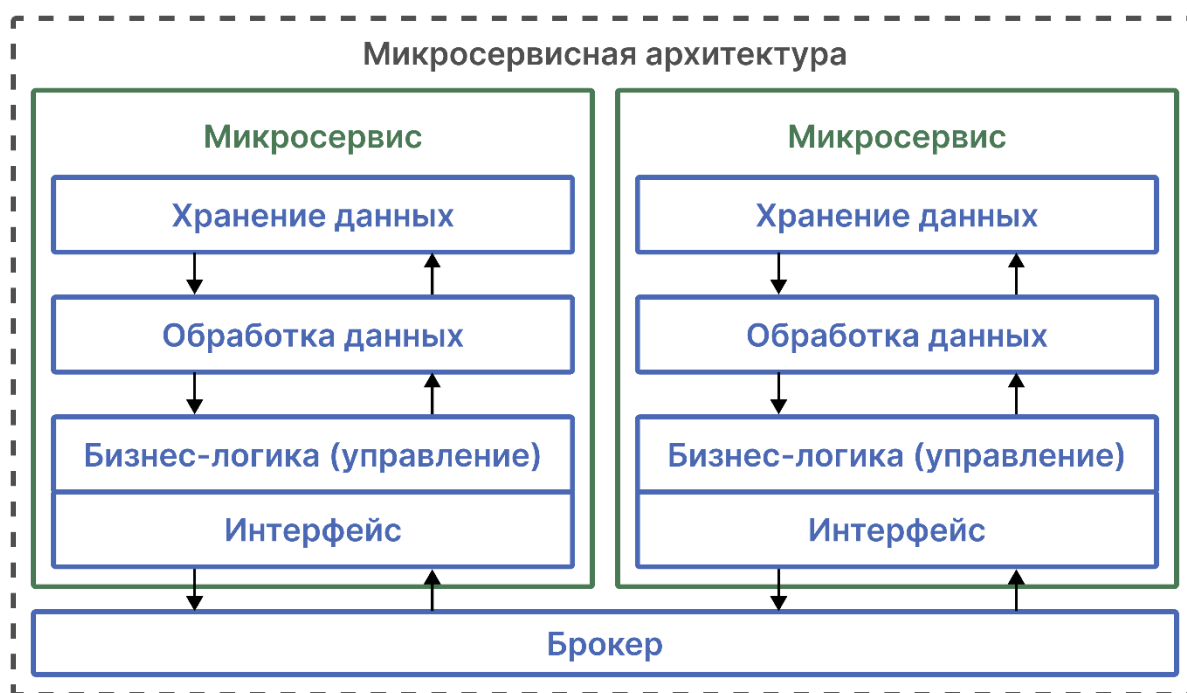


Рисунок 9. Схематичное представление микросервисной архитектуры

В настоящий момент в той или иной мере все рассмотренные типы архитектур применяются при построении различного рода программных систем. Однако не существует каких-либо явных формальных способов их унифицированного описания и оценки для возможности анализа и сопоставления эффективности применения. Выбор конкретного типа архитектуры, как правило, осуществляется на основе экспертных мнений на стадии проектирования Жизненного Цикла архитектуры системы, а оценка эффективности осуществляется по результатам анализа тестирования и функционирования в действующей среде.

1.2.3. Задачи управления архитектурой

Процессы управления архитектурами систем, в целом, также не имеют строго единообразного представления и унифицированных методов. В зависимости от типа рассматриваемой системы – предприятия или программного приложения – задачи и методы управления архитектурой формулируются с учётом специфики Жизненного Цикла соответствующего объекта [76, 83-84]. Так, жизненный цикл архитектуры предприятия чаще направлен на формализацию бизнес-целей, требований, задач; их непрерывное выполнение и анализ эффективности для выработки стратегий улучшения показателей [67-71, 78, 79]; в то время как жизненный цикл архитектур программ предполагает вариативную последовательность выполнения процедур анализа, проектирования, разработки, тестирования, внедрения и поддержки структур кода, реализующих набор функций [73, 80-

84]. Однако, несмотря на некоторые различия в целеполагании и прикладной направленности процессов, можно вывести общность и согласованность подходов управления архитектурой инвариантно типу системы. Используя в качестве наиболее общей базовой модели жизненного цикла архитектуры фреймворк TOGAF 10 [69-70], в Таблице 1 представлено аналитическое сопоставление целей и результатов задач управления архитектурами предприятия и программ.

Таблица 1

Аналитическое сравнение задач жизненного цикла архитектур

Задача	Предприятие		Программа	
	Цель	Результат	Цель	Результат
Анализ	Выявление наиболее полного и точного состояния, целей, требований, условий и ограничений по модификации бизнес-процессов	Модель архитектуры «as is»	Выявление наиболее полного и точного состояния, целей, требований, условий и ограничений по модификации функциональности	Модель архитектуры «as is»
Проектирование	Планирование (оптимизация) эволюции бизнес-процессов	Модель архитектуры «to be»	Планирование (оптимизация) эволюции функциональности	Модель архитектуры «to be»
Разработка	Декомпозиция задач и реализация плана эволюции бизнес-процессов	Архитектура «to be»	Декомпозиция задач и реализация плана эволюции функциональности	Архитектура «to be»
Тестирование	Проверка соответствия полученной архитектуры запланированной модели	Оценка качества и соответствия модели	Проверка соответствия полученной архитектуры запланированной модели	Оценка качества и соответствия модели
Внедрение	Имплементация архитектуры и эволюционный переход состояния	Архитектура «as is»	Имплементация архитектуры и эволюционный переход состояния	Архитектура «as is»
Поддержка	Мониторинг состояния и определение необходимости улучшений бизнес-процессов	Оценка качества архитектуры	Мониторинг состояния и определение необходимости улучшений функциональности	Оценка качества архитектуры

Исходя из Таблицы 1 можно отметить, что реализуемые задачи имеют общую принципиальную направленность целей и результатов, однако де-факто описывают состояния системы с различных точек зрения. Однако, в условиях цифровой трансформации предприятий, где реализация бизнес-процессов должна осуществляться исключительно с использованием программных компонентов, очевидна необходимость их формального сопоставления и приведения к эквивалентности. В таком случае в качестве задач управления архитектурой в контексте настоящего исследования можно сформулировать следующее:

1. Формирование полной, точной и непротиворечивой модели «as is», регламентирующей текущее состояние архитектуры системы и требования, условия и ограничения по её эволюционной модификации.
2. Формирование оптимальной модели архитектуры «to be», удовлетворяющей заданным требованиям, условиям и ограничениям эволюционной модификации.
3. Реализация архитектуры «to be» в соответствии с оптимальной моделью эволюционной модификации.
4. Оценка качества соответствия разработанной архитектуры с модельным планом.
5. Реализация эволюционного перехода архитектуры из состояний «as is» - «to be» - «as is».
6. Мониторинг и оценка качества архитектуры системы.

Необходимо также отметить, что инвариантной декларативной целью решения таких задач является минимизация трудозатрат на жизненный цикл архитектуры без потерь её качества.

1.3 Методы оценки и анализа архитектур систем

1.3.1 Методы оценки качества архитектур предприятий

В области оценки качества архитектур предприятий основным методом оценки является экспертный анализ, предусмотренный, например, фреймворком TOGAF 10 [69-70, 76, 83]. В рамках экспертного анализа осуществляется сопоставление моделей «as is» и «to-be» архитектур, а результатом является вывод о степени их соответствия. При этом очевидно, что такой подход имеет ряд существенных ограничений связанных с исходным качеством формируемых моделей, а также отсутствием строго формализованных количественных методов их инструментального сопоставления. Так составление каталогов, матриц и различного рода графических схем [74, 75], хоть и является удобной формой абстрактного представления архитектуры для экспертного анализа, влечёт за собой высокие риски возникновения ошибок различного рода. К таким ошибкам можно отнести неполноту исходного содержания моделей; противоречивость, тавтологичность и неоднозначность семантических конструкций при абстрагировании знаний об архитектуре; несоответствие таких семантических конструкций для описания одних и тех же сущностей на различных этапах жизненного цикла и др. Критерий же степени соответствия моделей также является нестрого формализованным и может интерпретироваться неоднозначно, а процедура экспертного анализа нести риски возникновения ошибки при невнимательности или низкой квалификации эксперта. Наличие подобных рисков может приводить в свою очередь к частичному или полному несоответствию реализуемых архитектур, а как следствие необходимости переделок и повышению стоимости жизненного цикла [84].

Таким образом на сегодня не существует релевантных методов анализа качества архитектур предприятий, отвечающих требованиям к построению автономно функционирующих промышленных систем. Однако, следует отметить главную и важную особенность стандартов и фреймворков в данной области – это попытка формализовать связь между принципиально разными компонентами предприятий, такими как абстрактные концепты организации бизнес-процессов (регламенты, условия, требования, роли персонала и т.п.), реально существующими объектами физического мира (датчики, оборудование, продукция, ресурсы, инфраструктурные элементы), информацией (формальными и неформальными экспертными знаниями, цифровыми, аналоговыми и нейрофизиологическими сигналами, измеряемыми параметрами и др.) и программными сущностями, реализующими набор функциональных задач управления [39, 48-51, 55, 67-72].

Иными словами, можно сказать, что в контексте построения моделей архитектуры предприятия, речь идёт о формировании знаний о системе с целью построения механизмов управления, то есть фактически – об онтологическом моделировании [85-89]. В таком случае модель архитектуры можно определить как некоторую онтологию, которая может быть трансформирована в Базу Знаний с механизмами доказательства непротиворечивости и истинности заложенных в неё высказываний в качестве инструмента первичного анализа качества.

Создание онтологий и Баз Знаний для различных предметных областей занимает значительное место в истории искусственного интеллекта [90-94], и, как правило, связано с построением экспертных систем, способных помогать профессионалам при решении различных проблемных задач. Так для горнодобывающих производств, в качестве примера, можно привести ряд работ по построению подобного рода формальных систем: для классификации типов и свойств угледобывающего оборудования [86], согласования терминологии IT-систем и эксплуатируемого горнодобывающего оборудования [87, 88] или управления техническими осмотрами и ремонтами оборудования [89]. По большей степени, несмотря на высокую степень детализации «таксономии», разрабатываемые онтологии направлены на решение узкоспециализированных проблем и получаемые на их основе экспертные системы предназначены исключительно для взаимодействия с персоналом в процессах управления предприятиями. Согласование же таких специализированных онтологий между собой или с более общими онтологиями знаний может быть достаточно сложным или невозможным вовсе, что ограничивает их научно-практическую значимость.

В то же время сами понятия онтология и База Знаний могли бы быть использованы, с одной стороны, для формализации и проверки знаний при построении модели архитектуры предприятия, а с другой, быть использованы как инструменты насыщения (обучения) системы знаниями о самой себе для построения автономно функционирующей кибер-физической индустриальной системы. Одним из наиболее выразительных, гибких и перспективных инструментов – языков построения онтологий, сегодня является OWL (Open World Language или Web Ontology Language) [95, 96], рекомендованный для создания семантических сетей Web 3.0 и основанный на дескрипционной логике – высоко выразительном и разрешимом формальном логическом языке [97, 98]. Основу дескрипционной логики и OWL составляют: одноместные и двуместные атомарные аксиоматические высказывания – «концепты» и «роли»; иерархические структуры (классы или категории) взаимосвязей между высказываниями; сводимость классов и суперклассов к их экземплярам с возможностью присвоения значений (данных); конечный набор свойств отношений между такими высказываниями (транзитивность, рефлексивность и др.),

подчиняющиеся базовой алгебре логики; а также набор «ризонеров» (с англ. reasoner) [98] – развитых механизмов логического вывода для доказательства полноты, непротиворечивости, консистентности онтологии, выведения скрытых (явно не задекларированных) аксиом и, в целом, диалогового взаимодействия с Базой Знаний.

1.3.2 Методы оценки качества архитектур программных систем

В области методов оценки качества архитектур программных систем существует большое число качественно-количественных метрик и инструментальных подходов с различной степенью пригодности. Одним из ключевых подходов является модель качества программного продукта Международного стандарта ISO/IEC 25010:2023 «Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Product quality model» (ГОСТ Р ИСО/МЭК 25020-2023 «Системная и программная инженерия. Требования и оценка качества систем и программной продукции (SQuaRE)». Основные принципы измерения качества). Данный стандарт определяет эталонную модель, включающую восемь характеристик, каждая из которых декомпозируется на подхарактеристики:

- функциональная пригодность (functional suitability);
- производительности (performance);
- совместимость (compatibility);
- удобство использования (usability);
- надёжность (reliability);
- безопасность (security);
- сопровождаемость (maintainability);
- гибкость (flexibility).

В то же время стандарт декларирует исключительно сами метрики, но не предписывает способы их вычисления. Значимость отдельных характеристик варьируется на различных этапах жизненного цикла архитектуры: на этапе разработки приоритетны функциональная пригодность и тестируемость; на этапе эксплуатации – надёжность, производительность и безопасность; на этапе поддержки – сопровождаемость и гибкость. Наиболее существенными метриками с точки зрения эволюционного развития архитектур являются функциональная целостность, модульность, модифицируемость, расширяемость и масштабируемость, определяющие способность архитектуры к развитию. Однако перечисленные характеристики лишены в стандарте прямых способов какой-либо количественной оценки и сопоставления.

Также применяются более формальные способы экспертного анализа – сценарные методы оценки качества архитектур. Метод SAAM (Software Architecture Analysis Method), предложенный Р. Казманом [77, 83] и соавторами в Институте программной инженерии Университета Карнеги-Меллон, стал первым систематическим подходом к экспертной оценке архитектуры на основе сценариев. Процедура SAAM включает шесть шагов:

- (1) разработка сценариев заинтересованными сторонами;
- (2) описание архитектуры;
- (3) классификация сценариев на прямые (реализуемые без изменения архитектуры) и косвенные (требующие модификации);
- (4) оценка каждого косвенного сценария — определение компонентов, подлежащих изменению;
- (5) выявление взаимодействия сценариев (scenario interaction), возникающего, когда несколько косвенных сценариев затрагивают один и тот же компонент, что свидетельствует о потенциальном архитектурном риске;
- (6) общая оценка архитектуры.

Ограничением SAAM является субъективность экспертных суждений и отсутствие количественного результата.

Метод ATAM (Architecture Tradeoff Analysis Method) [76], развивающий идеи SAAM, включает четыре фазы и девять шагов. Центральным инструментом является дерево полезности атрибутов качества (Quality Attribute Utility Tree), в котором абстрактные характеристики качества декомпозируются до конкретных сценариев с приоритизацией по двум осям: важность для бизнеса и техническая сложность. Результатами ATAM являются: риски (архитектурные решения, потенциально негативно влияющие на атрибуты качества), не-риски, точки чувствительности (параметры, к которым один атрибут качества особенно чувствителен) и точки компромисса (параметры, влияющие на несколько атрибутов в противоположных направлениях). Методология ATAM рекомендует участие 15–30 экспертов, что с одной стороны может обеспечивать объективизацию оценок, но с другой достаточно трудноосуществимо.

Большое внимание уделяется оценкам качества структуризации программного кода реализации функций. Фундаментальные принципы структурного проектирования программных систем – сцепление (coupling) и связность (cohesion), формализованные Стивенсом, Майерсом и Константайном [99] и систематизированные Йорданом и Константайном [100]. Метрика связности оценивается по семиуровневой шкале (от функциональной до случайной), сцепление – по шестиуровневой (от связи по данным до связи по содержимому). При этом данный метод предполагает представление структуры

кода программы в виде графа, в котором нодами являются отдельные единицы кода, а рёбра – их логико-программной связью. Проектным идеалом является высокая связность и низкое сцепление между нодами графа – разделение графа на подграфы связанных одним ребром. Существенным ограничением является порядковый характер шкал оценки, при которой значения не допускают арифметических операций, что не позволяет получать понятную количественную оценку.

Другой, количественный, метод оценки качества структур объектно-ориентированных программ – метрики Чидамбера–Кемерера, представляющих набор из шести метрик [101]:

- CBO (Coupling Between Objects) – число классов, с которыми данный класс связан;
- WMC (Weighted Methods per Class) – взвешенная сумма сложностей методов класса;
- DIT (Depth of Inheritance Tree) – глубина наследования методов;
- NOC (Number of Children) – количество потомков;
- RFC (Response For a Class) – мощность множества откликов;
- LCOM (Lack of Cohesion in Methods) – отсутствие связности метода.

Для каждой из метрик определены соответствующие расчётные формулы и установлены эффективные пороги, однако их совместная оценка не возможна, а величины порогов являются эмпирическими. Тем не менее валидация [102] подтвердила значимую корреляцию CBO, RFC и WMC с плотностью дефектов программного кода.

Также существует метод оценки структурной сложности программного кода – цикломатическая сложность Маккейба [103], рассчитываемой на основе графа потока управления по линейной зависимости вида:

$$V(G) = E - N + 2P, \quad (1)$$

где E – число рёбер графа потока управления, N – число вершин, P – компоненты связности. $V(G)$ равна числу линейно независимых путей.

Для предложенной оценки так же установлена шкала качественного измерения и эмпирический порог эффективности ≤ 10 , что нельзя считать в полной мере объективным подходом.

Ещё одним методов количественной оценки качества структуры программного кода, сравнения сложности алгоритмов и прогнозирования времени разработки и числа ошибок

является метрика Холстеда [104]. Данная метрика использует количество операторов и операндов и предусматривает следующие оценки:

- словарь программы: $n = n_1 + n_2$, где n_1 – количество уникальных операторов (ключевых слов, знаков операций), а n_2 – количество уникальных операндов (переменных и констант);
- длина программы: $N = N_1 + N_2$, где N_1 – общее количество операторов, а N_2 – общее количество операндов;
- объём программы: $V = N \log_2 n$;
- сложность программы: $D = \frac{n_1}{2} * \frac{N_2}{n_2}$;
- трудоёмкость: $E = D * V$.

Предложенные оценки являются достаточно универсальными, однако критика [105] показала, что предсказательная способность метрик не превышает простого подсчёта строк кода, что не может однозначно характеризовать качество программной системы.

1.3.3 Методы анализа сложных систем

Исходя из приведенного анализа разделов 1.3.1-1.3.2, можно отметить, что на сегодняшний день наблюдается разрыв между способами формального сопоставления архитектур предприятий и архитектур программных систем, несмотря на тот факт, что любое программное обеспечение, в сущности, своей создается для автоматизации задач предприятий, а следовательно, должна наблюдаться их эквивалентность и существовать формализованный переход от одного вида к другому. Предлагаемые же методы оценки качества архитектур в большей степени являются экспертными и не позволяют делать однозначные объективные количественные выводы о выборе лучшего варианта архитектуры инвариантно к любому предприятию. Более того, в случае цифровой трансформации горнопромышленных предприятий, речь идёт о сложных крупномасштабных системах с гетерогенными компонентами архитектуры, требующих при этом явного сопоставления, а следовательно, унифицированного представления.

Можно выделить следующие особенности методов, которые могли бы способствовать формированию методов оценки качества архитектур сложных крупномасштабных промышленных систем:

- представление архитектуры в виде графа с динамической структурой, модифицируемой в рамках цифровой трансформации с учётом требований, условий и ограничений функционирования конкретного предприятия;

- оценка сложности как исходной структуры такого графа, так и вариантов его модификаций, а также сложности самих процессов модификации и поддержки, т.е. сложности жизненного цикла, что прямо или косвенно влияет на стоимость трудозатрат.

В этой связи необходимо отметить существующие способы представления сложных систем в виде графовых моделей, методов оценки сложности графов, а также методов их модификаций.

В теоретико-информационном подходе к анализу сложных систем достаточно часто обращаются к классической мере неопределенности случайной величины – Энтропии Шеннона [106]:

$$H = - \sum_i P_i \log_2 P_i, \quad (2)$$

где P_i – вероятность i -го исхода в полной вероятности всех возможных исходов.

Шенноном доказана теорема единственности на основе аксиом непрерывности, монотонности и аддитивности группировки единственной функции. Рашевским [107] предложено использовать данную меру для оценки сложности графовых структур, что было в дальнейшем систематизировано и развито Мовшовицем [108-111], который предложил семейство теоретико-информационных мер сложности графов. Наиболее распространённый подход – энтропия на основе степенного распределения вершин:

$$H(G) = - \sum_i \left(\frac{d_i}{2m} \right) \log_2 \left(\frac{d_i}{2m} \right), \quad (3)$$

где d_i – степень i -й вершины, m – число рёбер графа, а величина $\frac{d_i}{2m}$ интерпретируется как вероятность того, что случайно выбранный конец ребра принадлежит вершине i .

Дехмер [112-115] обобщил данный подход, определив семейство графовых энтропий на основе произвольных информационных функционалов: $f: V \rightarrow \mathbb{R}^+$, частным случаем которого является степенная энтропия при $f(V_i) = d_i$.

В контексте программной инженерии энтропийные оценки также нашли применение в ряде направлений. Белادي и Леман [116] использовали понятие энтропии для формулировки законов эволюции программного обеспечения – в частности, второй закон Лемана утверждает, что сложность программной системы непрерывно возрастает в процессе развития, если не предпринимаются специальные меры по её снижению. Хассан

[117] применил энтропию изменений для предсказания дефектов в программных системах, показав статистически значимую корреляцию между распределением изменений по файлам и плотностью дефектов.

Вместе с тем необходимо подчеркнуть, что в перечисленных работах энтропия применялась для оценки сложности кода – распределения зависимостей между файлами, модулями, классами. Код, однако, не является эквивалентом архитектуры всей системы: архитектура описывает принципы организации системы на более высоком уровне абстракции с учётом бизнес-процессов, физических узлов размещения и др. Так что перенос энтропийного подхода с уровня кода на уровень архитектурного графа требует адаптации: необходимо определить, какие сущности выступают вершинами, какие – рёбрами, и какой смысл приобретает степень вершины в контексте конфигурации компонентов программной системы.

Помимо оценки статической сложности архитектуры, существенную роль играет оценка стоимости её изменения. В рамках жизненного цикла архитектура системы эволюционирует – добавляются новые компоненты, удаляются устаревшие, объединяются дублирующие, разделяются чрезмерно нагруженные. В условиях представления архитектуры в виде графа, каждое такое изменение может быть формализовано как операция редактирования. Понятие расстояния редактирования графов (Graph Edit Distance, GED) было введено независимо Бунке и Аллерманном [118-119] и Санфелиу и Фу [120] и определяет минимальную стоимость последовательности элементарных операций, преобразующих один граф в другой. К элементарным операциям редактирования относятся: вставка вершины, удаление вершины, подстановка (замена атрибутов) вершины, вставка ребра, удаление ребра, подстановка ребра. Каждой операции e_i сопоставляется стоимость $c(e_i) \geq 0$ посредством функции стоимости – расстояния редактирования:

$$GED(G_1, G_2) = \min_{(e_1, \dots, e_i) \in \mathcal{P}(G_1, G_2)} \sum_i c(e_i), \quad (4)$$

где $\mathcal{P}(G_1, G_2)$ — множество всех допустимых путей редактирования.

Вычисление точного GED является NP-трудной задачей [121]; для практических задач предложены приближённые методы [122-124], в том числе сведение к задаче о назначениях в двудольном графе с полиномиальной сложностью $O(n^3)$ [125].

1.4 Постановка задачи исследования

Проведённый в разделах 1.1–1.3 анализ позволяет сформулировать следующие обобщающие выводы. Современные горнопромышленные предприятия представляют собой сложные крупномасштабные пространственно-распределённые геотехнические системы с гетерогенной технологической средой. Исторически сложившаяся программно-техническая инфраструктура таких предприятий включает множество разнотипных специализированных систем (таких как АСУ ГТК, ГГИС, ERP), обладающих проприетарными архитектурами без унифицированных подходов к организации структурно-функционального и информационного взаимодействия.

Цифровая трансформация подобных предприятий в парадигме Индустрии 4.0 предполагает переход к автономно функционирующим кибер-физическим промышленным системам, что требует формирования модели архитектуры, обеспечивающей возможности самоорганизации и саморегуляции на основе знаний о ключевых компонентах и связях (технологических процессах и операциях, физических объектах, информации и акторов управления).

Анализ существующих стандартов и фреймворков (TOGAF 10, RAMI 4.0) показывает их направленность на онтологическое представление архитектуры предприятия, однако сами модели мало пригодны для формальных логических вычислений, а методы оценки качества архитектур предприятий носят преимущественно экспертный характер и основаны на субъективном сопоставлении моделей «as is» и «to be». При этом качество самих моделей может определяться полнотой, точностью и непротиворечивостью заложенных в них знаний, что, как следствие, может быть обеспечено современными средствами онтологического моделирования на языке дескрипционной логики OWL с использованием механизмов доказательства и логического вывода – рассуждателей.

Методы оценки качества архитектур программных систем включают как экспертные сценарные подходы (SAAM, ATAM), так и количественные метрики уровня программного кода (Чидамбера–Кемерера, Маккейба, Халстеда). Стандарт ISO/IEC 25010:2023 определяет номенклатуру характеристик качества программ, включая критически важные для эволюционного развития архитектур, однако не предписывает способов их вычисления. Количественные методы направлены на анализ качества программного кода и не распространяются на всю архитектуру системы. Теоретико-информационный подход к анализу сложных систем предоставляет инструментарий для количественной оценки сложности графовых структур на основе энтропии Шеннона, а концепция расстояния редактирования графов может обеспечить набор правил модификации архитектур в

условиях их цифровой трансформации. Вместе с тем перенос данных подходов с уровня кода на уровень архитектуры системы в существующей литературе не осуществлён.

Таким образом, в области управления архитектурой для построения автономно функционирующих горнопромышленных систем существует разрыв между потребностью в формальной количественной оценке качества архитектуры на всех этапах её жизненного цикла и отсутствием соответствующего инструментария. В соответствии с этим в рамках настоящего исследования ключевой проблематикой ставятся вопросы оценки и анализа архитектур сложных крупномасштабных пространственно-распределенных промышленных систем с открытым типом геотехнологических сред в условиях их цифровой трансформации. Фокус исследования направлен на определение способов построения такой архитектуры и таких механизмов анализа её работы, которые обеспечивали бы возможность построения автономно функционирующих киберфизических промышленных систем.

Имея в виду, что целью цифровой трансформации горнопромышленных предприятий в настоящем исследовании определено исключение человека из участия в реализации производственных процессов, очевидно, что в конечном счете в качестве акторов их выполнения следует подразумевать только программные сущности. Однако до стадии завершения цифровой трансформации предприятия, актором может выступать не только программа, но и человек, функции которого не были или не могут быть приведены к программному виду исполнения. В таком случае при попытке описания (составления модели) архитектуры предприятия могут возникнуть проблемы, связанные со сложностью сопоставления функций программного обеспечения и задач, решаемых персоналом. Так человек может выполнять ряд сложно организованных задач, манипулируя информацией из непересекающихся контуров управления, в то время как программное обеспечение, чаще всего, выполняет хорошо структурированные задачи, которые четко выделены на отдельные последовательные операции в узком контуре решений. Дифференцируемость же «природы» акторов управления может приводить к отсутствию прямого наследования задач между заменяемыми (и модернизируемыми) в проектных планах моделей архитектуры «as is» (начальной стадии ЦТ) и «to be» (стадии завершения ЦТ) и приводить в последствии к ошибкам – потере или дублированию отдельных задач и информации, а, следовательно, и к нарушению функциональной пригодности предприятия как системы. Под функциональной пригодностью понимается способность системы выполнять полный перечень производственных процессов в любой произвольный момент времени. При этом базовые компоненты структуры, обеспечивающие выполнение такого перечня производственных процессов, определяются в виде агентов управления.

Таким образом в контексте настоящего исследования предлагается найти способ отождествления архитектуры предприятия Индустрии 4.0 с понятием архитектуры программного обеспечения, поэтому в первую очередь формулируется следующее определение:

Агент управления архитектуры предприятия – это сущность, выступающая в роли актора при решении задач управления технологическими процессами и обладающая возможностями восприятия, обработки и передачи информации [26, 126-135].

В этой связи стоит отметить, что подход, начинающийся с описания перечня непрерывно изменяемых агентов управления в архитектуре, является не рациональным и, следовательно в первую очередь необходимо отталкиваться от статических (неизменяемых во времени) абстрактных представлений о функциональной деятельности системы – выполняемых задачах.

Строго говоря, задачей управления в производственных процессах всегда является проведение операций над данными, под которыми можно понимать когнитивные (логико-эвристические) и формальные математические вычисления. Так, независимо от типа решаемой задачи – управления исполнительными механизмами оборудования, оценки показателей производственных процессов, их оптимизации и т.п. – процедура её реализации представляет собой последовательную работу с информацией: цифровыми или аналоговыми (в т.ч. нейрофизиологическими) объектами данных [10, 136-139].

К типовым задачам можно отнести классические проблемы АСУ [10]: детектирование, идентификацию, прогнозирование, планирование, управление (и моделирование), обладающие принципиально разными назначениями и, соответственно, моделями вычислений, а так же, как правило, использующие отличные друг от друга структуры входящей и исходящей информации (Таблица 2).

Таблица 2. Классы задач управления предприятием, реализуемые агентами

Класс задачи	Назначение	Входящая информация	Исходящая информация
1 Детектирование	Определение текущего состояния объекта	Данные о текущих параметрах объекта	Данные о текущем состоянии объекта
2 Идентификация	Определение соответствия текущего состояния объекта требуемому	Данные о текущем состоянии объекта Данные о требуемом состоянии объекта	Данные о соответствии текущего состояния объекта с требуемым

Класс задачи	Назначение	Входящая информация	Исходящая информация
3 Прогнозирование	Определение будущего (возможного) состояния объекта	Данные о текущем состоянии объекта Данные об исторических состояниях объекта	Данные о будущем (возможном) состоянии объекта
4 Планирование	Определение требуемого состояния объекта	Данные о текущем состоянии объекта Данные о будущем (возможном) состоянии объекта	Данные о требуемом состоянии объекта
5 Управление	Определение перехода состояния объекта из текущего к требуемому	Данные о соответствии текущего состояния объекта с требуемым Данные о будущем (возможном) состоянии объекта Данные о требуемом состоянии объекта	Данные об изменении параметров текущего состояния для перехода к требуемому
6 Моделирование	Определение всех возможных состояний	Данные об исторических состояниях объекта	Данные о параметрах всех возможных состояний

При этом следует учитывать, что модели решения таких задач, как правило, представляют из себя не одну простую когнитивно-математическую операцию, а некоторую последовательность операций, продуцирующих промежуточные данные. Такие промежуточные данные не учитываются бизнес-логикой предприятия, в следствие отсутствия семантической значимости для описания состояний объектов производства и отсутствия необходимости долгосрочного хранения для возможности восстановления последовательности действий по изменению их состояний.

В соответствии с изложенным введем следующее определение:

Атомарная функциональная задача управления – это изолированная и неделимая процедура по обработке информации, описывающей какое-либо состояние (физического) объекта предприятия в произвольный момент времени, для изменения такого состояния в соответствии с заданной моделью поведения-вычислений.

Атомарность функциональной задачи управления подразумевает логическую связанность и неразделимость получаемых данных для описания конкретного состояния

какого-либо объекта предприятия. В такой ситуации **атомарной функциональной задачей управления будем называть высокоуровневую статическую абстракцию решаемой задачи бизнес-процессов в модели архитектуры предприятия**, в то время как **агент управления – это сущность, реализующая атомарную функциональную задачу**, форма исполнения которого может быть изменена, а в случаях цифровизации человеко-машинных процессов приведена к программному виду. На основании выявляемых атомарных функциональных задач управления при составлении модели архитектуры должен формироваться их неизменный во времени перечень, ассоциируемый в дальнейшем с конкретными агентами управления и их экземплярами. В качестве способа формального представления каталога атомарных функциональных задач управления и их ассоциации с конкретными агентами можно использовать составление соответствующих каталогов и матриц (по аналогии со стандартом TOGAF 10).

В то же время в контексте построения автономно функционирующих индустриальных систем формальное представление структуры, как самих агентов управления, так и их взаимосвязи, должны быть рассмотрены с позиции построения эффективных форм программной конфигурации. Процессы имплементации нового программного обеспечения в действующие архитектуры предприятий имеют ряд ограничений, связанных с проблемами исходной организации структур эксплуатируемых систем. Большая часть ПО в составе ядра архитектур предприятий является проприетарным, т.е. не предусматривающим прямой взаимосвязи с внешними системами, что существенно усложняет процесс цифровой трансформации и формирует специфические условия управления при нестационарности перечня агентов.

Решение проблемы нестационарности архитектуры вообще предусматривается при исполнении требований к качеству сложных крупномасштабных программных систем, регламентируемых стандартом ISO 25010 [79, 140, 141]. В рамках настоящего исследования данные метрики качества рассматриваются со следующих позиций:

- Функциональная пригодность – соответствие функций агентов управления перечню атомарных функциональных задач управления в любой произвольный момент времени жизненного цикла;
- Модульность – логико-программное и/или физическое разделение структуры программных компонентов;
- Модифицируемость – изменяемость структуры программных компонентов при сохранении атомарных функциональных задач управления;

- Масштабируемость – увеличение числа однотипных экземпляров программных компонентов при сохранении перечня атомарных функциональных задач управления;
- Расширяемость – изменяемость перечня атомарных функциональных задач управления, реализуемых программными агентами.

В этой связи предлагается исходить из позиции, что в рамках цифровой трансформации **атомарные функциональные задачи управления формируют неизменяемый каталог модели архитектуры**, а способы внесения изменений в каталог и валидация таких изменений остаются за рамками цифровой трансформации и настоящего исследования. В то же время, **обязательным условием существования и выделения агента управления является ассоциирование его с конкретной атомарной функциональной задачей управления**, в качестве которых предлагается использовать классификацию традиционных задач теории АСУ. Также следует учесть экспертный подход при проектировании архитектуры [142], который мощно обобщить в виде следующего критерия оптимизации: **обеспечение полноты функционального покрытия при размещении агентов управления на физических узлах за минимальную технико-экономическую стоимость жизненного цикла архитектуры системы.**

При этом технико-экономическая стоимость жизненного цикла архитектуры может определяться:

- **временными** затратами;
- **финансовыми** затратами;
- **сложностью** архитектуры.

Временные и финансовые затраты являются прямой оценкой обеспечения жизненного цикла архитектуры, в то время как под сложностью следует понимать некоторую экспертную (эвристическую) оценку таких затрат, которая, например, может охарактеризовать структурно-функциональный масштаб системы и быть выражена в виде суммарного числа элементарных операций по требуемому взаимодействию со слоями, классами и функциями архитектуры для приведения её к требуемому виду. Эвристическая сложность же, в таком случае, может быть определена по количеству операций для приведения текущей (as is) архитектуры к требуемой (to be) в соответствии с моделью жизненного цикла (Рисунок 10).

Таким образом целью настоящего исследования является обеспечение процессов цифровой трансформации горнопромышленных предприятий посредством разработки формальных критериев, моделей и алгоритмов оценки и анализа архитектур систем.

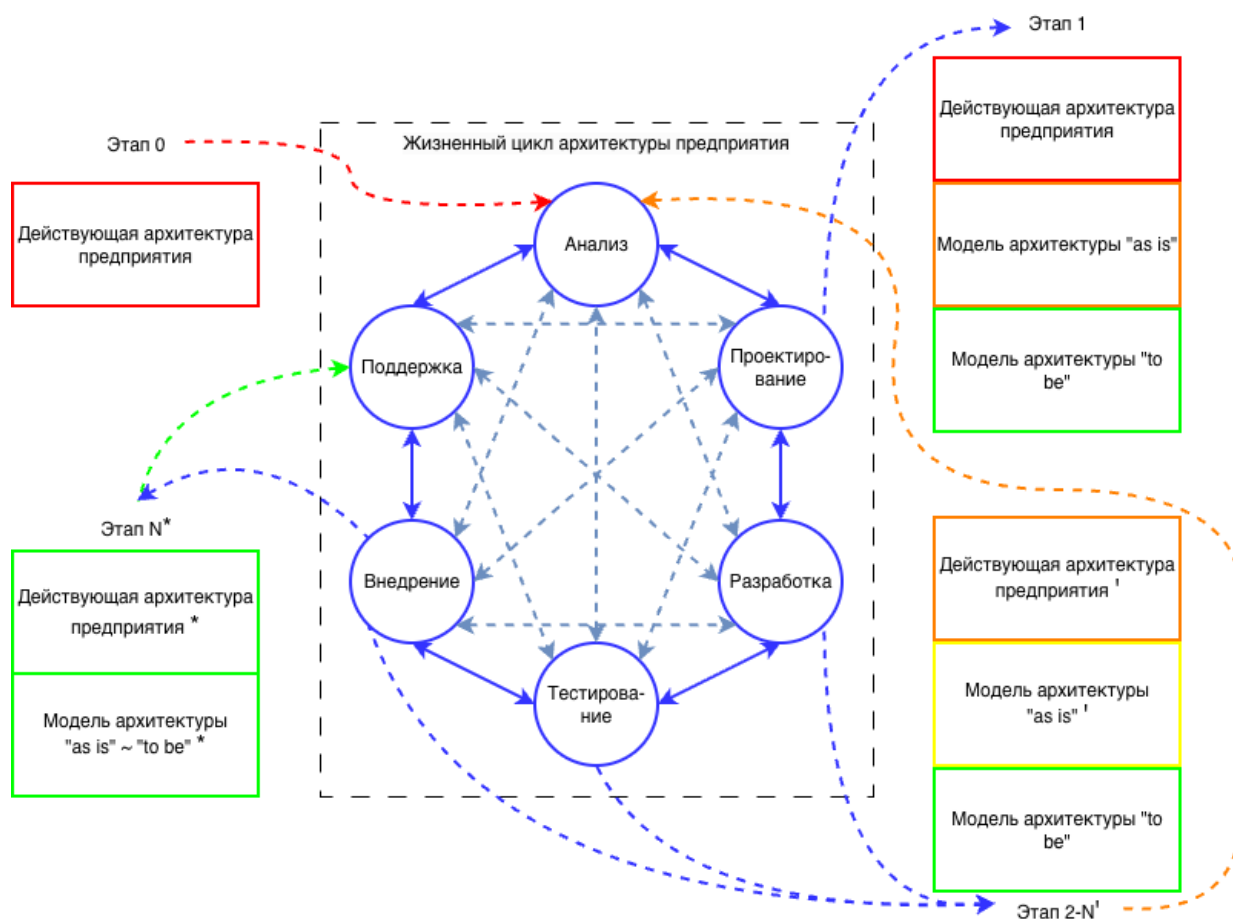


Рисунок 10. Концептуальная модель управления жизненным циклом архитектуры предприятия

Для достижения поставленной цели должны быть решены следующие задачи:

1. Разработать онтологическую модель архитектуры пространственно-распределенной геотехнической индустриальной системы с открытым типом производственной среды для обеспечения точности, полноты и непротиворечивости логико-семантического вывода ключевых компонентов и связей.
2. Формализовать количественные критерии и разработать инструментальные способы оценки и анализа архитектур индустриальных систем инвариантно этапам жизненного цикла их цифровой трансформации.
3. Разработать алгоритмические механизмы реализации предложенных моделей, критериев оценки и способов анализа архитектур для проведения экспериментального моделирования и апробации результатов.

1.5 Выводы по Главе 1

В первой главе проведён комплексный анализ современных архитектурных проблем горнопромышленных систем в условиях их цифровой трансформации. По результатам установлены основные ограничения существующих подходов к оценке и анализу архитектур и обоснована необходимость разработки нового методологического инструментария.

В разделе 1.1 рассмотрена история автоматизации горнопромышленных предприятий от систем класса SCADA, обеспечивавших централизованный оперативный мониторинг и диспетчеризацию горнотранспортных комплексов, через системы управления производственными процессами класса MES к современным проектам роботизации и создания автономно функционирующих производств. На примере действующих структурно-функциональных схем организации процессов показано, что программно-техническая инфраструктура современных горнопромышленных предприятий характеризуется наличием множества разнотипных специализированных систем (АСУ ГТК, ГГИС, ERP) с проприетарными архитектурами, отсутствием унифицированных подходов к организации межсистемного взаимодействия и, как следствие, высокими барьерами на пути интеграции новых компонентов. Определено, что ключевой проблемой цифровой трансформации является необходимость формирования единой модели архитектуры, объединяющей бизнес-процессы, физические объекты технологической среды, информацию и программные компоненты для создания автономно функционирующей кибер-физической системы.

В разделе 1.2 проведён анализ существующих моделей представления и задач управления архитектурами систем. Рассмотрены ведущие международные стандарты TOGAF 10 и RAMI 4.0, показана их направленность на многослойное онтологическое представление предприятия, охватывающее бизнес-архитектуру, архитектуру данных, приложений и технологий. Вместе с тем установлено, что предлагаемые стандартами модели ориентированы преимущественно на экспертный анализ и мало пригодны для формальных логических вычислений, обеспечивающих точность, полноту и непротиворечивость закладываемых знаний. Выполнена классификация и сравнительный анализ типовых архитектур программных систем с позиции распределения слоёв, классов и функций. Введены формальные определения понятий архитектуры, слоя, класса и функции программного обеспечения. Выявлено отсутствие формальных способов унифицированного описания и количественного сопоставления различных типов архитектур. Проведено аналитическое сравнение задач жизненного цикла архитектур

предприятий и программных систем, показавшее их принципиальную общность при различии в уровнях абстракции, что подчеркивает актуальность и обоснованность поиска единого подхода к оценке и сопоставлению.

В разделе 1.3 проведён обзор существующих методов оценки и анализа архитектур. В части для методов оценки архитектур предприятий (подраздел 1.3.1) установлено, что доминирующим подходом является экспертный анализ на основе сопоставления моделей «as is» и «to be» с использованием каталогов, матриц и диаграмм. Показано, что такой подход влечёт риски неполноты, противоречивости и неоднозначности семантических конструкций, а критерий соответствия моделей не является строго формализованным. Обоснована возможность применения онтологического моделирования на языке OWL с использованием механизмов дескриптивной логики и рассуждателей для формальной верификации полноты, непротиворечивости и консистентности моделей архитектуры предприятия. В части оценки архитектур программных систем (подраздел 1.3.2) проанализированы сценарные методы экспертной оценки (SAAM, ATAM, ARID), стандарт ISO/IEC 25010:2023, определяющий номенклатуру характеристик качества, а также количественные метрики уровня программного кода – связность и сцепление, метрики Чидамбера–Кемерера, цикломатическая сложность Маккейба и метрики Халстеда. Установлено, что сценарные методы дают качественную, но не количественную оценку; стандарт ISO/IEC 25010 определяет, что следует измерять, но не предлагает способов измерения; а количественные метрики действуют исключительно на уровне программного кода и не распространяются на всю архитектуру предприятия. В части анализа сложных систем (подраздел 1.3.3) рассмотрен теоретико-информационный подход на основе энтропии Шеннона применительно к оценке сложности графовых структур, а также концепция расстояния редактирования графов как принцип модификации графовых структур. Отмечено, что совмещение энтропийной оценки сложности конфигурации с оценкой стоимости операций редактирования создаёт основу для формирования комплексного критерия управления жизненным циклом архитектуры, однако перенос данных подходов с уровня кода на уровень архитектуры системы в существующей литературе не осуществлён.

На основании проведённого анализа в разделе 1.4 сформулирована постановка задачи исследования. Определён разрыв между потребностью в формальной количественной оценке качества архитектуры на всех этапах жизненного цикла и отсутствием соответствующего инструментария. Введены понятия агента управления и атомарной функциональной задачи как базовых компонентов модели архитектуры. Конкретизированы требования к качеству архитектуры в терминах стандарта ISO/IEC

25010 – функциональная пригодность, модульность, модифицируемость, масштабируемость и расширяемость. Сформулирован экспертный критерий оптимизации: обеспечение полноты функционального покрытия при размещении агентов управления на физических узлах за минимальную стоимость жизненного цикла архитектуры. Для достижения цели исследования определены три задачи: разработка онтологической модели архитектуры горнопромышленной системы для формального представления и верификации знаний при логико-семантическом выводе; формализация количественных критериев оценки и инструментальных способов анализа качества архитектуры, инвариантных к типу системы и этапам её жизненного цикла; разработка алгоритмических механизмов анализа архитектур с учётом стоимости жизненного цикла её цифровой трансформации.

ГЛАВА 2. РАЗРАБОТКА МОДЕЛЕЙ ПРЕДСТАВЛЕНИЯ АРХИТЕКТУР ИНДУСТРИАЛЬНЫХ СИСТЕМ

2.1 Формализация критериев и задач цифровой трансформации промышленных систем

В силу специфичности и отсутствия терминологического и семантического консенсуса в рассматриваемой области, в первую очередь охарактеризуем используемые в настоящем исследовании понятия и дадим ряд формальных определений:

Под *архитектурой промышленной системы* – A будем понимать совокупность компонент, их структурной конфигурации и множества типов связей между ними, определяющей фактическую форму организации и принципы управления производственными процессами предприятия. Инвариантно сфере функционирования промышленной системы в качестве таких компонентов выступают понятия принципиально разной природы, которые могут быть представлены в виде формальной системы:

$$A = \langle C^1, C^2, C^3, C^4 \rangle, \quad (5)$$

где:

C^1 – *технологические процессы и операции* – набор целенаправленных взаимодействий объектов физического мира, приводящих к изменению состояний таких объектов;

C^2 – *физические объекты* – объекты технологической среды, взаимодействующие между собой при выполнении производственных задач и которые в контексте основного экспериментального объекта, рассматриваемого в данной диссертации – горнопромышленного предприятия, ведущего добычу полезных ископаемых открытым способом – можно классифицировать в виде устройств (самосвалы, экскаваторы и т.д.), инфраструктуры (геопространственные объекты и технологические зоны), продукции (у.е. горной породы) и ресурсов (топливно-энергетических и др.);

C^3 – *агенты управления* – компоненты структуры управления предприятием (люди или программные системы), способные воспринимать, обрабатывать и передавать информацию о состояниях физических объектов для определения порядка выполнения технологических процессов и операций;

C^4 – *информация* – возможные состояния физических объектов, описываемые набором измеримых параметров, и данные, формирующие такие состояния.

Задачу управления (трансформации) архитектуры предприятия (предприятие или индустриальная система) можно сформулировать как выбор такой архитектуры, при которой оптимизируется интегральный показатель эффективности, в качестве которого может быть выбран, как более общий, объем получаемой прибыли или, что более релевантно предприятию по добыче полезных ископаемых, объем отгружаемого полезного ископаемого:

$$F: A \rightarrow A'; \Pi_{\Sigma}(A') \rightarrow opt, \quad (6)$$

где F – есть некоторая процедура трансформации архитектуры и выбора такой её конфигурации, которая обеспечивала бы оптимум Π_{Σ} – интегрального критерия эффективности предприятия.

Очевидно, что для решения подобной гиперсложной оптимизационной задачи необходима более строгая формальная постановка, требующая в свою очередь учёта большей детализации факторов, выявления их взаимосвязи, чётко заданного критерия и условий поиска. Однако попытка привести данную проблему к виду единой математической функции, то есть полностью формализовать, представляется чрезвычайно сложной (и, вероятно, с неразрешимой вычислительной сложностью), а следовательно, необходимостью поиска иных способов её представления.

Соответственно, можно сформулировать гипотезу, что существует такая метамодель Θ , которая позволила бы описать архитектуру предприятия A , определить вид процедуры управления трансформацией архитектуры F и свести задачу управления архитектурой к задаче оптимизации критерия Π_{Σ} .

В этой связи для большей ясности изложения введём ряд определений и допущений:

1. В любой произвольный момент времени предприятие сохраняет основной вид деятельности, а следовательно перечень выполняемых технологических процессов и операций C^1 , а также перечень физических объектов C^2 , должны быть сопоставимы и эквивалентны.
2. В любой произвольный момент времени архитектура предприятия должна обеспечивать выполнение полного перечня технологических процессов и операций, объясняющих и регламентирующих деятельность такого предприятия.
3. Цифровая трансформация – это непрерывный процесс недетерминированных во времени изменений архитектуры предприятия, направленных на улучшение интегральной эффективности.

4. Интегральная эффективность предприятия зависит от эффективности решения отдельных задач управления технологическими процессами и операциями $\Phi = \{\varphi_1, \dots, \varphi_n\}$;
5. Решение задач управления реализуется агентами управления $C_i^{3,j}; i = \overline{1, I}; j \in \{0; 1\}$, в качестве которых выступает персонал (должность) $C_i^{3,j=1}$ и программные системы $C_i^{3,j=0}$, у которых, в свою очередь существуют конкретные реализации в архитектуре – экземпляры: $c_{i,k}^{3,1}; k = \overline{1, K}$ и $c_{i,k}^{3,0}$.
6. Эффективность решения задач управления $Eff_{C_i^{3,j}}$ может быть определена как минимизация расхождений плановых значений производственных показателей с фактическими (точность $Acc_{C_i^{3,j}}$) при минимальной затрате времени и/или ресурсов (ресурсоемкость $Rsr_{C_i^{3,j}}$). При этом для каждого i -ого агента управления показатели могут быть определены только в виде интегрально вычисленных значений по соответствующим k -ым экземплярам.
7. Улучшение показателей эффективности решения задач управления может быть достигнуто путем разработки и внедрения программных компонентов с более совершенными (относительно точности и ресурсоемкости) вычислительными механизмами $C_i'^{3,j=0}$.
8. Внедряемые программные компоненты, как агенты управления, должны быть размещены на каких-либо вычислительных устройствах – физических узлах $Y = \{v_1, \dots, v_m\}$. При этом схема размещения программных компонентов на физических узлах определяет конфигурацию программной архитектуры предприятия.
9. В качестве экземпляров типовых физических узлов Y размещения могут выступать как непосредственно бортовые системы технологического оборудования u_p^2 из числа физических объектов C^2 , так и централизованные или распределенные сервера u^1 предприятия: $U = \{u^1; u_1^2, \dots, u_p^2\}$.

В сущности, каждый из описываемых компонентов (атомарные функциональные задачи управления, агенты, экземпляры, физические узлы) может быть представлен в виде каталогов – списков, где каждому из элементов такого списка можно поставить в соответствии элемент другого, а сами каталоги и отношения связей между ними будут формировать некоторую описательную метамодель архитектуры.

Тогда под цифровой трансформацией будем понимать процесс замены действующих агентов управления на более эффективные программные компоненты $C_i^{3,j=0}$. Непосредственным триггером модификации архитектуры является наличие (и возможность внедрения в состав архитектуры) такого программного компонента, который выполняет задачу управления на уровне эффективности не ниже, чем действующий агент управления. Следовательно, задачу управления архитектурой в ходе цифровой трансформации можно записать как:

$$\begin{aligned} \Pi_{\Sigma}(A') &= \sum_{i=1}^I j_i \rightarrow \min \\ F(A): \{C_i^{3,j}\} &\rightarrow \{C_i^{3,j=0}\} \\ Eff_{C_i^{3,j}} &= \frac{Acc_{C_i^{3,j=0}}}{Rsr_{C_i^{3,j=0}}} \geq Eff_{C_i^{3,j}} = \frac{Rsr_{C_i^{3,j=0}} = \sum_{i=1}^I Acc_{C_{i,k}^{3,j}}}{Perf_{C_i^{3,j=0}} \sum_{i=1}^I Perf_{C_{i,k}^{3,j}}} \\ i &= \overline{1, I}, j \in \{0; 1\} \end{aligned} \quad (7)$$

Отметим, что появление нового программного компонента $C_i^{3,j}$ обеспечивается вне процедур управления архитектурой предприятия – путём его создания и размещения в репозитории (витрине приложений) разработчиками программных систем. Возможность же внедрения нового программного компонента в качестве агента управления должна определяться сохранением функциональной пригодности всей производственной системы – способности системы реализовывать полный перечень задач управления Φ .

$$\{\Phi \multimap C_i^{3,j}\}; \{C_i^{3,j} \multimap C_{i,k}^{3,j}\}, \quad (8)$$

В то же время условия функциональной пригодности должны подразумевать обязательную реализацию отдельных атомарных функциональных задач управления на конкретных типовых физических объектах (например, оперативное управление движением техники на конкретных экземплярах такой техники), а следовательно каждый агент управления должен быть реализован на каком-либо типовом физическом узле и, соответственно, каждый экземпляр такого агента на каком-либо экземпляре физического узла:

$$\{\Phi \multimap Y\}; \{C_i^{3,j} \multimap Y\}; \{Y \multimap U\}; \{C_{i,k}^{3,j} \multimap U\}, \quad (9)$$

Очевидно, что для определения соответствия архитектуры условиям (8)-(9) в целях выполнения процедуры (7) необходимо оперировать модельным представлением. Если каждый из компонентов $\Phi, Y, U, C_i^{3,j}, c_{i,k}^{3,j}$ можно представить в виде каталога-списка, а для каждой пары таких каталогов можно определить бинарные отношения " $\rightarrow$ " (множественное отображение одного элемента каталога к множеству элементов другого) в виде соответствующих им связей, тогда каждая пара каталогов может быть представлена в виде двудольных графов:

1. Граф связи атомарных функциональных задач управления и агентов управления $g_1(\Phi, C_i^{3,j})$.
2. Граф связи атомарных функциональных задач управления и физических узлов размещения $g_2(\Phi, Y)$.
3. Граф связи агентов управления и физических узлов размещения $g_3(C_i^{3,j}, Y)$.
4. Граф связи агентов управления и их экземпляров $g_4(C_i^{3,j}, c_{i,k}^{3,j})$.
5. Граф связи физических узлов размещения и их экземпляров $g_5(Y, U)$.
6. Граф связи экземпляров агентов и экземпляров физических узлов $g_6(c_{i,k}^{3,j}, U)$.

Для каждого такого графа $\langle g_1, \dots, g_6 \rangle$ зафиксируем условия их формирования, правила оценки и изменений:

1. Каталоги атомарных функциональных задач управления Φ и физических узлов размещения Y являются инвариантами, как и граф g_2 . Их получение должно осуществляться на основе объективных априорных знаний и формальных правил, строго задающих каркас архитектуры конкретной индустриальной системы. Изменение каталогов Φ, Y и графа g_2 в рамках ЦТ архитектуры будем считать условно недопустимым.
2. Каталоги агентов управления $C_i^{3,j}$, их экземпляров $c_{i,k}^{3,j}$, экземпляров физических узлов U , а также графы g_4 и g_6 формируются на основе фактической номенклатуры с использованием механизмов экстракции знаний (или экспертно-аналитическим путём на первоначальном этапе ЦТ) из распределенного хранилища информации предприятия с последующей поддержкой ассоциативной связи по идентификаторам отдельных элементов. Изменение данных каталогов и графов допустимо и регламентируется процедурами управления архитектурой – добавлением, удалением компонентов и/или перераспределением связей в целях оптимизации.

3. Построение графов g_1, g_3, g_5 на первоначальном этапе ЦТ требует экспертно-аналитического сопоставления соответствующих каталогов, однако в последующем также должно поддерживаться ассоциативной связью по идентификаторам. Изменение таких графов также допустимо и регламентировано процедурами управления архитектурой.

При формировании графов $\langle g_1, \dots, g_6 \rangle$ могут образовываться типовые формы реляционных связей: один к одному (1:1), один ко многим (1: ∞), многие к одному (∞ :1), многие ко многим (∞ : ∞). Следует учитывать, что в настоящий момент в действующих архитектурах промышленных систем может быть явное или неявное дублирование реализации уникальных атомарных функциональных задач управления разными агентами (например, одновременная реализация оперативного управления движением самосвала водителем и бортовым модулем автономного управления роботизированной техникой АСУ ГТК). При этом будем считать, что дублирование реализации атомарных функциональных задач управления в автономно функционирующей кибер-физической промышленной системе является иррациональным и недопустимым ввиду возможного повышения неопределенности в процессах управления (неочевидности выбора релевантного источника информации для принятия решений) и противоречия критерию оптимальности (7) – выбору наиболее эффективных агентов управления.

Так в Таблице 3 представлена типовая матрица отношений связи в графах каталогов архитектуры, предусматривающая допустимость наличия дублирования реализаций атомарных функциональных задач управления при описании состояния «as is».

Таблица 3

Матрица отношений связи в графах каталогов архитектуры (состояние «as is»)

Каталоги	Φ	Υ	$C_i^{3,j}$	$c_{i,k}^{3,j}$	U
Φ	-	$\infty;\infty$	$\infty;\infty$	-	-
Υ	$\infty;\infty$	-	$\infty;\infty$	-	1: ∞
$C_i^{3,j}$	$\infty;\infty$	$\infty;\infty$	-	1: ∞	-
$c_{i,k}^{3,j}$	-	-	$\infty;1$	-	$\infty;\infty$
U	-	$\infty;1$	-	$\infty;\infty$	-

Тогда можно заключить, что одной из первичных задач (условий) управления архитектурой промышленной системы в рамках ЦТ является нормализация отношений в графе $g_1(\Phi, C_i^{3,j})$, приводящая тип связи многие ко многим ($\infty;\infty$) к виду многие к одному

($\infty:1$). Таким образом отношения связи в графах архитектуры в состоянии «to be» должны иметь вид по Таблице 4.

Таблица 4

Матрица отношений связи в нормализованных графах каталогов архитектуры (состояние «to be»)

Каталоги	Φ	Υ	C_i^{3j}	$c_{i,k}^{3j}$	U
Φ	-	$\infty:\infty$	$\infty:1$	-	-
Υ	$\infty:\infty$	-	$\infty:\infty$	-	$1:\infty$
C_i^{3j}	$1:\infty$	$\infty:\infty$	-	$1:\infty$	-
$c_{i,k}^{3j}$	-	-	$\infty:1$	-	$\infty:\infty$
U	-	$\infty:1$	-	$\infty:\infty$	-

Отметим, что приведение графа g_1 к совершенному парасочетанию в виде отношений один к одному (1:1) допустимо, но неочевидно с точки зрения эффективности.

Тогда задача ЦТ может быть записана как выбор такой конфигурации архитектуры в состоянии «to be», которая:

1. Нормализует связи в графе g_1 в части устранения дублирования реализации одной атомарной функциональной задачи управления разными агентами («остаться должен только один» - с лучшим показателем эффективности).
2. Заменяет агентов управления из числа персонала $C_i^{3,j=1}$ на программных агентов $C_i^{3,j=0}$ (все агенты должны стать программными: $C_i^{*3,j=0}$).
3. Не нарушает структуру графа g_2 и модифицирует структуры графов g_1, g_3, g_4, g_5, g_6 .

В таком случае необходимо формализовать критерий оптимальности для таких структур и архитектуры в целом. Для начала определим, что графы g_1, g_2, g_3, g_5 регламентируют связи между абстрактными сущностями (концептами) для описания логической конфигурации архитектуры (типовое назначение задач агентам, задач узлам, агентов узлам, узлов их экземплярам); а графы g_4, g_6 отражают техническую реализацию такой конфигурации (множественное копирование агентов и узлов с учётом фактических масштабов производства, технических возможностей и ограничений).

Набор графов $\langle g_1, \dots, g_6 \rangle$ определяет условия конфигурации всей архитектуры индустриальной системы A , которая может быть также представлена в виде некоторого графа $G(A)$ и процедуры конфигурации F_{cnfg} :

$$F_{cnfg}(g_1, \dots, g_6) = G(A), \quad (10)$$

Тогда запишем следующее уравнение, отражающее подход экспертно-аналитической оптимизации архитектуры:

$$F_{opt}(A) = G'(A'), \quad (11)$$

В таком случае можно:

1. Определить критерий и способ оценки качества архитектуры A через её представление в виде графа $G(A)$.
2. Определить правила формирования и модификации графа архитектуры, обеспечивающие возможности оценки и сопоставления состояний «as is» и «to be» в целях выбора оптимального решения.

Тогда в качестве критериев оценки и оптимальности можно рассматривать следующее:

1. Структурное качество архитектуры, под которой может пониматься сложность логической и/или технической конфигурации в статическом состоянии.
2. Качество (сложность) структурных изменений архитектуры, учитывающее потенциальную стоимость модификации логической и/или технической конфигурации из состояния «as is» в состояние «to be».
3. Качество (сложность) жизненного цикла архитектуры, учитывающее стоимость поддержки архитектуры при недетерминированных во времени изменениях в целях улучшения показателей эффективности.

В такой постановке, очевидно, может быть использован комбинированный аппарат оценки сложности графовых структур на базе энтропийных оценок и процедур редактирования графов (3)-(4). Однако применение такого комбинированного математического аппарата требует формального определения структуры графа $G(A)$.

2.2 Формализация задачи онтологического моделирования архитектуры индустриальной системы

Как было отмечено в разделе 2.1. каталоги Φ и Υ должны обеспечивать статическую часть модели, формализующую неизменяемые устойчивые знания о системе и регламентирующие принципы организации архитектуры инвариантно этапам жизненного цикла. В то же время каталоги $U, C_i^{3,j}, c_{i,k}^{3,j}$ формировать динамическую часть модели, определяющей фактическую конфигурацию архитектуры в целях её структурной и численной оптимизации в ходе цифровой трансформации.

Соответственно существует некоторая содержательная часть архитектуры, которая является инвариантной на любом произвольном промежутке времени, независимо от её изменений, вызванных цифровой трансформацией, в то время как часть архитектуры, преимущественно связанная с агентами управления, может и должна быть модифицирована. Однако получение полного перечня атомарных функциональных задач управления осложнено рядом проблем:

1. Степень глубины детализации: задачи должны быть в достаточной степени «атомизированы», но не представлять из себя избыточную декомпозицию до элементарных операций.
2. Степень полноты: задачи должны описывать всю производственную деятельность предприятия, в том числе такую, которая в настоящий момент не является программной функцией (реализуется в виде «умственного» труда персонала), но является неотъемлемой операцией цепочки технологических процессов.
3. Степень точности: задача должна объяснять взаимосвязь разнотипных компонентов архитектуры и быть эквивалентна формальной вычислительной процедуре.
4. Степень непротиворечивости: задача должна быть однозначной по своему описанию и содержанию, а также не быть эквивалентной другой задаче.

Исходя из сформулированных проблем можно заключить, что их решение и получение каких-либо количественных оценок возможно с использованием такой модели, которая обеспечивала бы представление гетерогенных компонентов архитектуры в виде абстрактных понятий, связывала бы их формальными логическими правилами и предоставляла вычислительный аппарат, гарантирующий возможность работы с высказываниями.

мере охарактеризовать как конкретный производственный объект, так и предоставить удобную универсальную инструментальную форму обобщения. Как правило, на действующих предприятиях элементы модели архитектуры могут быть представлены в виде текстовых документов, сопровождаемых схемами в различных нотациях графического моделирования. В наиболее общем виде, с точки зрения структурно-функциональных описаний, ландшафтная архитектура представлена на Рисунке 11, которая отражает ключевые физические сущности, процессы и управляющие компоненты в парадигме иерархии АСУ на примере предприятий по открытой добыче полезных ископаемых. Однако, использование подобной схемы для реализации цифровой трансформации трудноосуществимо ввиду низкой информативности и невозможности проведения каких-либо вычислительных процедур.

Таким образом, как было отмечено ранее, в качестве вида и механизмов построения модели архитектуры предлагается рассматривать онтологическое моделирование, для которого введём следующее:

Пусть A^0 – это исходный вид архитектуры предприятия на начальном этапе цифровой трансформации t_0 , который может быть описан метамоделью «as is» – Θ^0 ; $A', \dots, A'' \dots'$ – виды архитектуры предприятия на отдельных этапах её изменений, которые могут быть описаны метамоделями $\Theta', \dots, \Theta'' \dots'$; а A^* – вид архитектуры предприятия по завершению цифровой трансформации t_* , который может быть описан метамоделью «to be» – Θ^* . В каждый произвольный момент времени t предприятие обладает некоторой архитектурой A^t , вид которой определяется через представление метамодели $\Theta^t = \langle g_1, \dots, g_6 \rangle$. Непрерывный процесс цифровой трансформации предприятия может быть представлен как совокупность дискретных недетерминированных во времени шагов $\xrightarrow{DT}$ по изменению архитектуры A^t , осуществляемых до того момента пока её вид не станет эквивалентен A^* , что может быть определено через сопоставление метамodelей Θ^t и Θ^* .

Соответственно на начальном шаге t_0 должен быть определен исходный вид архитектуры A^0 , составлена метамодель Θ^0 , определен конечный вид архитектуры A^* и составлена оптимальная метамодель Θ^* . Далее начиная с шага t_1 и до t_* необходимо выполнять оценку соответствия текущей метамодели архитектуры Θ^t требуемой Θ^* , определять необходимость и возможность модификации архитектуры, а также осуществлять принятие решений по внесению соответствующих изменений. Концептуальная схема структуры управления архитектурой индустриальной системы в условиях её цифровой трансформации представлена на Рисунке 12.

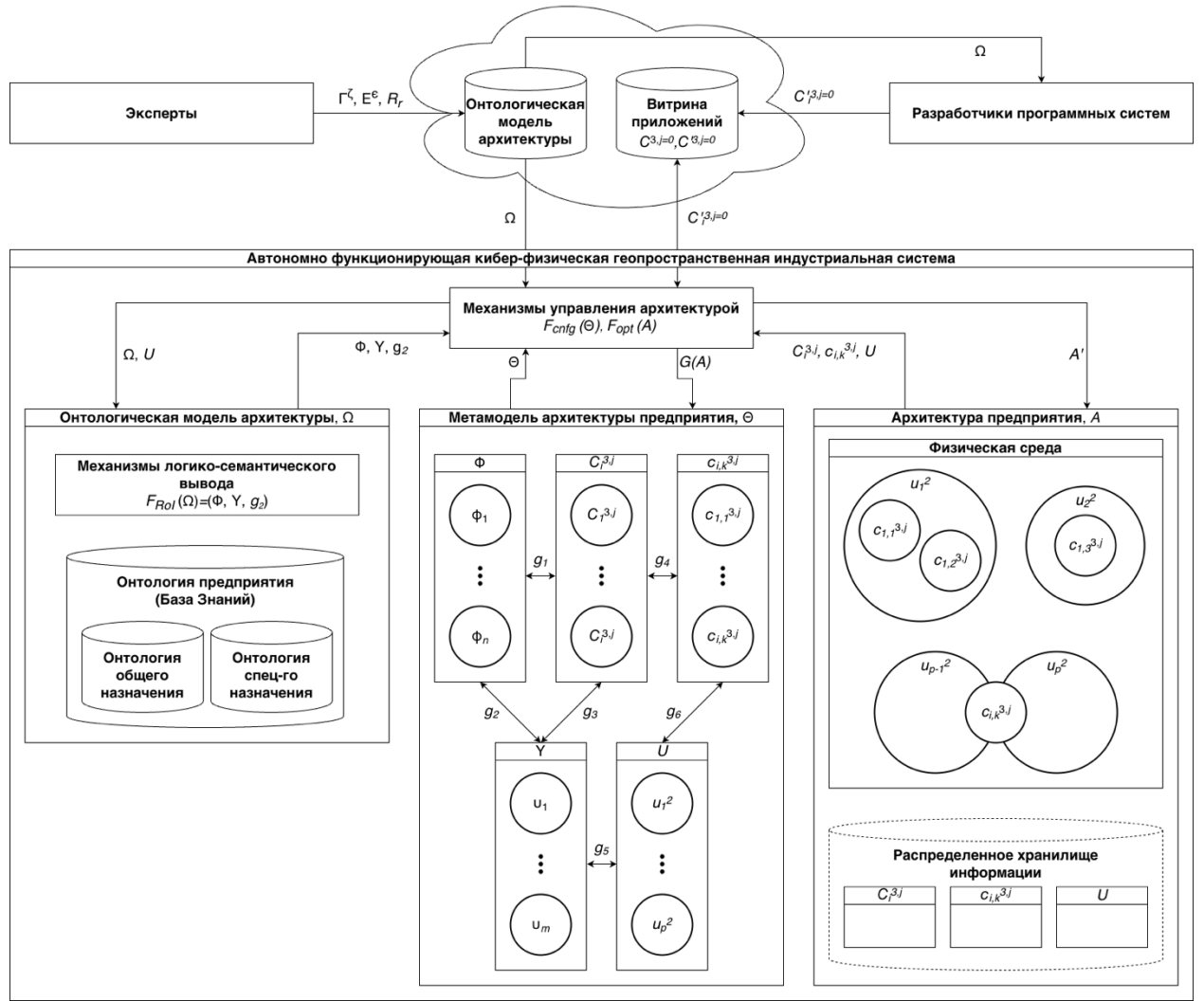


Рисунок 12. Концептуальная схема структуры управления автономно функционирующей кибер-физической промышленной системой

Тогда процесс управления архитектурой предприятия в условиях цифровой трансформации, состоящей из множества дискретных шагов DT , может быть записан как:

$$F: A^0 \xrightarrow{DT} A^1 \xrightarrow{DT} \dots \xrightarrow{DT} A^*, \quad (12)$$

где $\xrightarrow{DT}$ соответствует шагу изменения архитектуры (7) для приведения предприятия к виду автономно функционирующей промышленной системы.

При этом на каждом из этапов должно выполняться условие:

$$\begin{cases} \Theta^{t-1} \Leftrightarrow \Theta^t \Leftrightarrow \Theta^{t+1} \\ (\Phi^t \multimap \Upsilon^t) \in \Theta^t \\ \Phi^{t-1} \Leftrightarrow \Phi^t \Leftrightarrow \Phi^{t+1}, \\ \Upsilon^{t-1} \Leftrightarrow \Upsilon^t \Leftrightarrow \Upsilon^{t+1} \end{cases}, \forall t \geq 1 \quad (13)$$

А также должно выполняться условие вида:

$$F_{Rol}(\Omega) = g_2(\Phi^t, \Upsilon^t), \quad (14)$$

где F_{Rol} – правила механизма логико-семантического вывода каталогов атомарных функциональных задач управления, типовых физических узлов размещения и их графа связи из онтологической модели архитектуры Ω .

В то же время достижимость (14), т.е. выводимость соответствия $\Phi^t \multimap \Upsilon^t$ из онтологической модели Ω , возможно лишь тогда, когда модель Ω – консистентна и когерентна.

Под консистентностью и когерентностью модели следует понимать истинность и непротиворечивость всех заложенных в неё аксиоматических высказываний в виде объявленных понятий и отношений между ними. Следовательно, первоначально необходимо составить набор концептуальных выражений, которые лягут в основу онтологической модели архитектуры пространственно-распределенной геотехнической индустриальной системы, связать их соответствующим набором отношений, определяющих деятельность предприятия (на примере горнопромышленного), а полученную онтологию проверить на консистентность и когерентность в целях обеспечения точности и непротиворечивости получаемой метамодели архитектуры. Полнота же метамодели не может быть определена полнотой онтологической модели в силу известных теорем (Теорема Гёделя о неполноте, теорема Open-World Assumption). В таком случае под полнотой модели будем подразумевать жесткий комбинаторный перебор всех возможных высказываний на базе строгого механизма логико-семантического вывода.

2.3 Разработка онтологической модели архитектуры горнопромышленной системы

Для составления онтологической модели архитектуры горнопромышленной системы будут использованы принципы и подходы современной дескрипционной логики и языка OWL, представляющие из себя наборы аксиоматических двуместных и трехместных предикатов, подчиняющихся формальным логическим исчислениям и объявляющих:

- Концепты: сущностей иерархической модели семантических классов;
- Индивиды: экземпляров классов и подклассов в иерархии концептов;
- Роли: отношения семантической связи между классами иерархии концептов.

Структура онтологической модели архитектуры горнопромышленной системы будет иметь следующий вид:

1. Онтология общего назначения, предопределяющая набор базовых инвариантных концептов кибер-физической пространственно-распределенной геотехнической промышленной системы с открытым типом производственной среды в виде иерархии классов и отношений между ними.
2. Онтология специального назначения, предопределяющая структуру унифицированного горнопромышленного предприятия на примере предприятия по открытой добыче полезных ископаемых в виде конечного набора индивидов иерархии классов.

2.3.1 Разработка онтологии общего назначения

Для объявления онтологии общего назначения введём первую базовую часть алфавита, представляющую из себя семантический триплет вида:

$$R_0(\Gamma^{\zeta}, \Gamma^{\zeta\zeta}) \in \Omega, \quad (15)$$

где Γ^{ζ} – абстрактное понятие (концепт), являющийся суперклассом (родительским концептом) для понятия $\Gamma^{\zeta\zeta}$, а R_0 – предикат иерархии классов типа «SubclassOf».

Тогда опишем иерархию концептов типовой архитектуры кибер-физической пространственно-распределенной геотехнической промышленной системы с открытым типом производственной среды.

Концепт «Предприятие» A включает в себя (имеет подкласс/субконцепт – R_0) концепты «Технологический процесс» Γ^1 , «Физический объект» Γ^2 , «Агент управления» Γ^3 , «Информация» Γ^4 (Таблица 5):

$$R_0(A, \Gamma^1 \sqcap \Gamma^2 \sqcap \Gamma^3 \sqcap \Gamma^4), \quad (16)$$

Таблица 5

Иерархия субконцептов концепта «Предприятие»

№ п/п	Класс/концепт, подкласс/субконцепт	Наименование
0	Γ^0	Предприятие
1	Γ^1	Технологический процесс
2	Γ^2	Физический объект
3	Γ^3	Агент управления
4	Γ^4	Информация

В Таблице 6 представлена иерархия субконцептов концепта «Технологический процесс»:

$$R_0(\Gamma^1, \Gamma^{1,1} \sqcap R_0(\Gamma^{1,2}, \Gamma^{1,2,1} \sqcap \Gamma^{1,2,2} \sqcap \Gamma^{1,2,3})), \quad (17)$$

Таблица 6

Иерархия субконцептов концепта «Технологический процесс»

№ п/п	Класс/концепт, подкласс/субконцепт	Наименование
1.1	$\Gamma^{1,1}$	Взаимодействие
1.2	$\Gamma^{1,2}$	Процесс
1.2.1	$\Gamma^{1,2,1}$	Основной процесс
1.2.2	$\Gamma^{1,2,2}$	Вспомогательный процесс
1.2.3	$\Gamma^{1,2,3}$	Обеспечивающий процесс

В Таблице 7 представлена иерархия субконцептов концепта «Физический объект»:

$$R_0(\Gamma^2, R_0(\Gamma^{2,1}, \Gamma^{2,1,1} \sqcap \Gamma^{2,1,2}) \sqcap R_0(\Gamma^{2,1}, \Gamma^{2,2,1} \sqcap \Gamma^{2,2,2}) \sqcap R_0(\Gamma^{2,3}, \Gamma^{2,3,1} \sqcap \Gamma^{2,3,2}) \sqcap R_0(\Gamma^{2,4}, \Gamma^{2,4,1} \sqcap \Gamma^{2,4,2} \sqcap \Gamma^{2,4,3}) \sqcap R_0(\Gamma^{2,5}, R_0(\Gamma^{2,5,1}, \Gamma^{2,5,1,1} \sqcap \Gamma^{2,5,1,2}) \sqcap R_0(\Gamma^{2,5,2}, \Gamma^{2,5,2,1} \sqcap \Gamma^{2,5,2,2} \sqcap \Gamma^{2,5,2,3} \sqcap \Gamma^{2,5,2,4}) \sqcap \Gamma^{2,5,3})), \quad (18)$$

Иерархия субконцептов концепта «Физический объект»

№ п/п	Класс/концепт, подкласс/субконцепт	Наименование
2.1	$\Gamma^{2,1}$	Оборудование
2.1.1	$\Gamma^{2,1,1}$	Мобильное оборудование
2.2.2	$\Gamma^{2,1,2}$	Стационарное оборудование
2.2	$\Gamma^{2,2}$	Инфраструктура
2.2.1	$\Gamma^{2,2,1}$	Геопространственная инфраструктура
2.2.2	$\Gamma^{2,2,2}$	Техническая инфраструктура
2.3	$\Gamma^{2,3}$	Ресурс
2.3.1	$\Gamma^{2,3,1}$	Энергетический ресурс
2.3.2	$\Gamma^{2,3,2}$	Инструментальный ресурс
2.4	$\Gamma^{2,4}$	Продукт
2.4.1	$\Gamma^{2,4,1}$	Природный продукт
2.4.2	$\Gamma^{2,4,2}$	Геотехнический продукт
2.4.3	$\Gamma^{2,4,3}$	Технический продукт
2.5	$\Gamma^{2,5}$	Устройство
2.5.1	$\Gamma^{2,5,1}$	Устройство пространственного взаимодействия
2.5.1.1	$\Gamma^{2,5,1,1}$	Динамическое устройство пространственного взаимодействия
2.5.1.2	$\Gamma^{2,5,1,2}$	Статическое устройство пространственного взаимодействия
2.5.2	$\Gamma^{2,5,2}$	Устройство технологического взаимодействия
2.5.2.1	$\Gamma^{2,5,2,1}$	Устройство технологического взаимодействия с природным продуктом
2.5.2.2	$\Gamma^{2,5,2,2}$	Устройство технологического взаимодействия с геотехническим продуктом
2.5.2.3	$\Gamma^{2,5,2,3}$	Устройство технологического взаимодействия с оборудованием
2.5.2.4	$\Gamma^{2,5,2,4}$	Устройство технологического взаимодействия с ресурсом
2.5.3	$\Gamma^{2,5,3}$	Устройство эксплуатационного взаимодействия

В Таблице 8 представлена иерархия субконцептов концепта «Агент управления»:

$$R_0(\Gamma^3, \Gamma^{3,1} \sqcap \Gamma^{3,2} \sqcap \Gamma^{3,3}), \quad (19)$$

Таблица 8

Иерархия субконцептов концепта «Агент управления»

№ п/п	Класс/концепт, подкласс/субконцепт	Наименование
3.1	$\Gamma^{3,1}$	Функция
3.2	$\Gamma^{3,2}$	Горизонт принятия решения
3.3	$\Gamma^{3,3}$	Количество объектов управления

В Таблице 9 представлена иерархия субконцептов концепта «Информация»:

$$R_0(\Gamma^4, R_0(\Gamma^{4,1}, \Gamma^{4,1,1} \sqcap \Gamma^{4,1,2} \sqcap \Gamma^{4,1,3} \sqcap \Gamma^{4,1,4}) \sqcap \Gamma^{4,2} \sqcap \Gamma^{4,3}), \quad (20)$$

Таблица 9

Иерархия субконцептов концепта «Информация»

№ п/п	Класс/концепт, подкласс/субконцепт	Наименование
4.1	$\Gamma^{4,1}$	Состояние
4.1.1	$\Gamma^{4,1,1}$	Пространственное состояние
4.1.2	$\Gamma^{4,1,2}$	Технологическое состояние
4.1.3	$\Gamma^{4,1,3}$	Эксплуатационное состояние
4.1.4	$\Gamma^{4,1,4}$	Экономическое состояние
4.2	$\Gamma^{4,2}$	Время
4.3	$\Gamma^{4,3}$	Роль

Тогда полная иерархия концептов онтологии общего назначения архитектуры кибер-физической пространственно-распределенной геотехнической индустриальной системы будет выглядеть как:

$$R_0(A, (R_0(\Gamma^1, \Gamma^{1,1} \sqcap R_0(\Gamma^{1,2}, \Gamma^{1,2,1} \sqcap \Gamma^{1,2,2} \sqcap \Gamma^{1,2,3}))) \sqcap (R_0(\Gamma^2, R_0(\Gamma^{2,1}, \Gamma^{2,1,1} \sqcap \Gamma^{2,1,2}) \sqcap R_0(\Gamma^{2,1}, \Gamma^{2,2,1} \sqcap \Gamma^{2,2,2}) \sqcap R_0(\Gamma^{2,3}, \Gamma^{2,3,1} \sqcap \Gamma^{2,3,2}) \sqcap R_0(\Gamma^{2,4}, \Gamma^{2,4,1} \sqcap \Gamma^{2,4,2} \sqcap \Gamma^{2,4,3}) \sqcap R_0(\Gamma^{2,5}, R_0(\Gamma^{2,5,1}, \Gamma^{2,5,1,1} \sqcap \Gamma^{2,5,1,2}) \sqcap R_0(\Gamma^{2,5,2}, \Gamma^{2,5,2,1} \sqcap \Gamma^{2,5,2,2} \sqcap \Gamma^{2,5,2,3} \sqcap \Gamma^{2,5,2,4}) \sqcap \Gamma^{2,5,3}) \sqcap (R_0(\Gamma^3, \Gamma^{3,1} \sqcap \Gamma^{3,2} \sqcap \Gamma^{3,3})) \sqcap (R_0(\Gamma^4, R_0(\Gamma^{4,1}, \Gamma^{4,1,1} \sqcap \Gamma^{4,1,2} \sqcap \Gamma^{4,1,3} \sqcap \Gamma^{4,1,4}) \sqcap \Gamma^{4,2} \sqcap \Gamma^{4,3}))) \in \Omega, \quad (21)$$

Далее введем ряд ролей – семантических конструкций в виде отношений в иерархии концептов, определяющих структурно-функциональную архитектуру кибер-физической пространственно-распределенной геотехнической индустриальной системы.

$$R_r(\forall \Gamma^z, \exists \Gamma^z) \in \Omega, \quad (22)$$

где R_r – множество ролевых отношений между концептами Γ^z , квантификатор «всеобщности» $\forall$ – обозначает указание распространения роли на все субконцепты и индивиды, а квантификатор «существования» $\exists$ – распространение на некоторые (не все, какие-либо) субконцепты и индивиды. При этом запись $R_r(\forall \Gamma^z, \exists \Gamma^{zz})$ означает, что для любого элемента $x^z \in \Gamma^z$ существует такой $x^{zz} \in \Gamma^{zz}$, что между ними выполнено отношение $R_r(x^z, x^{zz})$.

Так, каждый «Физический объект» выполняет какой-либо «Технологический процесс» – R_1 и, соответственно, каждый «Технологический процесс» выполняется каким-либо «Физическим объектом»:

$$R_1(\forall \Gamma^2, \exists \Gamma^1), \quad (23)$$

«Физический объект» может иметь роль «Субъект» или «Объект», которую он имеет в каком-либо «Взаимодействии»:

$$R_{1,1}(\exists \Gamma^2, \exists \Gamma^{1,1}), \quad (24)$$

$$R_{1,2}(\exists \Gamma^2, \exists \Gamma^{1,2}), \quad (25)$$

Таблица 10
Роли концептов «Физический объект» и «Роли»

№ п/п	Отношение/роль	Наименование	Область	Диапазон
1	R_1	Выполняет	Γ^2	Γ^1
1.1	$R_{1,1}$	Является субъектом взаимодействия	Γ^2	$\Gamma^{1,1}$
1.2	$R_{1,2}$	Является объектом взаимодействия	Γ^2	$\Gamma^{1,1}$

Соответственно при выполнении каких-либо «Технологических процессов» и «Взаимодействий» в какой-либо роли каждый «Физический объект» взаимодействует с каким-либо другим «Физическим объектом»:

$$R_3(\forall \Gamma^2, \exists \Gamma^2), \quad (26)$$

«Оборудование» производит «Продукт», а «Продукт» производится «Оборудованием»:

$$R_{3,1}(\forall \Gamma^{2,1}, \exists \Gamma^{2,4}), \quad (27)$$

«Оборудование» состоит из «Устройств», а «Устройства» являются частью «Оборудования». В том числе «Оборудование» может состоять из «Устройств пространственного, технологического и эксплуатационного взаимодействий»:

$$R_{3,2}(\forall \Gamma^{2,1}, \exists \Gamma^{2,5}), \quad (28)$$

$$R_{3,2,1}(\forall \Gamma^{2,5}, \exists \Gamma^{2,5,1}), \quad (29)$$

$$R_{3,2,2}(\forall \Gamma^{2,1}, \exists \Gamma^{2,5,2}), \quad (30)$$

$$R_{3,2,3}(\forall \Gamma^{2,1}, \exists \Gamma^{2,5,3}), \quad (31)$$

Каждая «Инфраструктура» размещает какое-либо «Оборудование», «Ресурс», «Продукт» или «Устройство»:

$$R_{3,3}(\forall \Gamma^{2,2}, \exists \Gamma^{2,1} \sqcap \exists \Gamma^{2,3} \sqcap \exists \Gamma^{2,4} \sqcap \exists \Gamma^{2,5}), \quad (32)$$

$$R_{3,3,1}(\exists \Gamma^{2,2}, \exists \Gamma^{2,1}), \quad (33)$$

$$R_{3,3,2}(\exists \Gamma^{2,2}, \exists \Gamma^{2,3}), \quad (34)$$

$$R_{3,3,3}(\exists \Gamma^{2,2}, \exists \Gamma^{2,4}), \quad (35)$$

$$R_{3,3,4}(\exists \Gamma^{2,2}, \exists \Gamma^{2,5}), \quad (36)$$

Каждое «Оборудование» или «Устройство» содержит «Оборудование» или «Ресурс»:

$$R_{3,4}(\forall \Gamma^{2,1} \sqcap \forall \Gamma^{2,5}, \exists \Gamma^{2,1} \sqcap \exists \Gamma^{2,3}), \quad (37)$$

$$R_{3,4,1}(\exists \Gamma^{2,1}, \exists \Gamma^{2,1}), \quad (38)$$

$$R_{3,4,2}(\exists \Gamma^{2,1}, \exists \Gamma^{2,3}), \quad (39)$$

$$R_{3,4,3}(\exists \Gamma^{2,5}, \exists \Gamma^{2,1}), \quad (40)$$

$$R_{3,4,4}(\exists \Gamma^{2,5}, \exists \Gamma^{2,3}), \quad (41)$$

Каждое «Устройство» потребляет какой-либо «Ресурс»:

$$R_{3,5}(\forall \Gamma^{2,5}, \exists \Gamma^{2,3}), \quad (42)$$

Каждое «Устройство технологического взаимодействия» обрабатывает какой-либо «Продукт»:

$$R_{3,6}(\forall \Gamma^{2,5,2}, \exists \Gamma^{2,4}), \quad (43)$$

Таблица 11

Роли субконцептов концепта «Физический объект»

№ п/п	Отношение/роль	Наименование	Область	Диапазон
3	R_3	Взаимодействует с	Γ^2	Γ^2
3.1	$R_{3,1}$	Производит	$\Gamma^{2,1}$	$\Gamma^{2,4}$
3.2	$R_{3,2}$	Состоит из	$\Gamma^{2,1}$	$\Gamma^{2,5}$
3.2.1	$R_{3,2,1}$	Состоит из устройства пространственного взаимодействия	$\Gamma^{2,1}$	$\Gamma^{2,5,1}$
3.2.2	$R_{3,2,2}$	Состоит из устройства технологического взаимодействия	$\Gamma^{2,1}$	$\Gamma^{2,5,2}$
3.2.3	$R_{3,2,3}$	Состоит из устройства эксплуатационного взаимодействия	$\Gamma^{2,1}$	$\Gamma^{2,5,3}$
3.3	$R_{3,3}$	Размещает	$\Gamma^{2,2}$	$\Gamma^{2,1} \sqcap \Gamma^{2,3} \sqcap \Gamma^{2,4} \sqcap \Gamma^{2,5}$
3.3.1	$R_{3,3,1}$	Размещает (оборудование)	$\Gamma^{2,2}$	$\Gamma^{2,1}$
3.3.2	$R_{3,3,2}$	Размещает (ресурс)	$\Gamma^{2,2}$	$\Gamma^{2,3}$
3.3.3	$R_{3,3,3}$	Размещает (продукт)	$\Gamma^{2,2}$	$\Gamma^{2,4}$
3.3.4	$R_{3,3,4}$	Размещает (устройство)	$\Gamma^{2,2}$	$\Gamma^{2,5}$
3.4	$R_{3,4}$	Содержит	$\Gamma^{2,1} \sqcap \Gamma^{2,5}$	$\Gamma^{2,1} \sqcap \Gamma^{2,3}$
3.4.1	$R_{3,4,1}$	(оборудование) Содержит (оборудование)	$\Gamma^{2,1}$	$\Gamma^{2,1}$
3.4.2	$R_{3,4,2}$	(оборудование) Содержит (ресурс)	$\Gamma^{2,1}$	$\Gamma^{2,3}$
3.4.3	$R_{3,4,3}$	(устройство) Содержит (оборудование)	$\Gamma^{2,5}$	$\Gamma^{2,1}$
3.4.4	$R_{3,4,4}$	(устройство) Содержит (ресурс)	$\Gamma^{2,5}$	$\Gamma^{2,3}$
3.5	$R_{3,5}$	Потребляет	$\Gamma^{2,5}$	$\Gamma^{2,3}$
3.6	$R_{3,6}$	Обрабатывает	$\Gamma^{2,5,1}$	$\Gamma^{2,4}$

«Информация» описывает «Физические объекты», так что соответствующим образом «Роль» описывает «Субъектность» или «Объектность» «Физических объектов» при выполнении «Технологических процессов», а «Состояние» описывает те состояния, в которых могут находиться такие «Физические объекты»:

$$R_4(\forall \Gamma^4, \exists \Gamma^2), \quad (44)$$

$$R_{4,1}(\forall \Gamma^{4,1}, \exists \Gamma^2), \quad (45)$$

$$R_{4,1,1}(\exists \Gamma^{4,1,1}, \exists \Gamma^2), \quad (46)$$

$$R_{4,1,2}(\exists \Gamma^{4,1,2}, \exists \Gamma^2), \quad (47)$$

$$R_{4,1,3}(\exists \Gamma^{4,1,3}, \exists \Gamma^2), \quad (48)$$

$$R_{4,1,4}(\exists \Gamma^{4,1,4}, \exists \Gamma^2), \quad (49)$$

$$R_{4,2}(\exists \Gamma^{4,2}, \exists \Gamma^2), \quad (50)$$

Таблица 12

Роли концептов «Информация» и «Физический объект»

№ п/п	Отношение/роль	Наименование	Область	Диапазон
4	R_4	Описывает	Γ^4	Γ^2
4.1	$R_{4,1}$	Описывает состояние	$\Gamma^{4,1}$	Γ^2
4.1.1	$R_{4,1,1}$	Описывает пространственное состояние	$\Gamma^{4,1,1}$	Γ^2
4.1.2	$R_{4,1,2}$	Описывает технологическое состояние	$\Gamma^{4,1,1}$	Γ^2
4.1.3	$R_{4,1,3}$	Описывает эксплуатационное состояние	$\Gamma^{4,1,1}$	Γ^2
4.1.4	$R_{4,1,4}$	Описывает экономическое состояние	$\Gamma^{4,1,1}$	Γ^2
4.2	$R_{4,2}$	Описывает роль	$\Gamma^{4,2}$	Γ^2

Выполнение «Технологических процессов» изменяет «Информацию» о «Физических объектах», а в частности любые «Взаимодействия» приводят к изменению их «Состояний», так что:

$$R_5(\forall \Gamma^1, \exists \Gamma^4), \quad (51)$$

$$R_{5,1}(\forall \Gamma^{1,1}, \exists \Gamma^{4,1}), \quad (52)$$

$$R_{5,1,1}(\exists \Gamma^{1,1}, \exists \Gamma^{4,1,1}), \quad (53)$$

$$R_{5,1,2}(\exists \Gamma^{1,1}, \exists \Gamma^{4,1,2}), \quad (54)$$

$$R_{5,1,3}(\exists \Gamma^{1,1}, \exists \Gamma^{4,1,3}), \quad (55)$$

$$R_{5,1,4}(\exists \Gamma^{1,1}, \exists \Gamma^{4,1,4}), \quad (56)$$

Таблица 13

Роли концептов «Технологический процесс» и «Информация»

№ п/п	Отношение/роль	Наименование	Область	Диапазон
5	R_5	Изменяет	Γ^1	Γ^4
5.1	$R_{5,1}$	Изменяет состояние	$\Gamma^{1,1}$	$\Gamma^{4,1}$
5.1.1	$R_{5,1,1}$	Изменяет пространственное состояние	$\Gamma^{1,1}$	$\Gamma^{4,1,1}$
5.1.2	$R_{5,1,2}$	Изменяет технологическое состояние	$\Gamma^{1,1}$	$\Gamma^{4,1,2}$
5.1.3	$R_{5,1,3}$	Изменяет эксплуатационное состояние	$\Gamma^{1,1}$	$\Gamma^{4,1,3}$
5.1.4	$R_{5,1,4}$	Изменяет экономическое состояние	$\Gamma^{1,1}$	$\Gamma^{4,1,4}$

Каждый «Агент управления» управляет каким-либо «Физическим объектом», определяет «Технологический процесс» и использует «Состояния» как «Входной или выходной параметр» «Функции»:

$$R_6(\forall \Gamma^3, \exists \Gamma^2), \quad (57)$$

$$R_7(\forall \Gamma^3, \exists \Gamma^1), \quad (58)$$

$$R_8(\forall \Gamma^3, \exists \Gamma^4), \quad (59)$$

$$R_{8,1}(\forall \Gamma^{3,1}, \exists \Gamma^4), \quad (60)$$

$$R_{8,2}(\forall \Gamma^{3,1}, \exists \Gamma^4), \quad (61)$$

Таблица 14

Роли концептов «Агент управления» и «Физический объект», «Технологический процесс», «Информация»

№ п/п	Отношение/роль	Наименование	Область	Диапазон
6	R_6	Управляет	Γ^3	Γ^2
7	R_7	Определяет	Γ^3	Γ^1
8	R_8	Использует	Γ^3	Γ^4

№ п/п	Отношение/роль	Наименование	Область	Диапазон
8.1	$R_{8,1}$	Использует как входной параметр	$\Gamma^{3,1}$	Γ^4
8.2	$R_{8,2}$	Использует как выходной параметр	$\Gamma^{3,1}$	Γ^4

На основании введенных Таблиц 5-14 и формул (23)-(61), представляющих собой набор аксиоматических высказываний, может быть сформирована онтология общего назначения архитектуры кибер-физической пространственно-распределенной геотехнической индустриальной системы. Исходя из введенных концептов и ролей можно формировать явные определения для отдельных компонентов архитектуры. Так, в качестве примера, можно вывести определение F_{ROI} для концепта «Технологический процесс»:

$$\begin{aligned}
& (\Gamma^1, (R_0(\Gamma^1, \Gamma^{1,1} \sqcap R_0(\Gamma^{1,2}, \Gamma^{1,2,1} \sqcap \Gamma^{1,2,2} \sqcap \Gamma^{1,2,3}))) \sqcap \\
& (R_1(\Gamma^2, \Gamma^1) \sqcap R_{1,1}(\Gamma^2, \Gamma^{1,1}) \sqcap R_{1,2}(\Gamma^2, \Gamma^{1,1}) \\
& \sqcap R_5(\Gamma^1, \Gamma^4) \sqcap R_{5,1}(\Gamma^{1,1}, \Gamma^{4,1}) \sqcap R_{5,1,1}(\Gamma^{1,1}, \Gamma^{4,1,1}) \sqcap R_{5,1,2}(\Gamma^{1,1}, \Gamma^{4,1,2}) \\
& \sqcap R_{5,1,3}(\Gamma^{1,1}, \Gamma^{4,1,3}) \sqcap R_{5,1,4}(\Gamma^{1,1}, \Gamma^{4,1,4})) \\
& \}, \tag{62}
\end{aligned}$$

что семантически может быть определено как

«Технологический процесс»:

- «Включает» «Взаимодействи» $\Gamma^{1,1}$;
- «Включает» «Процесс» $\Gamma^{1,2}$, «Основной» $\Gamma^{1,2,1}$, «Вспомогательный» $\Gamma^{1,2,2}$ и «Обеспечивающий» $\Gamma^{1,2,3}$;
- «Выполняется» «Физический объект» R_1 ;
- «Имеет» «Субъект» $R_{1,1}$;
- «Имеет» «Объект» $R_{1,2}$;
- «Изменяет» «Информация» R_5 ;
- «Изменяет» «Состояние» $R_{5,1}$;
- «Изменяет» «Пространственное состояние» $R_{5,1,1}$;
- «Изменяет» «Технологическое состояние» $R_{5,1,2}$;
- «Изменяет» «Эксплуатационное состояние» $R_{5,1,3}$;
- «Изменяет» «Экономическое состояние» $R_{5,1,4}$.

Заложенные иерархия концептов и роли, с одной стороны, в достаточной мере покрывают общие знания об инвариантной архитектуре системы подобного типа, а с другой потенциально обеспечивают возможность вывода новых (скрытых) знаний о системе при формировании соответствующих логико-семантических обращений. Следует уточнить, что в определенном смысле данная онтология, как и любая другая, не может характеризоваться полнотой, а формируемый на основании ролевых аксиом семантический граф может быть расширен и углублен при необходимости составления более детального описания системы.

В контексте настоящего исследования предполагается использование онтологии общего назначения как универсального ядра онтологической модели архитектуры, обеспечивающего консистентность и когерентность объективных знаний, при этом не ставится цели по формированию наиболее точной (с точки зрения субъективной семантики концептов и ролей) и полной (с точки зрения степени детализации) модели архитектуры. В соответствии с концептуальной схемой управления автономно функционирующей кибер-физической индустриальной системой (Рисунок 12) предполагается, что полнота и точность онтологической модели может достигаться только в случае её коллегиального экспертного наполнения при участии множества заинтересованных лиц.

2.3.2 Разработка онтологии специального назначения

В рамках настоящего исследования под онтологией специального назначения подразумевается составление такой части онтологической модели, которая позволяла бы явно характеризовать знания о конкретной индустриальной системе и выводить необходимые знания о структурно-функциональной конфигурации её архитектуры для создания каталогов атомарных функциональных задач управления Φ , физических узлов размещения Υ и их графа связи q_2 . Для этой цели в качестве специальной части онтологической модели предлагается вводить набор индивидов (экземпляров) для конечных классов иерархии концептов, описывающих типовые сущности индустриальной системы (на примере предприятия по открытой добыче полезных ископаемых) и понятия из теории АСУ. При этом для каждого из введённого экземпляра должны быть явно определены допустимые и недопустимые роли с учётом заданных отношений онтологии общего назначения.

В первую очередь, дополним алфавит онтологии и введем:

$$R_{10}(\Gamma^z, E^e) \in \Omega, \quad (63)$$

где R_{10} отношение типа концепт/субконцепт Γ^z «ИмеетИндивида» E^e .

В соответствии с (63) введём ряд примеров индивидов для описания горнопромышленной системы:

Для конечного в иерархии классов концепта «Взаимодействия» $\Gamma^{1,1}$ определим индивидов, представляющих набор примеров типовых взаимодействий двух физических объектов горнопромышленного предприятия: «Движение» $E^{1,1,1}$, «Доставка» $E^{1,1,2}$, «Перемещение» $E^{1,1,3}$, «Размещение» $E^{1,1,4}$, «Вмещение» $E^{1,1,5}$, «Расположение» $E^{1,1,6}$, «Обработка» $E^{1,1,7}$, «Изготовление» $E^{1,1,8}$, «Изнашивание» $E^{1,1,9}$, «Потребление» $E^{1,1,10}$, «Расходование» $E^{1,1,11}$.

Для концепта «Оборудование» определим индивидов, представляющих набор примеров типовых мобильных объектов горнотранспортного комплекса $\Gamma^{2,1,1}$ и стационарных технологических установок $\Gamma^{2,1,2}$: Самосвал $E^{2,1,1,1}$, Экскаватор $E^{2,1,1,2}$, $E^{2,1,1,3}$, Погрузчик $E^{2,1,1,3}$, Грейдер $E^{2,1,1,4}$, Буровзрывная машина $E^{2,1,1,5}$, Топливозаправщик $E^{2,1,1,6}$, Конвейер $E^{2,1,2,1}$.

Для концепта «Инфраструктура» определим индивидов, представляющих набор примеров типовых геопространственных $\Gamma^{2,2,1}$ и технических $\Gamma^{2,2,2}$ элементов производственной среды: Дорога $E^{2,2,1,1}$, Зона погрузки $E^{2,2,1,2}$, Зона разгрузки $E^{2,2,1,3}$, Зона БВР $E^{2,2,1,3}$, Гараж $E^{2,2,2,1}$, Склад $E^{2,2,2,2}$.

Для концепта «Ресурс» определим индивидов, представляющих набор примеров типовых энергетических $\Gamma^{2,3,1}$ и инструментальных $\Gamma^{2,3,2}$ ресурсов, потребляемых оборудованием: Дизельное топливо $E^{2,3,1,1}$, Электричество $E^{2,3,1,2}$, Взрывчатка $E^{2,3,2,1}$, Вода $E^{2,3,2,2}$.

Для концепта «Продукт» определим индивидов, представляющих набор природных (горно-геологических) $\Gamma^{2,4,1}$ и геотехнических $\Gamma^{2,4,2}$ продуктов, получаемых в результате реализации технологических процессов: Горная порода $E^{2,4,1,1}$, Дорожное покрытие $E^{2,4,2,1}$, Скважина $E^{2,4,2,2}$.

Для концепта «Устройства» определим индивидов, представляющих набор примеров типовых устройств пространственного $\Gamma^{2,5,1}$, технологического $\Gamma^{2,5,2}$ и эксплуатационного $\Gamma^{2,5,3}$ взаимодействий, являющихся составными частями оборудования: Колесное шасси $E^{2,5,1,1}$, Гусеничное шасси $E^{2,5,1,2}$, Платформа $E^{2,5,1,3}$, Ковш $E^{2,5,2,1}$, Кузов $E^{2,5,2,2}$, Лента $E^{2,5,2,3}$, Бур $E^{2,5,2,4}$, Отвал с ножом $E^{2,5,2,5}$, Цистерна $E^{2,5,2,6}$, Дизельный двигатель $E^{2,5,3,1}$, Электрический двигатель $E^{2,5,3,2}$.

Для концепта «Функция» $\Gamma^{3,1}$ определим индивидов, представляющих конечный набор типовых классов вычислительных задач Теории АСУ: Детектирование $E^{3,1,1}$,

Идентификация $E^{3,1,2}$, Прогнозирование $E^{3,1,3}$, Планирование $E^{3,1,4}$, Управление $E^{3,1,5}$, Моделирование $E^{3,1,6}$.

Для концепта «Горизонт принятия решений» $\Gamma^{3,2}$ определим индивидов, представляющих конечный набор типовых временных отрезков теории управления (организации производств): Оперативный $E^{3,2,1}$, Тактический $E^{3,2,2}$ и Стратегический $E^{3,2,3}$.

Для концепта «Количество объектов управления» $\Gamma^{3,3}$ определим индивидов, представляющих конечный набор типовых группировок объектов управления: Экземпляр $E^{3,3,1}$ и Группа $E^{3,2,3}$.

Для концептов «Пространственное» $\Gamma^{4,1,1}$, «Технологическое» $\Gamma^{4,1,2}$, «Эксплуатационное» $\Gamma^{4,1,3}$ и «Экономическое» $\Gamma^{4,1,4}$ состояния определим индивидов, представляющих набор примеров информационных состояний объектов горнопромышленного предприятия, которые могут быть описаны наборами конкретных данных в виде измеряемых физико-технических, физико-геологических и экономических параметров: Местоположение $E^{4,1,1,1}$, Положение $E^{4,1,1,2}$, Производительность $E^{4,1,2,1}$, Вместимость $E^{4,1,2,2}$, ЭнергоОбъем $E^{4,1,2,3}$, Энергоемкость $E^{4,1,2,4}$, Массогабаритность $E^{4,1,2,5}$, Работоспособность $E^{4,1,3,1}$, Консистенция $E^{4,1,3,2}$, Конфигурация $E^{4,1,3,3}$, Целостность $E^{4,1,3,4}$, Электрический ток $E^{4,1,3,4}$, Стоимость $E^{4,1,4,1}$.

Для концепта «Время» $\Gamma^{4,2}$ определим конечный набор типовых восприятий временных информационных объектов, формализующих принципы решения вычислительных задач управления: Исторический момент $E^{4,2,1}$, Текущий момент $E^{4,2,2}$, Прогнозный момент $E^{4,2,3}$, Плановый момент $E^{4,2,4}$, Фиктивный момент $E^{4,2,5}$, Оперативный интервал $E^{4,2,6}$, Тактический интервал $E^{4,2,7}$, Стратегический интервал $E^{4,2,8}$.

Для концепта «Роль» $\Gamma^{4,3}$ определим ролевые модели физических объектов при взаимодействии: Субъект $E^{4,3,1}$ и Объект $E^{4,3,2}$.

Для каждого конкретного экземпляра аксиоматически определяется конкретный набор ролевых отношений с другими экземплярами, концептами или условия отрицания таких отношений:

$$R_r(E^\epsilon, E^\epsilon) \in \Omega, \quad (64)$$

$$R_r(E^\epsilon, \Gamma^\zeta) \in \Omega, \quad (65)$$

$$\neg R_r(E^\epsilon, E^\epsilon) \in \Omega, \quad (66)$$

$$\neg R_r(E^\epsilon, \Gamma^\zeta) \in \Omega, \quad (67)$$

Приведём таких ролевых отношений для индивида взаимодействия «Доставка»:

- изменяет пространственное состояние субъекта «Местоположение» $R_{5,1,1,1}(E^{1,1,2}, E^{4,1,1,1})$;
- изменяет пространственное состояние объекта «Положение» $R_{5,1,1,2}(E^{1,1,2}, E^{4,1,1,2})$;
- изменяет технологическое состояние субъекта «Производительность» $R_{5,1,2,1}(E^{1,1,2}, E^{4,1,2,1})$;
- изменяет технологическое состояние объекта «Массогабаритность» $R_{5,1,2,2}(E^{1,1,2}, E^{4,1,2,5})$;
- не изменяет эксплуатационное состояние субъекта $\neg R_{5,1,3,1}(E^{1,1,2}, \forall \Gamma^{4,1,3})$;
- не изменяет эксплуатационное состояние объекта $\neg R_{5,1,3,2}(E^{1,1,2}, \forall \Gamma^{4,1,3})$;
- не изменяет экономическое состояние субъекта $\neg R_{5,1,4,1}(E^{1,1,2}, \forall \Gamma^{4,1,4})$;
- изменяет экономическое состояние объекта «Стоимость» $R_{5,1,4,1}(E^{1,1,2}, E^{4,1,4,1})$;
- имеет субъекта взаимодействия «Самосвал» $R_{1,1}(E^{2,1,1,1}, E^{1,1,2})$;
- имеет объекта взаимодействия «Горная порода» $R_{1,2}(E^{2,4,1,1}, E^{1,1,2})$.

По аналогии с приведенным примером может быть описана структурно-функциональная архитектура кибер-физической пространственно-распределенной геотехнической индустриальной системы. Семантические триплеты ролей отдельных индивидов задают строгую формальную конструкцию с возможностью исчислений для проверки истинности и непротиворечивости. В то же время все явно незадекларированные (скрытые аксиомы) для конкретных индивидов роли классов распространяются на все индивиды таких классов, что может быть выведено с использованием механизмов «рассуждений» в онтологиях, таких как HermiT Reasoner языка OWL.

2.4 Разработка алгоритмических механизмов формирования метамодели архитектуры

2.4.1 Разработка процедур логико-семантического вывода из онтологии

Несмотря на высокую выразительность и доказанную разрешимость современной дескрипционной логики и языка OWL, а также наличие достаточно эффективных инструментальных средств формальных запросов к онтологиям SPAQRL Query для вывода явных или скрытых аксиоматических знаний, их применение в рамках настоящего исследования имеет ряд ограничений, а именно:

- предлагаемая онтологическая модель Ω в сущности состоит из множества двуместных и трехместных предикатов. Языки OWL и SPAQRL Query допускают составные предикативные конструкции с использованием традиционных операций конъюнкции-дизъюнкции и квантификаторов существования-всеобщности для вывода, удовлетворяющего условиям одновременной истинности всех предикатов. Однако формирование более сложных предикативных конструкций с внутренним вложением семантических триплетов не допустимо в целях гарантированной разрешимости логического вывода.
- из онтологии возможен вывод только тех явных или скрытых аксиоматических высказываний, которые задекларированы через концепты, роли и индивиды, или выводимы с использованием рассуждателей. Получение же знаний, не охватываемых онтологией по содержанию, но предполагаемых по её семантическому назначению, фактически невозможно в силу принципиальной сущности самих онтологий и Баз Знаний.

Имеется в виду, что для получения каталогов атомарных функциональных задач управления Φ и физических узлов Y из онтологии необходимо либо их явно объявить и описать в самой онтологии, либо найти способ экстракции за счет иных средств логико-семантического вывода $F_{Rol}(\Omega) = (\Phi, Y, g_2)$. Введение данных каталогов (в частности речь об атомарных функциональных задачах управления) как концептов и индивидов онтологии невозможно в виду их исходной сложной семантической конструкции, не декларируемой простыми двуместными и трехместными предикатами. Более того, составление полного перечня всех возможных индивидов является крайней трудозатратным процессом, а декларирование концептов со сложной семантикой может приводить к тем же ошибкам, что и традиционный способ экспертно-аналитического проектирования моделей архитектур – субъективности, неоднозначности и неполноте.

С учётом предложенного ранее определения атомарной функциональной задачи управления введём более строгую формальную логико-семантическую конструкцию. Для начала определим, что:

атомарная функциональная задача управления $\varphi_n \in \Phi$ это то, что вычисляет для агента управления на заданном горизонте принятия решения значение функции для заданного числа физических объектов, выполняющих допустимое взаимодействие в роли субъекта и объекта, изменяющее их состояния, и использующее для этого в качестве входной информации такие состояния во времени, соответствующему функции, а в качестве выходной информации состояния субъекта или объекта во времени, соответствующему функции.

Очевидно, что подобная семантика определения достаточно сложна, как для человеческого восприятия, так и для составления логической конструкции на языке OWL. Разобьём данную конструкцию на отдельные предикаты:

- «Вычисляет» на заданном «Горизонт принятия решения» $\Gamma^{3,2}$;
- «Вычисляет» значение «Функция» $\Gamma^{3,1}$;
- «Вычисляет» для заданного «Число объектов» $\Gamma^{3,3}$;
- «Физический объект» Γ^2 , «Выполняют допустимое «Взаимодействие» в «Роль» «Субъект» $R_{1,1}(\exists \Gamma^2, \exists \Gamma^{1,1})$ и «Объект» $R_{1,2}(\exists \Gamma^2, \exists \Gamma^{1,1})$: $R_{1,1}(\exists \Gamma^2, \exists \Gamma^{1,1}) \sqcap R_{1,2}(\exists \Gamma^2, \exists \Gamma^{1,1})$;
- «Взаимодействие» «Изменяет состояние» $R_{5,1}(\exists \Gamma^{1,1}, \exists \Gamma^{4,1})$;
- «Состояние» $R_{4,1}(\exists \Gamma^2, \exists \Gamma^{4,1})$;
- «Использует входную информацию» «Состояние» $R_{8,1}(\exists \Gamma^{3,1}, \exists \Gamma^{4,1})$ во «Время» $R_{8,1}(\exists \Gamma^{3,1}, \exists \Gamma^{4,2})$;

$$(R_{8,1}(\exists \Gamma^{3,1}, \exists \Gamma^{4,1} \sqcap R_{4,1}(\exists \Gamma^2 \sqcap R_{1,1}(\exists \Gamma^2, \exists \Gamma^{1,1}), \exists \Gamma^{4,1}))) \sqcap R_{8,1}(\exists \Gamma^{3,1}, \exists \Gamma^{4,2})) \sqcap$$

$$(R_{8,1}(\exists \Gamma^{3,1}, \exists \Gamma^{4,1} \sqcap R_{4,1}(\exists \Gamma^2 \sqcap R_{1,2}(\exists \Gamma^2, \exists \Gamma^{1,1}), \exists \Gamma^{4,1}))) \sqcap R_{8,1}(\exists \Gamma^{3,1}, \exists \Gamma^{4,2}));$$

- «Использует выходную информацию» «Состояние» $R_{8,2}(\exists \Gamma^{3,1}, \exists \Gamma^{4,1})$ «Субъект» или «Объект» во «Время» $R_{8,1}(\exists \Gamma^{3,1}, \exists \Gamma^{4,2})$:

$$(R_{8,2}(\exists \Gamma^{3,1}, \exists \Gamma^{4,1} \sqcap R_{4,1}(\exists \Gamma^2 \sqcap R_{1,1}(\exists \Gamma^2, \exists \Gamma^{1,1}), \exists \Gamma^{4,1}))) \sqcap R_{8,2}(\exists \Gamma^{3,1}, \exists \Gamma^{4,2})) \sqcup$$

$$(R_{8,2}(\exists \Gamma^{3,1}, \exists \Gamma^{4,1} \sqcap R_{4,1}(\exists \Gamma^2 \sqcap R_{1,2}(\exists \Gamma^2, \exists \Gamma^{1,1}), \exists \Gamma^{4,1}))) \sqcap R_{8,2}(\exists \Gamma^{3,1}, \exists \Gamma^{4,2})).$$

Тогда полная запись такого выражения может быть представлена в следующем виде φ_n :

$$\begin{aligned}
& \exists \Gamma^{3,2} \sqcap \exists \Gamma^{3,1} \sqcap \exists \Gamma^{3,3} \sqcap \exists (R_{8,1} (\exists \Gamma^{3,1}, \exists \Gamma^{4,1} \sqcap R_{4,1} (\exists \Gamma^2 \sqcap R_{1,1} (\exists \Gamma^2, \exists \Gamma^{1,1}), \exists \Gamma^{4,1}))) \\
& \sqcap R_{8,1} (\exists \Gamma^{3,1}, \exists \Gamma^{4,2})) \\
& \sqcap (R_{8,1} (\exists \Gamma^{3,1}, \exists \Gamma^{4,1} \sqcap R_{4,1} (\exists \Gamma^2 \sqcap R_{1,2} (\exists \Gamma^2, \exists \Gamma^{1,1}), \exists \Gamma^{4,1}))) \\
& \sqcap R_{8,1} (\exists \Gamma^{3,1}, \exists \Gamma^{4,2})) \\
& \sqcap \exists (R_{8,2} (\exists \Gamma^{3,1}, \exists \Gamma^{4,1} \sqcap R_{4,1} (\exists \Gamma^2 \sqcap R_{1,1} (\exists \Gamma^2, \exists \Gamma^{1,1}), \exists \Gamma^{4,1}))) \\
& \sqcap R_{8,2} (\exists \Gamma^{3,1}, \exists \Gamma^{4,2})) \\
& \sqcup (R_{8,2} (\exists \Gamma^{3,1}, \exists \Gamma^{4,1} \sqcap R_{4,1} (\exists \Gamma^2 \sqcap R_{1,2} (\exists \Gamma^2, \exists \Gamma^{1,1}), \exists \Gamma^{4,1}))) \\
& \sqcap R_{8,2} (\exists \Gamma^{3,1}, \exists \Gamma^{4,2})), \tag{68}
\end{aligned}$$

В соответствии с (68) определим механизм логико-семантического вывода $F_{Rol}(\Omega)$ каталога атомарных функциональных задач управления Φ , представляющий из себя перебор всех допустимых комбинаций, удовлетворяющих условиям для существующих индивидов в онтологии:

$$\Phi = \{\varphi_N \mid \varphi_n \sqsubseteq R_r(\forall \Gamma^\zeta, \exists E^\epsilon)\}, \tag{69}$$

где φ_n – это логико-семантическая конструкция, обладающая уникальным наименованием на базе имен индивидов составных концептов (Таблица 15) и набором атрибутов в соответствии с комбинаторным выводом, определяющих структурно-функциональную конфигурацию атомарной функциональной задачи управления (Таблица 16); а знак « $\sqsubseteq$ » обозначает отношение «включения в себя».

Таблица 15

Конструкция наименования атомарной функциональной задачи управления

Название концепта индивида	Значение
Количество объектов управления	$\exists E^{3,3}$
Горизонт принятия решения	$\exists E^{3,2}$
Функция	$\exists E^{3,1}$
Изменяемые (выходные) состояния субъекта или объекта	
Пространственное состояние	$\exists E^{4,1,1}$
Технологическое состояние	$\exists E^{4,1,2}$
Эксплуатационное состояние	$\exists E^{4,1,3}$
Экономическое состояние	$\exists E^{4,1,4}$

Название концепта индивида	Значение
Физический объект с ролью субъекта или объекта	$\exists E^2$
Взаимодействие	$\exists E^{1,1}$
Физический объект с ролью объекта или субъекта	$\exists E^2$

Таблица 16

Структурно-функциональная конфигурация атомарной функциональной задачи управления

Название концепта индивида	Значение
Количество объектов управления	$\exists E^{3,3}$
Горизонт принятия решения	$\exists E^{3,2}$
Функция	$\exists E^{3,1}$
Назначение по роли	$E^{4,3,1} \sqcup E^{4,3,2}$
Взаимодействие	$\exists E^{1,1}$
Входное время	$\exists E^{4,2}$
Наименование субъекта	$\exists E^2$
Входные состояния субъекта	
Пространственное состояние субъекта	$\exists E^{4,1,1}$
Технологическое состояние субъекта	$\exists E^{4,1,2}$
Эксплуатационное состояние субъекта	$\exists E^{4,1,3}$
Экономическое состояние субъекта	$\exists E^{4,1,4}$
Наименование объекта	$\exists E^2$
Входные состояния объекта	
Пространственное состояние	$\exists E^{4,1,1}$
Технологическое состояние	$\exists E^{4,1,2}$
Эксплуатационное состояние	$\exists E^{4,1,3}$
Экономическое состояние	$\exists E^{4,1,4}$
Выходное время	$\exists E^{4,2}$
Выходные состояния субъекта или объекта	
Пространственное состояние субъекта	$\exists E^{4,1,1}$
Технологическое состояние субъекта	$\exists E^{4,1,2}$
Эксплуатационное состояние субъекта	$\exists E^{4,1,3}$
Экономическое состояние субъекта	$\exists E^{4,1,4}$

Таким образом при комбинаторном переборе всех допустимых вариантов может быть реализован логико-семантический вывод полного перечня атомарных функциональных задач управления, формирующих соответствующий проиндексированный каталог со строгой уникальной конфигурацией каждой задачи. Так

что степень полноты каталога будет зависеть от ширины и глубины детализации составленной онтологической модели, но обеспечена строгим формальным выводом.

Для вывода $u_m^{l\wedge s}$ же каталога физических узлов Y определим следующее:

- атомарные функциональные задачи оперативного управления $E^{3,2,1}$ оборудованием $E^{2,1}$ всегда должны полностью реализовываться локально u^l на таком оборудовании:

$$\{u_m^l \mid u_m^l \equiv \exists E^{2,1}, \varphi_n \sqsubseteq \exists(E^{3,2,1} \sqcap \exists E^{2,1})\} \rightarrow (\{\varphi_n\} \multimap \{u_m^l\}), \quad (70)$$

где знак « $\equiv$ » обозначает отношение «эквивалентности» или «соответствия» элементов.

- атомарные функциональные задачи тактического $E^{3,2,2}$ и стратегического $E^{3,2,3}$ управления оборудованием всегда должны быть представлены на таком оборудовании, но могут реализовываться распределенно («клиент-серверно»):

$$\{u_m^{l\vee s} \mid (u_m^l \equiv \exists E^{2,1}) \sqcup (u_m^s \equiv E^{2,0}), \varphi_n \sqsubseteq \exists((E^{3,2,2} \sqcup E^{3,2,3}) \sqcap \exists E^{2,1})\} \rightarrow (\{\varphi_n\} \multimap \{u_m^l \vee u_m^s\}), \quad (71)$$

- все остальные атомарные функциональные задачи управления реализуются там, где это допустимо (есть свободные вычислительные ресурсы физического узла) и/или эффективно (по оценке логической и/или технической конфигурации архитектуры):

$$\{u_m^{l\vee s} \mid (u_m^l \equiv \exists E^{2,1}) \sqcup (u_m^s \equiv E^{2,0})\} \rightarrow (\{\varphi_n\} \multimap \{u_m^l \vee u_m^s\}), \quad (72)$$

Тогда механизм логико-семантического вывода $F_{ROI}(\Omega)$ каталога физических узлов может быть представлен в следующем виде:

$$Y = \{u_M^{l\wedge s} \mid u_M^{l\wedge s} \sqsubseteq \exists \varphi_n, \quad (73)$$

А, соответственно, для логико-семантического вывода графа g_2 необходимо воспользоваться уравнениями (70)-(72): $(\{\varphi_N\} \multimap \{u_M^{l\wedge s}\}) \rightarrow g_2$.

2.4.2 Алгоритм формирования метамодели архитектуры

В соответствии со схемой на Рисунке 12 и предложенными механизмами логико-семантического вывода (68)-(73) разработан алгоритм формирования метамодели архитектуры автономно функционирующей кибер-физической пространственно-распределенной геотехнической индустриальной системы, представленный в виде блок-схемы на Рисунке 13.

Назначением данного алгоритма является формализация процесса построения модели «as is» архитектуры индустриальной системы с учётом экспертно-аналитических и строгих логико-семантических процедур манипуляции знаниями для их представления в виде набора каталогов и графов связи, определяющих структурно-функциональную конфигурацию предприятия.

Особенностью алгоритма является регламентирование процедур логико-семантического вывода знаний из онтологической модели и конфигурации архитектуры в виде графа в целях обеспечения возможности оценки и анализа качества такой структуры для её последующей оптимизации на отдельных этапах жизненного цикла цифровой трансформации.

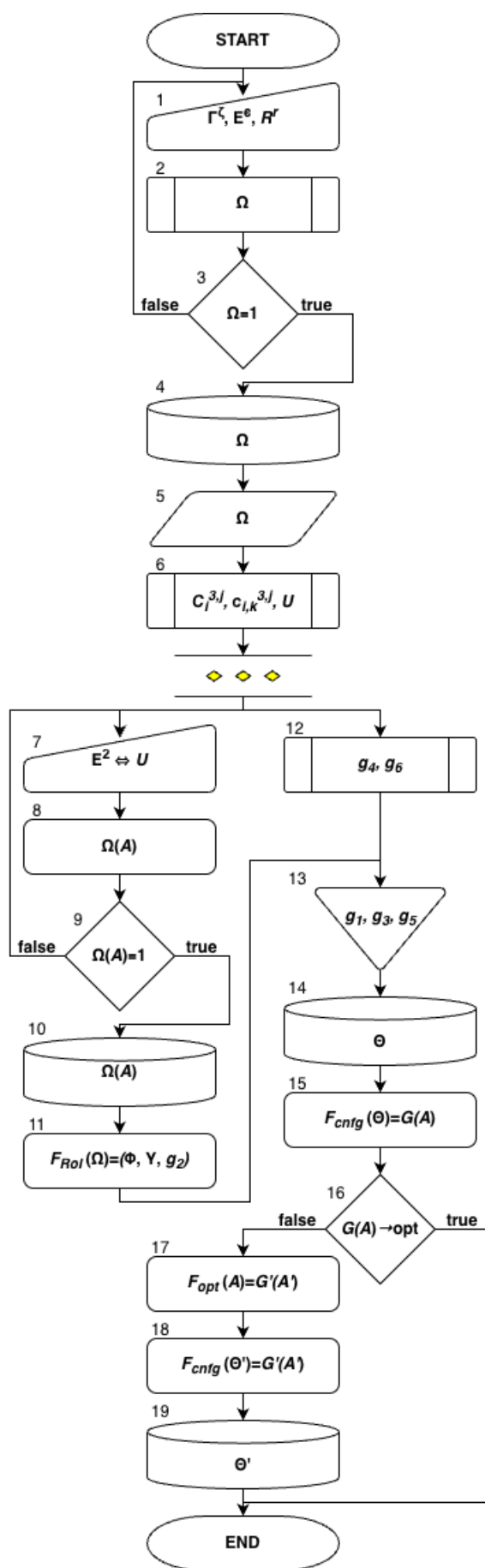


Рисунок 13. Блок-схема алгоритма формирования метамодели архитектуры автономно функционирующей кибер-физической пространственно-распределенной геотехнической индустриальной системы

Описание блок-схемы алгоритма:

START

Начало работы алгоритма.

Блок 1. Ручной ввод данных

Составление концептов Γ^{ζ} , индивидов E^{ϵ} и ролей R_r отраслевыми экспертами для формирования унифицированной онтологии архитектуры пространственно-распределенной геотехнической индустриальной системы с открытым типом технологической среды.

Блок 2. Предопределенный процесс

Формирование онтологии Ω на базе введенных предикатов с использованием формальной (дескрипционной) логики и программных инструментальных средств работы с Базами Знаний (например, среда Protégé для языка OWL).

Блок 3. Условие

Проверка консистентности и когерентности онтологии с использованием формальной логики и программных инструментальных средств – рассуждателей (например, HermiT). В случае, если все введение аксиоматические высказывания имеют конечную разрешимость и непротиворечивы, то переход к Блоку 5, иначе возврат к Блоку 2 для пересмотра концептов Γ^{ζ} , индивидов E^{ϵ} и ролей R_r экспертами.

Блок 4. Сохранение результата

Сохранение онтологии Ω в облачном репозитории для организации доступа всем заинтересованным сторонам.

Блок 5. Ввод данных

Автоматизированная процедура выгрузки онтологии Ω из облачного репозитория для её имплементации в цифровом пространстве конкретного предприятия.

Блок 6. Предопределенный процесс

Экспертно-аналитическая или автоматизированная процедура экстракции каталогов агентов управления $C_i^{3,j=[0,1]}$, экземпляров агентов управления $c_{i,k}^{3,j}$ и экземпляров физических узлов $U \in C_i^2$ из распределенного хранилища информации предприятия.

Параллелизация алгоритма: Блоки 7-11 и Блок 12.

Блок 7. Ручной ввод данных

Экспертно-аналитическое сопоставление фактически существующих экземпляров физических узлов U на конкретном предприятии с индивидами концептов онтологии $E^{2,1}$ для исключения нерелевантных знаний с точки зрения организации действующих технологических процессов.

Блок 8. Процесс

Программно-инструментальный рефакторинг онтологии для получения онтологической модели предприятия $\Omega(A)$ с учётом внесённых изменений в перечень индивидов.

Блок 9. Условие

Проверка консистентности и когерентности онтологической модели предприятия с использованием формальной логики и программных инструментальных средств – рассуждателей. В случае, если изменения аксиоматических высказываний имеют конечную разрешимость и непротиворечивы, то переход к Блоку 10, иначе возврат к Блоку 7.

Блок 10. Сохранение результата

Сохранение онтологической модели предприятия $\Omega(A)$ в локальном репозитории цифрового пространства.

Блок 11. Процесс

Процесс экстракции из онтологической модели предприятия $\Omega(A)$ каталога атомарных функциональных задач управления Φ , каталога физических узлов Υ и их графа связи g_2 на базе программно-реализуемых механизмов логико-семантического вывода F_{Rol} .

Блок 12. Предопределенный процесс

Экспертно-аналитическая или автоматизированная процедура сопоставления каталогов агентов управления $C_i^{3,j=[0,1]}$, экземпляров агентов управления $c_{i,k}^{3,j}$ и экземпляров физических узлов U для формирования их графов связи g_4, g_6 .

Блок 13. Слияние

Экспертно-аналитическая или автоматизированная процедура сопоставления каталогов $\Phi, \Upsilon, C_i^{3,j}, c_{i,k}^{3,j}, U$ и графов g_2, g_4, g_6 для формирования графов связи g_1, g_3, g_5 . Индексирование всех сущностей в каталогах (формирование метамодели архитектуры Θ).

Блок 14. Сохранение результата

Сохранение каталогов $\Phi, \Upsilon, C_i^{3,j}, c_{i,k}^{3,j}, U$ и графов связи $\{g_1, \dots, g_6\}$ как единой метамодели архитектуры кибер-физической пространственно-распределенной геотехнической промышленной системы Θ в локальном репозитории цифрового пространства.

Блок 15. Процесс

Программная процедура конфигурации архитектуры предприятия $F_{cnfg}(\Theta)$ в виде получения графа $G(A)$ на базе метамодели Θ .

Блок 16. Условие

Оценка и анализ полученной конфигурации архитектуры в виде графа $G(A)$ на соответствие критерию оптимальности. В случае, если (граф) архитектура оптимальна, то переход к **END** (завершению алгоритма), иначе переход к Блоку 17.

Блок 17. Процесс

Программная процедура модификации структурно-функциональной конфигурации архитектуры предприятия $F_{opt}(A)$ в виде графа $G(A)$ для получения графа $G'(A')$, удовлетворяющего критерию оптимальности.

Блок 18. Процесс

Программная процедура конфигурации архитектуры $F_{cnfg}(\Theta')$ оптимального графа архитектуры $G'(A')$ из модифицированной метамодели Θ' в целях последующей передачи управляющих воздействий на модификацию компонентов $C_i^{3,j}, c_{i,k}^{3,j}, U$ в действующей архитектуре предприятия A .

Блок 19. Сохранение результата

Сохранение модифицированной метамодели архитектуры автономно функционирующей кибер-физической пространственно-распределенной геотехнической индустриальной системы Θ' в локальном репозитории цифрового пространства.

END

Завершение алгоритма

2.5 Выводы по Главе 2

Во второй главе разработаны формальные модели представления и критерии оценки процессов конфигурирования архитектур для построения автономно функционирующих кибер-физических пространственно-распределенных геотехнических промышленных систем в условиях цифровой трансформации. Предложенный подход основан на комплексном применении онтологического моделирования для формализации унифицированных инвариантных знаний о предприятии и механизмов логико-семантического вывода для построения структурно-функциональной метамодели архитектуры как совокупности каталогов компонентов и графов связи между ними.

В разделе 2.1 сформулированы формальные определения и введены ключевые понятия архитектуры предприятия как совокупности четырёх взаимосвязанных групп компонентов: технологических процессов и операций, физических объектов, агентов управления и информации. Задача управления архитектурой определена как выбор такой её конфигурации, при которой оптимизируется интегральный показатель эффективности предприятия, а цифровая трансформация представлена как непрерывный процесс недетерминированных во времени замен агентов управления на более эффективные программные компоненты. Обоснована необходимость представления компонентов архитектуры в виде каталогов атомарных функциональных задач управления, агентов управления, их экземпляров и физических узлов размещения, а отношений между ними – в виде шести двудольных графов, формирующих описательную метамодель структурно-функциональной конфигурации архитектуры. Введены матрицы отношений связи в состояниях «as is» и «to be», показывающие, что одной из первичных задач цифровой трансформации является нормализация графа связи атомарных функциональных задач управления и агентов управления путём устранения дублирования реализации атомарных функциональных задач разными агентами. Сформулированы три уровня оценки качества архитектуры: структурное качество статической конфигурации, стоимость её модификации между состояниями «as is» и «to be», а также стоимость поддержки жизненного цикла при недетерминированных во времени изменениях.

В разделе 2.2 формализована задача онтологического моделирования архитектуры промышленной системы как способа представления гетерогенных компонентов в виде абстрактных понятий, связанных формальными логическими правилами. Определены четыре ключевые требования к получаемому перечню атомарных функциональных задач управления: достаточная степень атомизации без избыточной декомпозиции, полнота охвата производственной деятельности предприятия (включая процедуры, реализуемые

персоналом), точность определения взаимосвязи компонентов архитектуры и непротиворечивость описания задач. На основании данных требований предложено использование онтологической модели как инвариантного инструмента построения метамodelей архитектуры на отдельных этапах её жизненного цикла. Введены уравнения, определяющие непрерывный процесс цифровой трансформации как последовательность дискретных шагов модификации архитектуры от состояния «as is» к состоянию «to be», с обязательной проверкой сохранения функциональной пригодности системы на каждом шаге. Установлено, что достижимость механизма логико-семантического вывода каталогов компонентов архитектуры и их графов связи из онтологии обеспечивается только при условии консистентности и когерентности онтологической модели, понимаемых как истинность и непротиворечивость всех заложенных аксиоматических высказываний.

В разделе 2.3 разработана двухкомпонентная онтологическая модель архитектуры горнопромышленной системы на языке OWL в соответствии с принципами дескриптивной логики. В подразделе 2.3.1 предложена онтология общего назначения, систематически определяющая иерархию концептов типовой кибер-физической пространственно-распределенной геотехнической индустриальной системы. Сформирована иерархия из сорока классов-концептов по четырём основным ветвям – технологический процесс, физический объект, агент управления и информация – с глубиной детализации до пяти уровней вложенности в ветви «Физический объект». Введены пятьдесят два ролевых отношения, задающих семантические связи между концептами: ролевые отношения выполнения технологических процессов, взаимодействия физических объектов в ролях субъекта и объекта, размещения оборудования на инфраструктуре, содержания ресурсов в устройствах, описания состояний информацией, изменения состояний при взаимодействиях и управления физическими объектами агентами. На основе введённых концептов и ролей получены аксиоматические формулы, формирующие дескриптивно-логическую базу онтологии общего назначения и обеспечивающие возможность вывода явных определений для отдельных компонентов архитектуры, что продемонстрировано на примере вывода определения концепта «Технологический процесс».

В подразделе 2.3.2 разработана онтология специального назначения, конкретизирующая унифицированную горнопромышленную систему на примере предприятия по открытой добыче полезных ископаемых. Введено шестьдесят три индивида (экземпляра конечных классов иерархии) с явно определёнными ролевыми отношениями. Для каждого индивида аксиоматически определены наборы ролевых отношений с другими индивидами, что позволяет задать формальную структурно-функциональную конструкцию

архитектуры с возможностью логических исчислений для проверки истинности и непротиворечивости.

В разделе 2.4 разработаны алгоритмические механизмы формирования метамодели архитектуры на основе онтологической модели. В подразделе 2.4.1 обоснованы ограничения применения стандартных средств логического вывода на языке OWL и запросов SPARQL для построения каталогов атомарных функциональных задач управления, связанные с невозможностью декларирования составных семантических триплетов и принципиальной неполнотой онтологической модели по известным теоремам (Теорема о неполноте, Open-World Assumption). Введена более строгая формальная логико-семантическая конструкция атомарной функциональной задачи управления как сложной предикативного выражения с вложенностью. На основе формулы сформулирован механизм логико-семантического вывода каталога атомарных функциональных задач управления как комбинаторного перебора всех допустимых сочетаний индивидов онтологии, обеспечивающий получение проиндексированного каталога со строгой уникальной конфигурацией каждой задачи. Предложены правила формирования каталога физических узлов размещения с учётом семантики атомарных функциональных задач управления: задачи оперативного управления оборудованием реализуются локально, задачи тактического и стратегического управления могут реализовываться распределенно, прочие задачи – с учётом свободных вычислительных ресурсов и эффективности по критериям оптимальности логической и технической конфигурации.

В подразделе 2.4.2 разработан алгоритм формирования метамодели архитектуры автономно функционирующей кибер-физической пространственно-распределенной геотехнической индустриальной системы, представленный в виде блок-схемы. Алгоритм обеспечивает последовательное выполнение процедур ручного ввода концептов и индивидов отраслевыми экспертами, автоматического построения онтологии в среде Protégé, её проверки на консистентность и когерентность рассуждателем HermiT, экспертно-аналитического сопоставления онтологии с фактической номенклатурой конкретного предприятия, автоматизированного рефакторинга онтологической модели, экстракции каталогов и графов связи логико-семантическим выводом, формирования параллельно составляемых графов связи агентов и экземпляров физических узлов, индексирования сущностей и сохранения итоговой метамодели в локальном репозитории цифрового пространства.

Разработанные в главе формальные модели обеспечивают возможность представления архитектуры индустриальной системы в виде совокупности каталогов компонентов и графов связи, при этом центральным элементом алгоритма формирования

метамоделю является процедура конфигурирования архитектуры в виде графа с последующей оценкой её оптимальности и, при необходимости, модификации структурно-функциональной конфигурации до получения оптимального вида. Однако реализация указанных процедур конфигурации и оптимизации требует наличия трёх принципиальных элементов, которые в настоящее время в научной литературе не формализованы.

Во-первых, отсутствует формальный способ представления архитектуры индустриальной системы в виде графа, элементы которого отражали бы разнородные компоненты логической и технической конфигурации, а структура графа – допускала бы применение вычислительных процедур для её количественного анализа.

Во-вторых, отсутствуют критерии и механизмы количественной оценки архитектуры как графовой структуры, позволяющие сопоставлять её конфигурации в состояниях «as is» и «to be» и проводить сравнительный анализ между различными типами архитектур программных систем и различными вариантами их модификации.

В-третьих, отсутствует формальный набор правил модификации архитектуры как графа, обеспечивающий возможность дискретного пошагового преобразования структуры с сохранением корректности логической конфигурации и с количественной оценкой стоимости каждого преобразования.

ГЛАВА 3. РАЗРАБОТКА МОДЕЛЕЙ ОЦЕНКИ И АЛГОРИТМОВ КОНФИГУРИРОВАНИЯ АРХИТЕКТУР

В соответствии с выводами в Главе 2, в настоящей главе будет осуществлена разработка моделей представления, критериев оценки, алгоритмов конфигурирования и модификации архитектур промышленных систем в целях обеспечения решения задач их цифровой трансформации.

3.1 Разработка моделей представления и критериев оценки архитектуры

Пусть любую атомарную функциональную задачу управления φ_n , независимо от типа агента управления $C_i^{3,j=[0,1]}$, можно представить как набор связанных логических процедур $\{x_\alpha, \dots, x_\beta\}$ для вычисления состояний физических объектов, изменяющихся при их взаимодействии с другими физическими объектами в рамках реализуемых технологических процессов и операций предприятия.

Каждая такая атомарная функциональная задача должна быть представлена в виде программной функции в архитектуре предприятия, тогда к организующим её процедурам можно отнести следующие логические классы по аналогии с классами архитектурных слоев программных систем:

- x^p – представление (ввод-вывод) информации, используемой и/или получаемой при реализации атомарной функциональной задачи;
- x^c – вычисление информации, изменяемой при реализации атомарной функциональной задачи;
- x^s – хранение информации, используемой и/или получаемой при реализации атомарной функциональной задачи;
- x^b – бизнес-логика (управление порядком) представления, вычисления и хранения информации в рамках реализации атомарной функциональной задачи.

Так атомарная функциональная задача управления может быть определена в виде вектора $\varphi_n = \langle x^p, x^c, x^s, x^b \rangle$, в котором строгое наличие четырех переменных x каждого класса p, b, c, s является обязательным условием существования атомарной функциональной задачи управления.

Так как атомарную функциональную задачу управления обуславливает связанность логических процедур x^p, x^c, x^s, x^b , а класс x^b определяет порядок выполнения остальных, то между классом x^b и классами x^p, x^c, x^s существуют логические интерфейсы взаимодействия $u_{\alpha\beta}$, так что y^{pb} – интерфейс между классами x^p, x^b ; y^{bc} – интерфейс

между классами x^b, x^c ; y^{bs} – интерфейс между классами x^b, x^s . Для всех других пар классов x^p, x^c ; x^p, x^s ; x^c, x^s логических интерфейсов не существует $y^{pc}, y^{ps}, y^{cs} = 0$.

В таком случае атомарную функциональную задачу управления может представлена как простой неориентированный граф $G(\varphi_n)$ (Рисунок 14), формализующий логическую конфигурацию её архитектуры:

$$G(\varphi_n) = (X, Y) = (\{x^p, x^b, x^c, x^s\}, \{y^{pb}, y^{bc}, y^{bs}\}), \quad (74)$$

где логические классы $X = \{x^p, x^b, x^c, x^s\}$ – вершины или ноды графа, а интерфейсы $Y = \{y^{pb}, y^{bc}, y^{bs}\}$ – рёбра.

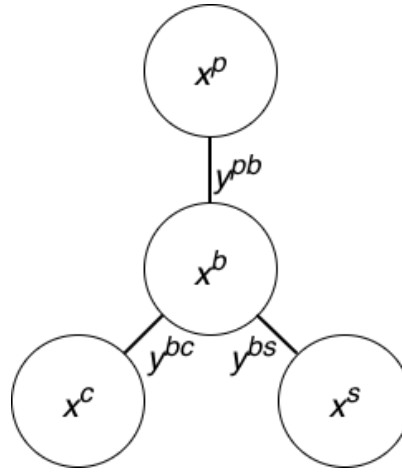


Рисунок 14. Граф логической архитектуры атомарной функциональной задачи управления

Учитывая, что никакая атомарная функциональная задача управления не существует сама по себе и обязательно реализуется каким-либо агентом управления $C_i^{3,j}$ на не менее чем одном физическом узле v_m , полное представление её архитектуры может быть записано в виде матрицы φ_n и графа $G(\varphi_n)$:

$$\begin{aligned} A(\varphi_n) &= (\{\varphi_n\}; G(\varphi_n)) \\ \varphi_n &= \langle x^p, x^b, x^c, x^s, C_i^{3,j}, v_m \rangle, \\ G(\varphi_n) &= (\{x^p, x^b, x^c, x^s\}, \{y^{pb}, y^{bc}, y^{bs}\}) \end{aligned} \quad (75)$$

Однако, когда речь идёт о более сложных архитектурах систем, такой вариант представления логической конфигурации может быть не релевантен, так как необходимо учитывать, что в соответствии с метамоделью Θ :

- архитектура сложной крупномасштабной системы может включать в себя множество агентов управления $A = \{C_1^{3,j}, \dots, C_I^{3,j}\}$;
- каждый из агентов управления, входящих в архитектуру системы, может реализовывать от 1 до N задач $(\{C_i^{3,j}\} \rightarrow \{\varphi_1, \dots, \varphi_N\}) \in g_1$;
- и размещаться на нескольких физических узлах $(\{C_i^{3,j}\} \rightarrow \{v_1, \dots, v_M\}) \in g_3$;
- а произвольный физический узел может размещать более, чем одного агента управления $(\{v_m\} \rightarrow \{C_1^{3,j}, \dots, C_I^{3,j}\}) \in g_3$;
- и, соответственно, более чем одну атомарную функциональную задачу управления $(\{v_m\} \rightarrow \{\varphi_1, \dots, \varphi_N\}) \in g_2$.

При этом каждый агент управления $C_i^{3,j}$ тоже обладает собственной архитектурой $A(C_i^{3,j})$, определяющий принципы конфигурационного размещения классов на локальных или серверных физических узлах, их численность и количество интерфейсов для реализации полного набора закреплённых за таким агентом атомарных функциональных задач управления – $G(C_i^{3,j})$. Под локальной реализацией классов подразумевается их размещение на децентрализованных физических узлах v_m^l . К таким децентрализованным физическим узлам главным образом относится оборудование $v_m^l \in E^{2,1}$ (70), оснащенное бортовыми вычислительными комплексами и участвующее в выполнении технологических процессов и операций. Другим типом физических узлов размещения, очевидно, являются централизованные – серверное оборудование – v_m^s . Выбор узла размещения отдельных атомарных функциональных задач определяется исходным технологическим регламентом и конфигурационными возможностями архитектуры агента управления и/или всей системы в целом на основании метамодели Θ .

В этой связи дополним типизацию классов и введем ряд правил их комбинаторной конфигурации при формировании атомарных функциональных задач управления в архитектуре системы:

1. Вектор атомарной функциональной задачи управления в архитектуре интегрированной сложной системы может состоять из переменных следующих классов:

$$\varphi_n = \langle x^{lp} \vee x^{sp}, x^{lb} \vee x^{sb}, x^{lc} \vee x^{sc}, x^{ls} \vee x^{ss} \rangle, \quad (76)$$

где x^{lp} – класс локального представления, x^{sp} – класс серверного представления, x^{lb} – класс локальной бизнес-логики, x^{sb} – класс серверной бизнес-логики, x^{lc} – класс

локального вычисления, x^{sc} – класс серверного вычисления, x^{ls} – класс локального хранения, x^{ss} – класс серверного хранения.

2. Допускаются комбинации классов, при которых первый класс представления может быть любым, но следующие за ним три класса принадлежат строго одной типизации размещения (локальной или серверной):

$$\varphi_n = \begin{pmatrix} x^{lp}, x^{lb}, x^{lc}, x^{ls} \\ x^{sp}, x^{sb}, x^{sc}, x^{ss} \\ x^{sp}, x^{sb}, x^{sc}, x^{ss} \\ x^{sp}, x^{lb}, x^{lc}, x^{ls} \end{pmatrix}, \quad (77)$$

Выражение (77) формально представляет таблицу истинности (Таблица 17) комбинации классов, организующих атомарную функциональную задачу управления:

Таблица 17

Таблицы истинности комбинации классов, организующих атомарную функциональную задачу управления

φ_n	x^p		x^b		x^c		x^s	
	x^{lp}	x^{sp}	x^{lb}	x^{sb}	x^{lc}	x^{sc}	x^{ls}	x^{ss}
1	1	0	1	0	1	0	1	0
1	0	1	0	1	0	1	0	1
1	1	0	0	1	0	1	0	1
1	0	1	1	0	1	0	1	0

Для всех прочих комбинаций, пропущенных в Таблице 17, в рамках настоящего исследование значение истинности принимается равным нулю. При этом классы интеграционной функции всегда серверные.

3. Количество атомарных функциональных задач управления независимо от этапа жизненного цикла цифровой трансформации архитектуры конечно и детерминировано $N(\varphi_n) = const$.
4. Количество атомарных функциональных задач управления эквивалентно количеству классов вычислений во всей системе: $N(\varphi_n) = \alpha(x^c) = \alpha(x^{lc}) + \alpha(x^{sc})$; а количество классов представления, бизнес-логики и хранения определяется от количества агентов I до количества атомарных функциональных задач управления: $I \leq \alpha(x^p) \leq N, I \leq \alpha(x^b) \leq N, I \leq \alpha(x^s) \leq N$.

5. Классы представления, бизнес-логики и хранения одного и того же агента могут переиспользоваться и организовывать (входить в состав) до N атомарных функциональных задач управления, если они размещаются на одном физическом узле:

$$\Theta = \{(\{C_i^{3,j}\} \multimap \{\varphi_1, \dots, \varphi_n\}) \in g_1; (\{v_m\} \multimap \{\varphi_1, \dots, \varphi_n\}) \in g_2; (\{C_i^{3,j}\} \multimap \{v_m\}) \in g_3; (\{v_m\} \multimap \{C_i^{3,j}\}) \in g_3\}, \quad (78)$$

6. Реализация атомарных функциональных задач управления на физических узлах $x_\alpha|v_m$ осуществляется через размещение отдельных классов, организующих её архитектуру, так что уравнение (75) можно переписать в следующий вид:

$$\begin{aligned} A(\varphi_N) &= (\{\varphi_n\}, G(\varphi_N)) \\ \varphi_n &= \langle x^p|v_m, x^b|v_m, x^c|v_m, x^s|v_m, C_i^{3,j} \rangle, \\ G(\varphi_N) &= (\{x^p, x^b, x^c, x^s\}, \{y^{pb}, y^{bc}, y^{bs}\}) \end{aligned} \quad (79)$$

Допустим, что в архитектуре системы A необходимо реализовать две атомарные функциональные задачи управления φ_1, φ_2 одного агента управления $C_1^{3,j}$ на одном локальном физическом узле v_1^l . Тогда условия для конфигурирования архитектуры можно представить как:

$$F_{cnfg}(\Theta) = G(A), \quad (80)$$

Исходя из условий (80) и правил конфигурирования архитектура может быть представлена в различных вариантах, например:

При минимально возможном числе классов (Рисунок 15), организующих атомарные функциональные задачи управления:

$$\begin{aligned} A_{\alpha_{min}(x)} &= (\{\varphi_1, \varphi_2\}, G(A_{\alpha_{min}(x)})) \\ \varphi_1 &= \langle x_1^{lp}|v_1^l, x_1^{lb}|v_1^l, x_1^{lc}|v_1^l, x_1^{ls}|v_1^l, C_1^{3,j} \rangle, \\ \varphi_2 &= \langle x_1^{lp}|v_1^l, x_1^{lb}|v_1^l, x_2^{lc}|v_1^l, x_1^{ls}|v_1^l, C_1^{3,j} \rangle, \\ G(A_{\alpha_{min}(x)}) &= (\{x_1^{lp}, x_1^{lb}, x_1^{lc}, x_2^{lc}, x_1^{ls}\}, \{y_{11}^{lp|lb}, y_{11}^{lb|lc}, y_{12}^{lb|lc}, y_{11}^{lb|ls}\}) \end{aligned} \quad (81)$$

Граф логической конфигурации архитектуры для двух атомарных задач с минимальным числом классов представлен на Рисунке 15:

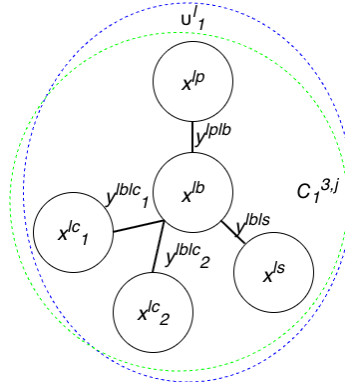


Рисунок 15. Граф логической конфигурации архитектуры для двух атомарных функциональных задач управления одного агента на одном физическом узле с минимальным числом классов

При максимально возможном числе классов (Рисунок 16), организующих атомарные функциональные задачи управления:

$$\begin{aligned}
 A_{\alpha_{max}(x)} &= (\{\varphi_1, \varphi_2\}, G(A_{\alpha_{max}(x)})) \\
 \varphi_1 &= \langle x_1^{lp} | u_1^l, x_1^{lb} | u_1^l, x_1^{lc} | u_1^l, x_1^{ls} | u_1^l, C_1^{3,j} \rangle, \\
 \varphi_2 &= \langle x_2^{lp} | u_1^l, x_2^{lb} | u_1^l, x_2^{lc} | u_1^l, x_2^{ls} | u_1^l, C_1^{3,j} \rangle, \\
 G(A_{\alpha_{max}(x)}) &= (\{x_1^{lp}, x_2^{lp}, x_1^{lb}, x_2^{lb}, x_1^{lc}, x_2^{lc}, x_1^{ls}, x_2^{ls}\}, \\
 &\quad \{y_{11}^{lp lb}, y_{22}^{lp lb}, y_{11}^{lb lc}, y_{22}^{lb lc}, y_{11}^{lb ls}, y_{22}^{lb ls}\})
 \end{aligned} \tag{82}$$

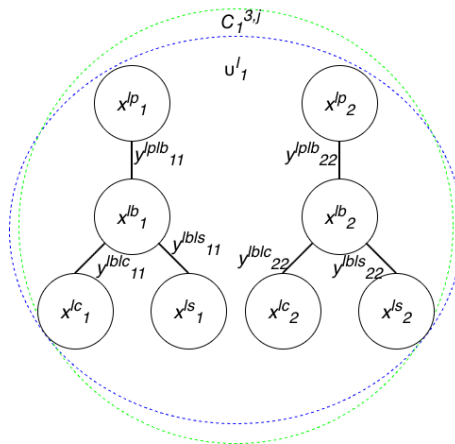


Рисунок 16. Граф логической конфигурации архитектуры для двух атомарных функциональных задач управления одного агента на одном физическом узле с максимальным числом классов

В случаях (81) и (82) любой из вариантов конфигурации архитектуры является допустимым, однако не ясно, какой из них является предпочтительным с учётом

удовлетворения требованию функциональной пригодности в обоих случаях. Более того, в условиях построения архитектуры автономно функционирующей кибер-физической пространственно-распределенной геотехнической индустриальной системы необходимо учитывать, что все атомарные функциональные задачи управления и, как следствие, агенты управления должны быть интегрированы в единую конфигурационную архитектуру предприятия $G(A^\phi)$. Данное условие может быть представлено в виде обязательного требования по дополнению к архитектуре независимой от агентов атомарной функциональной задачи управления интеграционным взаимодействием $\varphi^\phi = \langle x^{\phi p}, x^{\phi b}, x^{\phi c}, x^{\phi s} \rangle$, связываемой по интерфейсу $y^{\phi b}$ класс $x^{\phi p}$ с каждым существующим классом бизнес-логики.

Например, для случаев (81) и (82) графы таких интегрированных архитектур выглядели бы так, как показано на Рисунке 17.

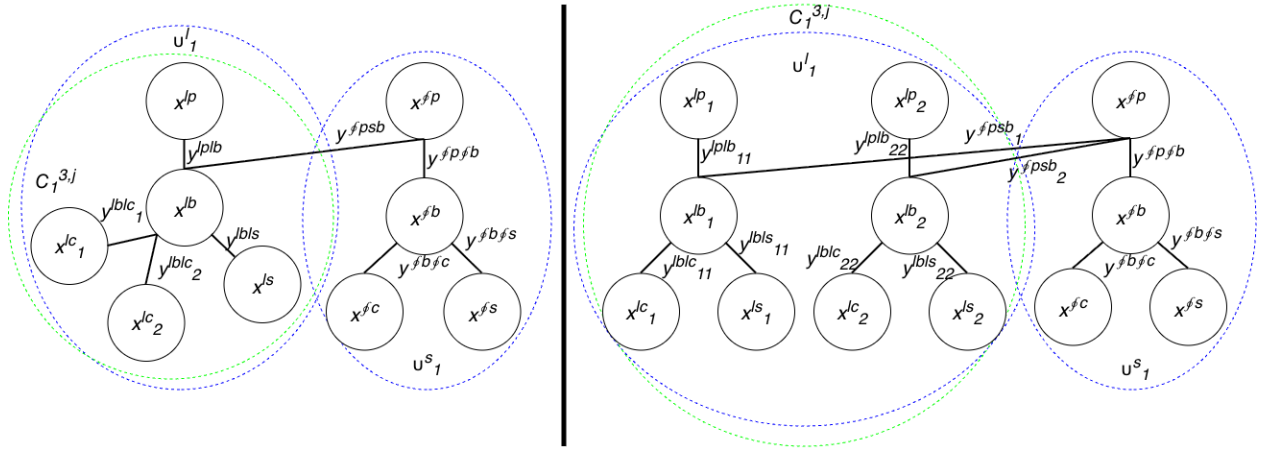


Рисунок 17. Графы логической конфигурации интегрированных архитектур для двух атомарных функциональных задач управления

А логические конфигурации таких архитектур имели бы следующий вид:

$$\begin{aligned}
 A_{\alpha_{min}(x)}^\phi &= \left(\{ \varphi_1, \varphi_2, \varphi_3^\phi \}, G(A_{\alpha_{min}(x)}^\phi) \right) \\
 \varphi_1 &= \langle x_1^{lp} | v_1^l, x_1^{lb} | v_1^l, x_1^{lc} | v_1^l, x_1^{ls} | v_1^l, C_1^{3,j} \rangle, \\
 \varphi_2 &= \langle x_1^{lp} | v_1^l, x_1^{lb} | v_1^l, x_2^{lc} | v_1^l, x_1^{ls} | v_1^l, C_1^{3,j} \rangle, \\
 \varphi_3^\phi &= \langle x_1^{\phi p} | v_1^s, x_1^{\phi b} | v_1^s, x_2^{\phi c} | v_1^s, x_1^{\phi s} | v_1^s \rangle, \\
 G(A_{\alpha_{min}(x)}^\phi) &= \left(\{ x_1^{\phi p}, x_1^{lp}, x_1^{\phi b}, x_1^{lb}, x_2^{\phi c}, x_1^{lc}, x_2^{lc}, x_1^{\phi s}, x_1^{ls} \}, \right. \\
 &\quad \left. \{ y_{11}^{\phi p \phi b}, y_{11}^{\phi plb}, y_{11}^{lp lb}, y_{11}^{\phi b \phi c}, y_{11}^{lb lc}, y_{12}^{lb lc}, y_{11}^{\phi b \phi s}, y_{11}^{lb ls} \} \right)
 \end{aligned} \tag{83}$$

$$\begin{aligned}
A_{\alpha_{max}(x)}^{\phi} &= \left(\left\{ \varphi_1, \varphi_2, \varphi_3^{\phi} \right\}, G \left(A_{\alpha_{max}(x)}^{\phi} \right) \right) \\
\varphi_1 &= \langle x_1^{lp} | v_1^l, x_1^{lb} | v_1^l, x_1^{lc} | v_1^l, x_1^{ls} | v_1^l, C_1^{3,j} \rangle, \\
\varphi_2 &= \langle x_2^{lp} | v_1^l, x_2^{lb} | v_1^l, x_2^{lc} | v_1^l, x_2^{ls} | v_1^l, C_1^{3,j} \rangle, \\
\varphi_3^{\phi} &= \langle x_1^{\phi p} | v_1^s, x_1^{\phi b} | v_1^s, x_2^{\phi c} | v_1^s, x_1^{\phi s} | v_1^s \rangle, \\
G \left(A_{\alpha_{max}(x)}^{\phi} \right) &= \left(\left\{ x_1^{\phi p}, x_1^{lp}, x_2^{lp}, x_1^{\phi b}, x_1^{lb}, x_2^{lb}, x_2^{\phi c}, x_1^{lc}, x_2^{lc}, x_1^{\phi s}, x_1^{ls}, x_2^{ls} \right\}, \right. \\
&\quad \left. \left\{ y_{11}^{\phi p \phi b}, y_{11}^{\phi plb}, y_{12}^{\phi plb}, y_{11}^{lp lb}, y_{22}^{lp lb}, y_{11}^{\phi b \phi c}, y_{11}^{lb lc}, y_{22}^{lb lc}, y_{11}^{\phi b \phi s}, y_{11}^{lb ls}, y_{22}^{lb ls} \right\} \right)
\end{aligned} \tag{84}$$

Очевидно, что в случае (84) архитектура системы в виде графа обладает большим числом уникальных структурных логических классов и интерфейсов, вследствие чего сам граф и, следовательно, архитектура обладает большей сложностью структурной конфигурации при равной функциональной пригодности.

В таком случае для оценки и сопоставления двух вариантов архитектур, заданных в виде графов, можно воспользоваться энтропийной оценкой суммарной сложности по степени вершин графа (3) так, что:

$$H(G(A)) = \sum_1^{\alpha} \log_2 \deg(x_{\alpha}), \tag{85}$$

где $\deg(x_{\alpha})$ – степень ноды, рассчитываемая по числу инцидентных рёбер (интерфейсов) и обуславливающая структурную сложность логического класса x_{α} , $H(G(A))$ – сложность логической конфигурации архитектуры.

В соответствии с (85) в Таблице 18 приведены оценки сложности для графов архитектур (75), (80), (81), (83) и (84):

Примеры оценок сложности различных архитектур

Уравнение	Архитектура	Кол-во задач	Кол-во классов	Кол-во интерфейсов	Сложность
(75)	$A(\varphi_n)$	1	4	3	1,585
(80)	$A_{\alpha_{min}(x)}$	2	5	4	2
(81)	$A_{\alpha_{max}(x)}$	2	8	6	3,17
(83)	$A^{\phi}_{\alpha_{min}(x)}$	2+1	8	8	4,4069
(84)	$A^{\phi}_{\alpha_{max}(x)}$	2+1	12	11	7,17

Исходя из полученных оценок можно сделать вполне очевидный вывод о монотонном возрастании сложности логической конфигурации архитектуры при увеличении числа классов и интерфейсов, организующих атомарные функциональные задачи управления. Таким образом предлагаемый критерий сложности логической конфигурации, в целом, позволяет определить оценку качества структурной организации архитектуры в статическом состоянии и сопоставлять такие оценки при разных вариантах реализации.

Однако, необходимо отметить, что предложенные примеры носят иллюстративный характер и не в полной мере отображают ряд особенностей конфигураций реальных архитектур сложных крупномасштабных систем, а также способы управления ими, в т.ч. в условиях цифровой трансформации. Для пространственно-распределенных геотехнических индустриальных систем исходная логическая конфигурация архитектуры детерминирована условиями организации технологических процессов с большим количеством различных (типовых) физических узлов и агентов управления. Так, например, атомарные функциональные задачи оперативного управления технологическим оборудованием не могут быть размещены (полностью) на серверном физическом узле и, в отдельных случаях, их реализация может дублироваться одновременно разными агентами управления – программными системами и персоналом. В том числе необходимо иметь в виду и тот факт, что действующие архитектуры предприятий, как правило, включают в себя наборы отдельных слабо связанных (или вообще несвязанных) специализированных автоматизированных систем с отдельно реализованными логико-программными конфигурациями, а сами агенты и физические узлы обладают экземплярами – $\theta: g_4, g_6$, что регламентирует техническую реализацию логической конфигурации архитектуры.

3.2 Разработка операций и правил модификации логической конфигурации архитектур

Как было отмечено в предыдущем разделе, предложенный способ оценки говорит лишь о состоянии некоторой статической конфигурации архитектур, зафиксированной во времени и неподверженной каким-либо модификациям в условиях развития системы и управления её жизненным циклом. Для определения же действительной «сложности» логической конфигурации при выборе того или иного вида архитектуры (например, модели «to be») необходимо учитывать не только статическую структуру, но и динамические изменения, вызванные, к примеру, требованиями цифровой трансформации.

В этой связи предлагается ввести следующие ключевые операции, которые могут быть совершены с логической конфигурацией архитектуры, по аналогии с операциями редактирования графов. К таким операциям модификации отнесем:

- добавление логического класса (ноды);
- удаление логического класса (ноды);
- слияние логических классов (нод);
- разделение логических классов (нод);
- обновление логического класса (ноды);
- добавление интерфейса (ребра);
- удаление интерфейса (ребра);
- слияние интерфейсов (рёбер);
- разделение интерфейсов (рёбер);
- обновление интерфейса (ребра);

Каждая операция модификации логической конфигурации обладает некоторой стоимостью в рамках управления жизненным циклом архитектуры, которую можно выразить в виде эвристической метрики – добавления петли y^{mod} к тем нодам, над которыми проводятся конкретные операции. При этом все операции модификаций интерфейсов (рёбер) не проявляются самостоятельно и являются следствием операций модификации классов (нод), что можно выразить следующими общими правилами:

1. Если добавляется какой-либо класс $x_{\alpha+1}$, то к такому классу необходимо добавить петлю y^{mod} и интерфейс $y_{\alpha+1,\beta}$, соединяющий его с соответствующим допустимым классом x_{β} . Далее для каждого из двух классов, к которым добавляется интерфейс $y_{\alpha+1,\beta}$, необходимо добавить петли y^{mod} (Рисунок 18):

$$\rho^{add}(X, Y, Y^{mod}) = (X \cup \{x'_{\alpha+1}\}, Y \cup \{y_{\alpha+1, \beta}\}, Y^{mod} \cup \{2y_{\alpha+1}^{mod}, y_{\beta}^{mod}\}), \quad (86)$$

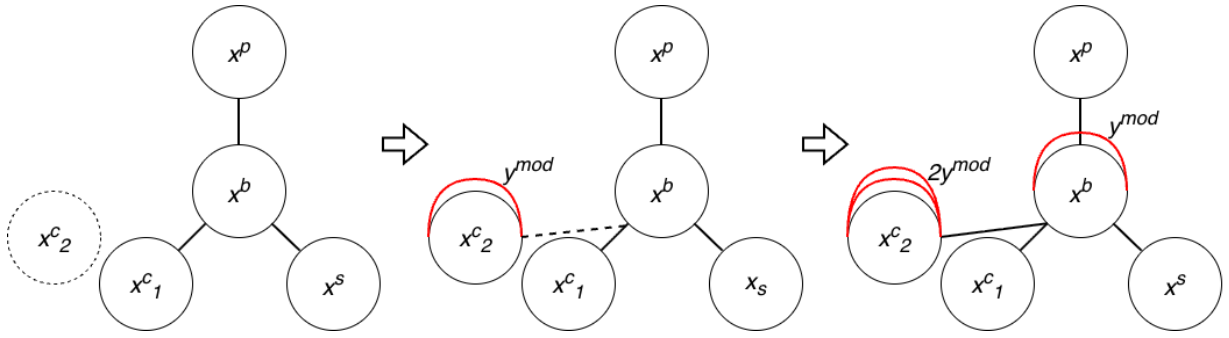


Рисунок 18. Схема операции добавления класса (x_2^c) к логической конфигурации архитектуры

2. Если удаляется какой-либо класс $x_{\alpha-1}$, то к такому классу необходимо добавить петлю y^{mod} и удалить все интерфейсы $y_{\alpha-1, \beta}$, соединяющие его с другими классами. Далее для каждого класса, у которых удаляется интерфейс $y_{\alpha-1, \beta}$, необходимо добавить петлю y^{mod} , а сами удаляемые класс и интерфейс(ы) удалить (Рисунок 19):

$$\rho^{dlt}(X, Y, Y^{mod}) = (X \setminus \{x_{\alpha-1}\}, Y \setminus \{y_{\alpha-1, \beta}\}, Y^{mod} \cup \{y_{\beta}^{mod}\}), \quad (87)$$

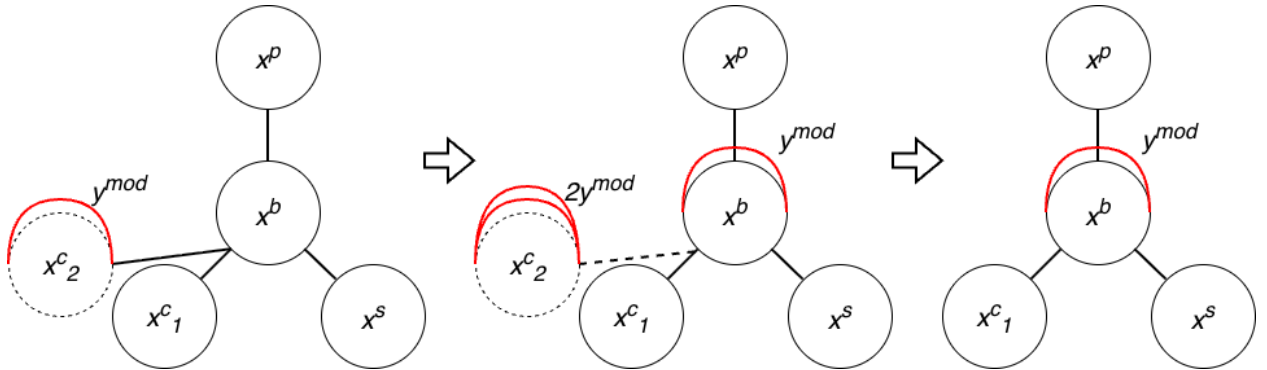


Рисунок 19. Схема операции удаления класса (x_2^c) из логической конфигурации архитектуры

3. Если два класса x_{α} и x_{β} сливаются в один класс, то к каждому из классов x_{α} и x_{β} необходимо добавить петли y^{mod} . Далее для каждого из классов, соединяющихся по интерфейсам с классами x_{α} и x_{β} необходимо добавить по одной петле y^{mod} за каждый интерфейс. После слияния классов x_{α} и x_{β} итоговому классу $x_{\alpha\beta}$ присвоить одну петлю (Рисунок 20).

$$\rho^{mrg}(X, Y, Y^{mod}) = ((X \setminus \{x_{\alpha}, x_{\beta}\}) \cup \{x'_{\alpha\beta}\}, (Y \setminus \{y_{\alpha\gamma}, y_{\beta\gamma}\}) \cup \{y_{\alpha\gamma}\}, Y^{mod} \cup \{y_{\alpha}^{mod}, 2y_{\gamma}^{mod}\}), \quad (88)$$

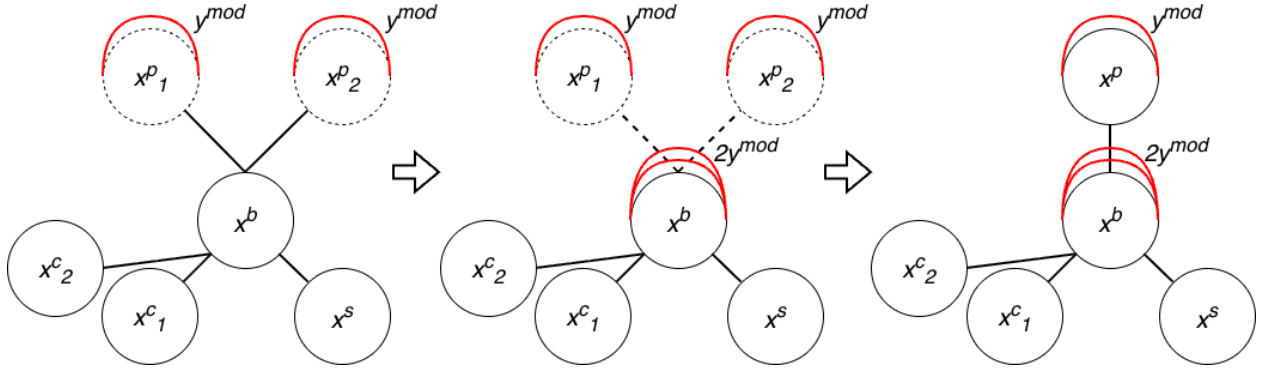


Рисунок 20. Схема операции слияния двух классов (x_1^p и x_2^p) в один класс (x^p) в логической конфигурации архитектуры

4. Если класс x_α разделяется на два класса x'_α и x'_β , то к такому классу необходимо добавить петлю y^{mod} . Далее для каждого из классов, соединяющихся по интерфейсам с классом x_α необходимо добавить по одной петле y^{mod} . После разделения класса x_α на классы x'_α и x'_β , каждому итоговому классу присвоить по одной петле (Рисунок 21):

$$\rho^{sprt}(X, Y, Y^{mod}) = ((X \setminus \{x_\alpha\}) \cup \{x'_\alpha, x'_\beta\}, (Y \setminus \{y_{\alpha\beta, \gamma}\}) \cup \{y_{\alpha\gamma}, y_{\beta\gamma}\}, Y^{mod} \cup \{y_\alpha^{mod}, y_\beta^{mod}, y_\gamma^{mod}\}), \quad (89)$$

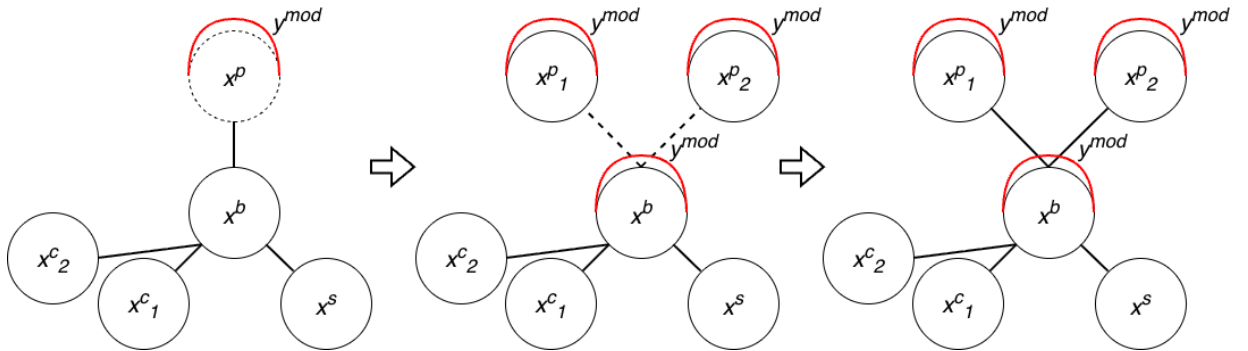


Рисунок 21. Схема операции разделения класса (x^p) на два класса (x_1^p и x_2^p) в логической конфигурации архитектуры

5. Если класс x_α обновляется, то к такому классу необходимо добавить петлю y^{mod} . Далее для всех классов, соединенных с классом x_α интерфейсами, также необходимо добавить по одной петле y^{mod} (Рисунок 22):

$$\rho^{updt}(X, Y, Y^{mod}) = (X, Y, Y^{mod} \cup \{y_\alpha^{mod}, y_\gamma^{mod}\}), \quad (90)$$

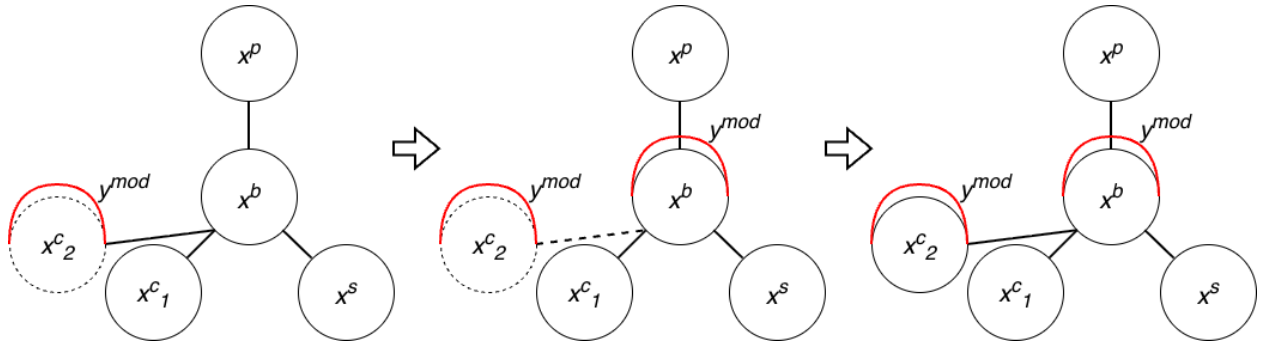


Рисунок 22. Схема операции обновления класса (x_2^c) в логической конфигурации архитектуры

Определим, что каждая петля y^{mod} повышает степень узла на единицу, тогда сложность процедур модификаций логической конфигурации архитектуры $H^{mod}(G'(A'))$ будет определяться как:

$$H^{mod}(G'(A')) = \sum_1^{\alpha} \log_2 \deg^{mod}(x'_\alpha), \quad (91)$$

а сложность самой модификации архитектуры определим как общую сложность получаемой структуры и процедур модификаций:

$$H'(G'(A')) = \sum_1^{\alpha} \log_2 \deg'(x'_\alpha), \quad (92)$$

где $\deg'(x'_\alpha) = \deg(x'_\alpha) + \deg^{mod}(x'_\alpha)$.

Используя уравнения (91) и (92) в Таблице 19 рассчитана сложность логической конфигурации до модификации $\deg(x_\alpha)$, сложность логической конфигурации после модификации $\deg(x'_\alpha)$, сложность проведения модификации логической конфигурации $\deg'(x'_\alpha)$ и сложность операций модификации $\deg^{mod}(x'_\alpha)$ архитектуры для приведенных примеров операций на Рисунках 18-21.

Оценки сложности операций модификации логической конфигурации архитектур

Операции	<i>H</i>			
	$deg(x_\alpha)$	$deg(x'_\alpha)$	$deg'(x'_\alpha)$	$deg^{mod}(x'_\alpha)$
Добавление класса	1,58496	2,0	3,90696	1,52102
Удаление класса	2,0	1,58496	2,0	0,5
Слияние классов	2,3219	2,0	3,085	1,1117
Разделение классов	2,0	2,3219	3,585	0,9308
Обновление класса	2,0	2,0	2,8219	0,71438

Полученные оценки в Таблице 19 не являются аксиоматическими константами для операций модификации, т.к. очевидно зависят от исходной сложности конфигурации классов в архитектуре, для которых они проводятся.

Дополним правила применения операций модификации с учётом правил структурной конфигурации архитектуры:

1. Если однотипные классы принадлежат одному и тому же агенту и размещаются (могут размещаться) на одном физическом узле, то для таких классов допустимо применение операций слияния ρ^{mrg} , иначе слияние не допустимо:

$$\left\{ \begin{array}{l} x_\alpha \wedge x_\beta \rightarrow C_i^{3,j} \\ x_\alpha \wedge x_\beta \rightarrow v_m^{lvs} \\ x_\alpha \wedge x_\beta \in \{p \vee b \vee s\} \end{array} \right. , \quad (93)$$

2. Если необходимо реорганизовать архитектуру так, чтобы какая-либо атомарная функциональная задача управления, реализуемая одним агентом, стала реализовываться другим, то необходимо удалить соответствующие классы у первого агента и добавить необходимые классы и интерфейсы другому:

$$\rho^{mp} \left((\varphi_n \rightarrow C_i^{3,j}) \rightarrow (\varphi'_n \rightarrow C_i^{3,j}) \right) = p^{dlt} \wedge p^{add}, \quad (94)$$

Таким образом уравнения (81)-(89) представляют собой основу процедур конфигурации и модификации архитектур, которые будут использованы далее при разработке алгоритмов.

3.3 Разработка критериев и механизмов решения задач цифровой трансформации архитектур

Для реализации алгоритмов оценки и анализа архитектур необходимо ввести ряд дополнительных формальных условий их конфигурации и модификации. В соответствии с алгоритмом формирования метамодели (Рисунок 13) первоначально может быть получена метамодель «as is» θ^0 , на основании которой должна быть сконфигурирована архитектура $F_{cnfg}(\theta^0)$. Тогда, с учётом правил модификации, составим следующие условия процедуры конфигурации архитектуры для состояния «as is»:

1. Все атомарные функциональные задачи управления φ_n , реализуемые агентами управления из числа «Персонала» $C_i^{3,j=1}$, должны быть представлены собственными отдельными логическими классами, неподлежащими слиянию p^{mrg} .
2. Все атомарные функциональные задачи управления φ_n , реализуемые программными агентами из числа из числа «Персонала» $C_i^{3,j=1}$, не могут иметь связи с атомарной функциональной задачей управления интеграционным взаимодействием φ^ϕ .

$$\begin{cases} p^{mrg} = \emptyset \mid \{\varphi_n\} \rightarrow \{C_i^{3,j=1}\} \\ y^{\phi pb} = \emptyset \mid \{\varphi_n\} \rightarrow \{C_i^{3,j=1}\} \end{cases} \quad (95)$$

В таком случае для полученного графа логической конфигурации архитектуры «as is» $G^0(A^0)$ может быть рассчитана оценка его структурной сложности $H^0(G^0(A^0))$, рассчитываемая по формуле (85).

Далее, в соответствии с алгоритмом (Рисунок 13) и общей моделью Жизненного Цикла цифровой трансформации архитектуры, необходимо получить некоторую модель «to be» (Рисунок 10), представляющую из себя вариант модификации логической конфигурации архитектуры по заданному критерию оптимальности. В случае построения плана цифровой трансформации под моделью «to be» подразумевается логическая конфигурация архитектуры $G^*(A^*)$, для которой формулируются следующие условия $F_{cnfg}(\theta^*)$:

1. Все атомарные функциональные задачи управления φ_n , реализуемые агентами управления из числа «Персонала» $C_i^{3,j=1}$, должны быть реализованы агентами управления из числа «Программные системы» $C_i^{3,j=0}$.

2. Замена агентов управления на «Программные системы» должна быть реализована с учётом правила ρ^{rmp} (94), при этом допускается как добавление новых агентов $C_{i+1}^{3,j=0}$, так и переназначение атомарных функциональных задач управления существующим.
3. Все атомарные функциональные задачи управления φ_n должны быть интегрированы с атомарной функциональной задачей управления интеграционным взаимодействием φ^Φ .
4. Каталог атомарных функциональных задач управления Φ , каталог физических узлов размещения Υ и их граф связи g_2 не должны изменяться.
5. Допускаются любые процедуры модификации архитектуры, не противоречащие заданным условиям и правилам модификаций.

$$\left\{ \begin{array}{l} C_i^{3,j=1} = \emptyset \\ \rho^{rmp} \mid \{\varphi_n\} \rightarrow \{C_i^{3,j=0}, C_{i+1}^{3,j=0}\} \\ \rho^{add} \mid \varphi^\Phi \\ y^{\Phi pb} \mid \{\varphi_n\} \rightarrow \{C_i^{3,j=0}, C_{i+1}^{3,j=0}\} \\ (\Phi, \Upsilon, g_2)^0 \Leftrightarrow (\Phi, \Upsilon, g_2)^* \end{array} \right., \quad (96)$$

В соответствии с условиями уравнения (91) может быть получена логическая конфигурация и метамодель архитектуры в состоянии «to be», удовлетворяющая критериям управления цифровой трансформации промышленных систем. Используя предложенные критерии и механизмы оценки качества архитектур могут быть получены количественные измерения состояний «as is» и различные варианты состояний «to be» в целях сопоставления и выбора оптимальной модификации (например, экспертно-аналитическим путём).

Однако используя известную формулу расстояния редактирования графа, можно записать следующее формальное условие оптимизации логической конфигурации архитектуры:

$$F_{opt}(A^*): GED \left(G(A), F_{cnfg}(\Theta^*) \right) = \min_{\rho^{mod} \in \mathcal{P}(G(A), G^*(A^*))} H^*(G^*(A^*)), \quad (97)$$

где $\rho^{mod} = \{\rho^{add}, \rho^{del}, \rho^{mrg}, \rho^{sprt}, \rho^{updt}, \rho^{rmp}\}$ – множество применяемых операций модификации, $\mathcal{P}(G(A), G^*(A^*))$ – набор путей модификации логической конфигурации архитектуры.

Таким образом в качестве ключевого критерия решения задачи по планированию оптимальной логической конфигурации архитектуры автономно функционирующей кибер-физической пространственно-распределенной геотехнической индустриальной системы, декларируется минимизация сложности модификации, учитывающей как итоговую статическую структуру, так и сложность самих процедур её получения, что обуславливает эвристическую оценку стоимости жизненного цикла цифровой трансформации.

Следует отметить, что данный критерий может быть рассмотрен в виде отдельных составляющих компонент (как в Таблице 19), так и расширен в виде условия поддержки и обновления агентов, а именно:

Вследствие недетерминированности во времени и дискретности фактических этапов модификации архитектуры в виде замены агентов управления, а также неминуемом развитии технологических стеков, применяемых при разработке программных систем, при планировании модели «to be» следует учитывать следующие условия $DT(A^*)$:

1. Каждая атомарная функциональная задача управления может быть полностью заменена на новую реализацию за весь жизненный цикл архитектуры.
2. Каждая атомарная функциональная задача управления должна хотя бы раз полностью обновлена за весь жизненный цикл архитектуры.

$$\begin{cases} p^{dlt} \vee p^{add} \\ p^{updt} \end{cases}, \quad (98)$$

В таком случае критерий оптимизации $F_{opt}(A^*)$ может быть дополнен оценкой сложности полного цикла цифровой трансформации $H^{DT}(G^*(A^*))$, учитывающей добавление петель y^{mod} за реализацию условий $DT(A^*)$.

3.4 Разработка алгоритмов конфигурирования и анализа архитектур

3.4.1 Алгоритм процесса конфигурирования и анализа архитектуры «as is»

На Рисунке 23 представлена блок-схема алгоритма процесса конфигурирования и анализа архитектуры в состоянии «as is» на основе результатов разделов 3.1-3.3.

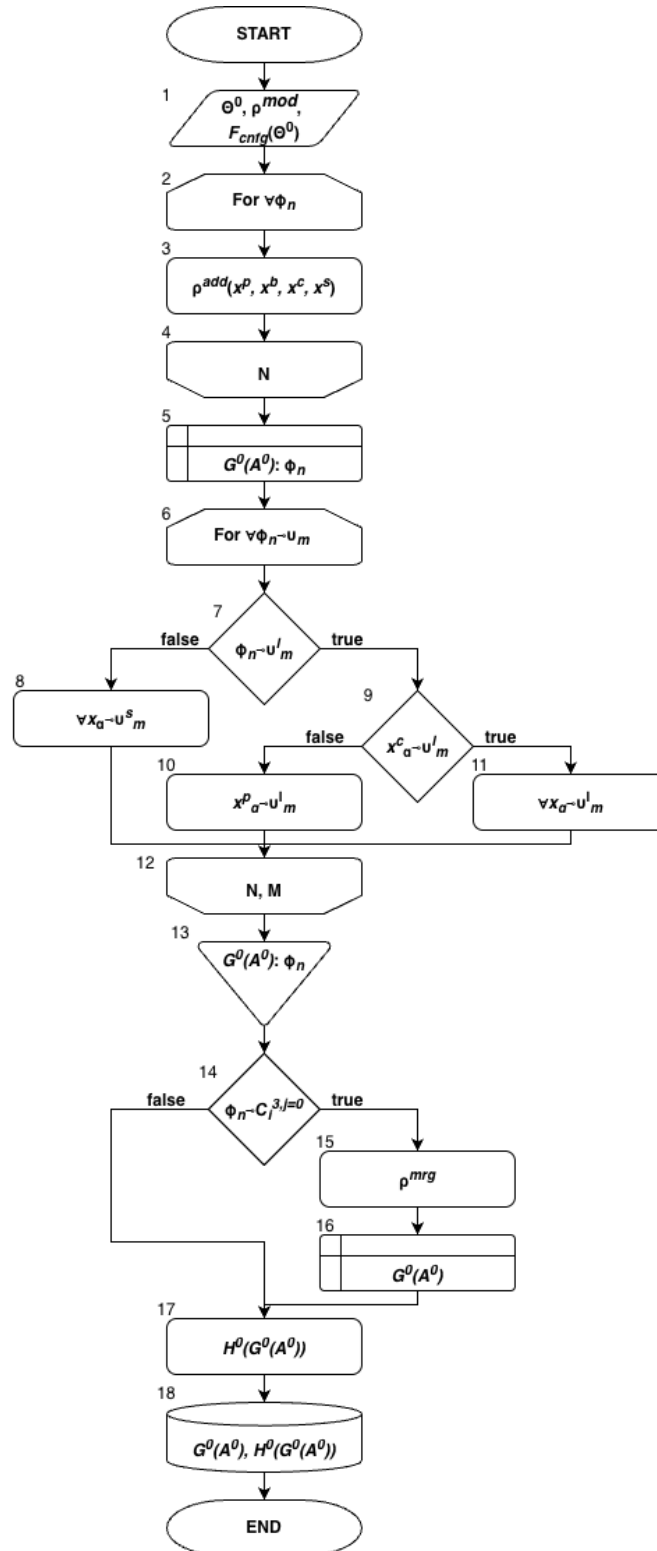


Рисунок 23. Блок-схема алгоритма процесса конфигурирования и анализа архитектуры «as is»

Описание блок-схемы алгоритма

START

Начало работы алгоритма.

Блок 1. Ввод данных

Загрузка из репозитория актуальной метамодели архитектуры «as is» θ^0 , правил конфигурации и модификации ρ^{mod} и условий процедуры конфигурации архитектуры «as is» $F_{cnfg}(\theta^0)$.

Блок 2. Начало цикла

Начало цикла создания логических классов x^p, x^b, x^c, x^s для всех атомарных функциональных задач управления φ_n .

Блок 3. Процесс

Процедура создания логических классов x^p, x^b, x^c, x^s и интерфейсов y^{pb}, y^{bc}, y^{bs} в соответствии с правилом добавления класса ρ^{add} . Индексирование классов и формирование матрицы смежности.

Блок 4. Конец цикла

Завершение цикла создания логических классов.

Блок 5. Внутреннее хранилище

Формирование первичного графа архитектуры $G^0(A^0)$ и сохранение результата в оперативную память.

Блок 6. Начало цикла

Начало цикла размещения логических классов x_α атомарных функциональных задач управления φ_n на физических узлах v_m по графу g_2 .

Блок 7. Условие

Если атомарная функциональная задача φ_n размещается на локальном физическом узле v_m^l , то переход к Блоку 9, иначе переход к Блоку 8.

Блок 8. Процесс

Процедура размещения всех логических классов x^p, x^b, x^c, x^s атомарной функциональной задачи управления φ_n на серверном физическом узле v_m^s .

Блок 9. Условие

Если атомарная функциональная задача управления φ_n относится к оперативному управлению с размещением на локальном физическом узле v_m^l , то переход к Блоку 11, иначе переход к Блоку 10.

Блок 10. Процесс

Процедура размещения логического класса представления x^p на локальном физическом узле v_m^l и размещения логических классов x^b, x^c, x^s на серверном физическом узле v_m^s .

Блок 11. Процесс

Процедура размещения всех логических классов x^p, x^b, x^c, x^s на локальном физическом узле v_m^l .

Блок 12. Конец цикла

Завершение цикла размещения логических классов на физических узлах.

Блок 13. Слияние

Процедура слияния результатов размещения логических классов и обновления матрицы смежности для графа архитектуры $G^0(A^0)$.

Блок 14. Условие

Если атомарная функциональная задача управления реализуется агентом управления из числа «Программных систем» $C_i^{3,j=0}$, то переход к Блоку 15, иначе переход к Блоку 17.

Блок 15. Процесс

Процедура слияния ρ^{mrq} и реиндексация однотипных логических классов, принадлежащих одному агенту и размещаемых на одном физическом узле.

Блок 16. Внутреннее хранилище

Обновление матрицы смежности графа архитектуры $G^0(A^0)$.

Блок 17. Процесс

Процедура вычисления и оценки структурной сложности полученной логической конфигурации архитектуры «as is» $H^0(G^0(A^0))$.

Блок 18. Сохранение результата

Сохранение полученной матрицы смежности и вычисленной оценки структурной сложности графа логической конфигурации архитектуры «as is».

END

Завершение работы алгоритма.

3.4.2 Алгоритм процесса конфигурирования и анализа архитектуры «to be»

На Рисунке 24 представлена блок-схема алгоритма процесса конфигурирования и анализа архитектуры в состоянии «to be» на основе результатов разделов 3.1-3.3

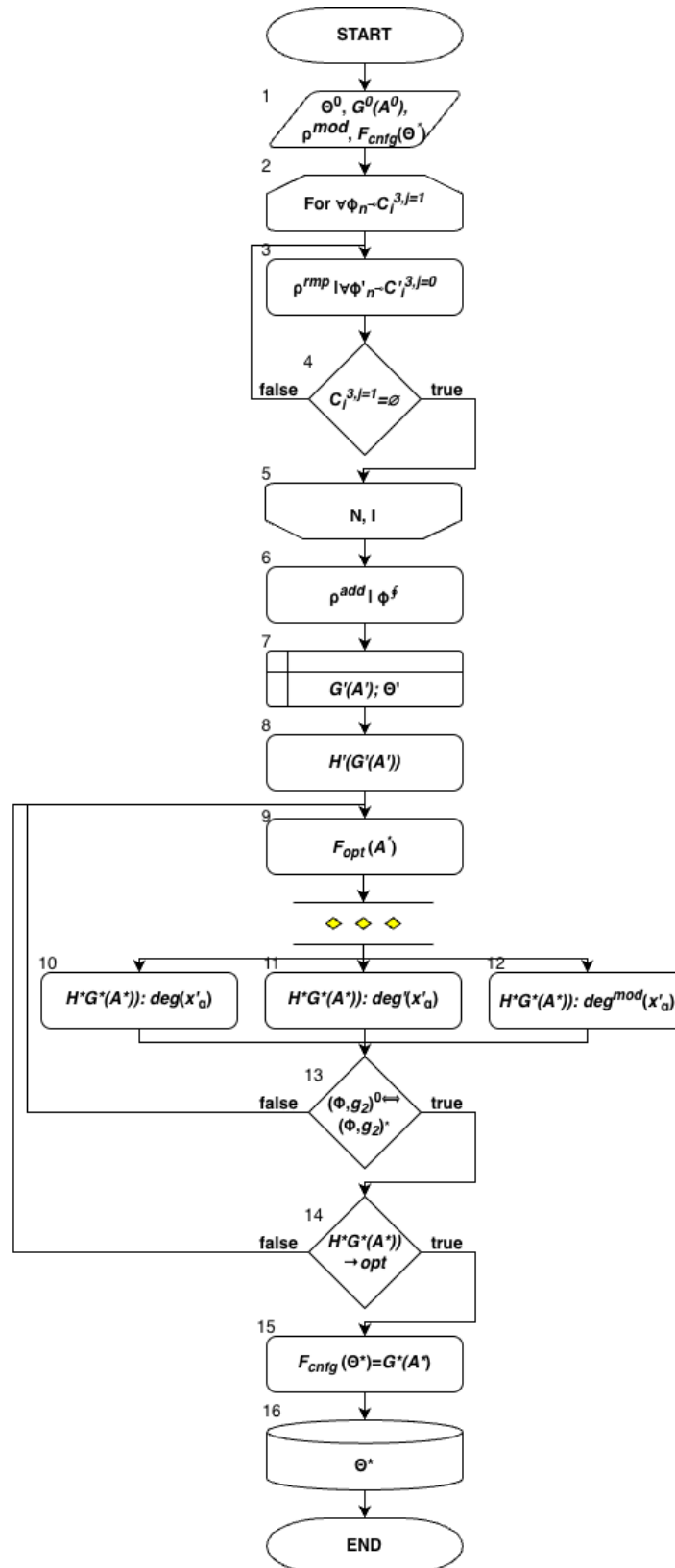


Рисунок 24. Блок-схема алгоритма процесса конфигурирования и анализа архитектуры «to be»

Описание блок-схемы алгоритма

START

Начало работы алгоритма.

Блок 1. Ввод данных

Загрузка из репозитория актуальных метамодели архитектуры «as is» θ^0 , матрицы смежности графа логической конфигурации архитектуры $G^0(A^0)$, правил конфигурации и модификации ρ^{mod} и условий процедуры конфигурации архитектуры «to be» $F_{cnfg}(\theta^*)$.

Блок 2. Начало цикла

Начало цикла перераспределения всех атомарных функциональных задач управления φ_n от агентов «Персонал» $C_i^{3,j=1}$ новым агентам управления «Программные системы» $C_i^{3,j=0}$.

Блок 3. Процесс

Процедура создания новых программных агентов $C_i^{3,j=0}$ в отношении 1:1 к агентам «Персонал» $C_i^{3,j=1}$ с их последующим удалением и перераспределения атомарных функциональных задач управления φ_n в соответствии с правилом перераспределения ρ^{rmp} .

Блок 4. Условие

Если больше не существует агентов управления из числа «Персонал» $C_i^{3,j=1}$, то переход к Блоку 5, иначе возврат к Блоку 3.

Блок 5. Конец цикла

Завершение цикла перераспределения атомарных функциональных задач управления φ_n от агентов «Персонал» $C_i^{3,j=1}$.

Блок 6. Процесс

Процедура создания атомарной функциональной задачи управления интеграционным взаимодействием φ^f и добавление интерфейса y^{fpb} между классом представления x^{fp} и классом бизнес-логики x^b для всех существующих атомарных функциональных задач управления φ_n в соответствии с правилом ρ^{add} .

Блок 7. Внутреннее хранилище

Формирование матрицы смежности модифицированного графа логической конфигурации $G'(A')$ и метамодели θ' архитектуры с сохранением в оперативную память.

Блок 8. Процесс

Процедура вычисления и оценки структурной сложности полученной логической конфигурации модифицированной архитектуры $H'(G'(A'))$.

Блок 9. Процесс

Запуск процедуры оптимизации логической конфигурации модифицированной архитектуры в виде поиска путей модификации графа при различных вариантах оценки сложности.

Параллелизация алгоритма

Параллельный запуск процедур оптимизации логической конфигурации для разных вариантов критерия оптимальности (сложности).

Блок 10. Процесс

Процедура вычисления и оценки структурной сложности оптимизированной логической конфигурации архитектуры $H^*(G^*(A^*))$: учёт степени нод только по числу интерфейсов между классами – $deg(x'_\alpha)$.

Блок 11. Процесс

Процедура вычисления и оценки суммарной структурной сложности и сложности модификации оптимизированной логической конфигурации архитектуры $H^*(G^*(A^*))$: учёт степени нод по числу интерфейсов между классами и числу петель процедур модификации – $deg'(x'_\alpha)$.

Блок 12. Процесс

Процедура вычисления и оценки сложности модификации оптимизированной логической конфигурации архитектуры $H^*(G^*(A^*))$: учёт степени нод только по числу петель процедур модификации – $deg^{mod}(x'_\alpha)$.

Блок 13. Условие

Если каталоги атомарных функциональных задач управления Φ^0, Φ^* и графы связи g_2^0, g_2^* для исходной метамодели архитектуры Θ^0 и метамодели, полученной при оптимизации, Θ^* эквиваленты, то переход к Блоку 14, иначе возврат к Блоку 9.

Блок 14. Условие

Если достигнута оптимальная сложность структурной модификации логической конфигурации архитектуры $H^*(G^*(A^*))$, то переход к Блоку 15, иначе возврат к Блоку 9.

Блок 15. Процесс

Процедура обратной конфигурации метамодели Θ^* по матрице смежности графа логической конфигурации оптимальной архитектуры $G^*(A^*)$.

Блок 16. Сохранение результата

Создание метамодели архитектуры «to be» Θ^* и сохранение результатов в репозитории.

END

Завершение работы алгоритма.

3.5 Выводы по Главе 3

В третьей главе разработаны математические модели представления и оценки логической конфигурации архитектуры, формальные операции и правила её модификации, критерии и механизмы решения задач цифровой трансформации, а также алгоритмы конфигурации и анализа архитектур в состояниях «as is» и «to be», обеспечивающие возможность количественного сопоставления различных вариантов архитектурных решений.

В разделе 3.1 предложена формализация модели представления атомарной функциональной задачи управления в виде вектора из четырёх логических классов: представления, бизнес-логики, вычисления и хранения, организуемых по аналогии с классами слоев архитектуры программных систем и унифицирующая вид задач инвариантно типу агентов управления. Показано, что совокупность векторов для всех атомарных функциональных задач управления образует матрицу смежности, формирующую граф логической конфигурации архитектуры $G(A)$, где нодами являются логические классы, а рёбра интерфейсы связи между ними. Обоснована звёздная топология интерфейсов с центральной ролью класса бизнес-логики, определяющая три типа рёбер: представление–бизнес-логика, бизнес-логика–вычисление и бизнес-логика–хранение. Введены правила размещения классов на физических узлах, регламентирующие распределение между локальными и серверными компонентами в зависимости от исходного состояния архитектуры, описываемого метамоделью.

Введен и обоснован универсальный критерий качества архитектур систем в виде оценки сложности логической конфигурации на базе функции от суммы логарифмов степеней вершин графа по основанию два. Предложенный критерий обеспечивает инвариантную количественную оценку качества архитектуры в статическом состоянии, допускающую сопоставление различных типов архитектурных решений.

В разделе 3.2 разработан набор из пяти базовых операций модификации логической конфигурации архитектуры, формализованных по аналогии с операциями редактирования графов: добавление, удаление, слияние, разделение и обновление класса. Для каждой операции определена формальная процедура преобразования матрицы смежности графа с указанием изменений множества вершин, рёбер и петель модификации. Введена эвристическая метрика стоимости модификации в виде петель на диагонали матрицы смежности, каждая из которых увеличивает степень соответствующей вершины на единицу, что позволяет учитывать стоимость проведённых операций в рамках единого критерия сложности. Сформулированы отдельные критерии качества архитектур в виде

оценок сложности процедур модификации и суммарной сложности модифицированной конфигурации, объединяющие статическую структуру графа и стоимость операций модификации. Определены строгие правила допустимости операций модификации с учётом исходной структурной конфигурации архитектуры.

В разделе 3.3 сформулированы условия конфигурирования архитектуры для состояний «as is» и «to be» в контексте цифровой трансформации промышленных систем. Для состояния «as is» определено условие, регламентирующее представление атомарных функциональных задач управления, реализуемых персоналом, в виде отдельных не интегрированных логических классов без возможности слияния. Для состояния «to be» определено формальное условие, требующее замены всех агентов из числа персонала на программные агенты с перераспределением атомарных функциональных задач управления, обязательной общей со-интеграцией и сохранением инвариантности каталогов атомарных функциональных задач управления и физических узлов.

Предложен формальный критерий оптимизации логической конфигурации архитектуры, основанный на минимизации сложности модификации по множеству допустимых путей редактирования графа, удовлетворяющих условию, определяющему допустимость путей как соответствие условиям конфигурации и правилам модификации. Введено условие полного цикла цифровой трансформации, требующее, чтобы каждая атомарная функциональная задача управления была хотя бы один раз полностью заменена или обновлена за весь жизненный цикл архитектуры, что обуславливает дополнение критерия оптимизации оценкой сложности полного цикла цифровой трансформации.

В соответствии с предложенными формальными моделями и процедурами в разделе 3.4 разработаны алгоритмы конфигурации и анализа архитектур для состояний «as is» и «to be». В подразделе 3.4.1 предложен алгоритм конфигурации архитектуры «as is» (Рисунок 23), реализующий последовательное формирование логических классов для всех атомарных функциональных задач управления, их размещение на физических узлах с учётом правил локального и серверного размещения, слияние однотипных классов программных агентов и вычисление структурной сложности полученной конфигурации. В подразделе 3.4.2 предложен алгоритм конфигурации архитектуры «to be» (Рисунок 24), реализующий перераспределение атомарных функциональных задач от агентов-персонала к программным агентам, создание интеграционных связей, формирование модифицированного графа логической конфигурации и запуск процедуры оценки сложности по трём вариантам критерия оптимальности: структурной сложности по числу интерфейсов, суммарной сложности с учётом петель модификации и сложности только процедур модификации. Предложенные алгоритмы обеспечивают формальную основу для

экспериментального моделирования и количественного анализа архитектур в целях принятия решений по управлению их жизненным циклом в условиях цифровой трансформации.

ГЛАВА 4. ЭКСПЕРИМЕНТАЛЬНОЕ МОДЕЛИРОВАНИЕ И АНАЛИЗ РЕЗУЛЬТАТОВ

В рамках экспериментального моделирования в настоящем исследовании поставлен следующий ряд задач:

1. В соответствии с разработанными структурами онтологий общего и специального назначений, приведённых в разделах 2.3.1 и 2.3.2, осуществить их экспериментальное моделирование в инструментальной среде, поддерживающей встроенные механизмы доказательства консистентности и когерентности аксиом для определения истинности и непротиворечивости заложенных знаний в метамодель архитектуры автономно функционирующей кибер-физической пространственно-распределенной геотехнической индустриальной системы.
2. Осуществить программную реализацию разработанных процедур механизмов логико-семантического вывода (раздел 2.4.1) и алгоритма формирования метамодели архитектуры (раздел 2.4.2) в целом для получения каталога атомарных функциональных задач управления, каталога физических узлов размещения и графа связи, а также определить полноту результирующего вывода.
3. По результатам формирования априорных компонентов, экспертно-аналитическим путём осуществить доработку метамодели в части введения каталогов агентов управления и графов их связи для получения экспериментального примера в целях реализации алгоритмов конфигурирования и анализа архитектур в различных состояниях жизненного цикла.
4. Осуществить программную реализацию алгоритмов конфигурирования и анализа архитектур «as is» (раздел 3.4.1) и «to be» (раздел 3.4.2) с использованием экспериментальных примеров в целях валидации и верификации разработанных моделей и алгоритмов оценки и анализа архитектур для решения задач цифровой трансформации и создания автономно функционирующих кибер-физических пространственно-распределенных геотехнических индустриальных систем.

4.1 Результаты моделирования работы алгоритма построения метамодели архитектуры

4.1.1 Результаты онтологического моделирования

Реализация онтологической модели архитектуры автономно функционирующей кибер-физической пространственно-распределенной геотехнической индустриальной системы осуществлялась на языке OWL в программной среде Protégé ver. 5.6.3.

Полный набор концептов и полученной иерархии онтологии общего назначения представлен на Рисунке 25.

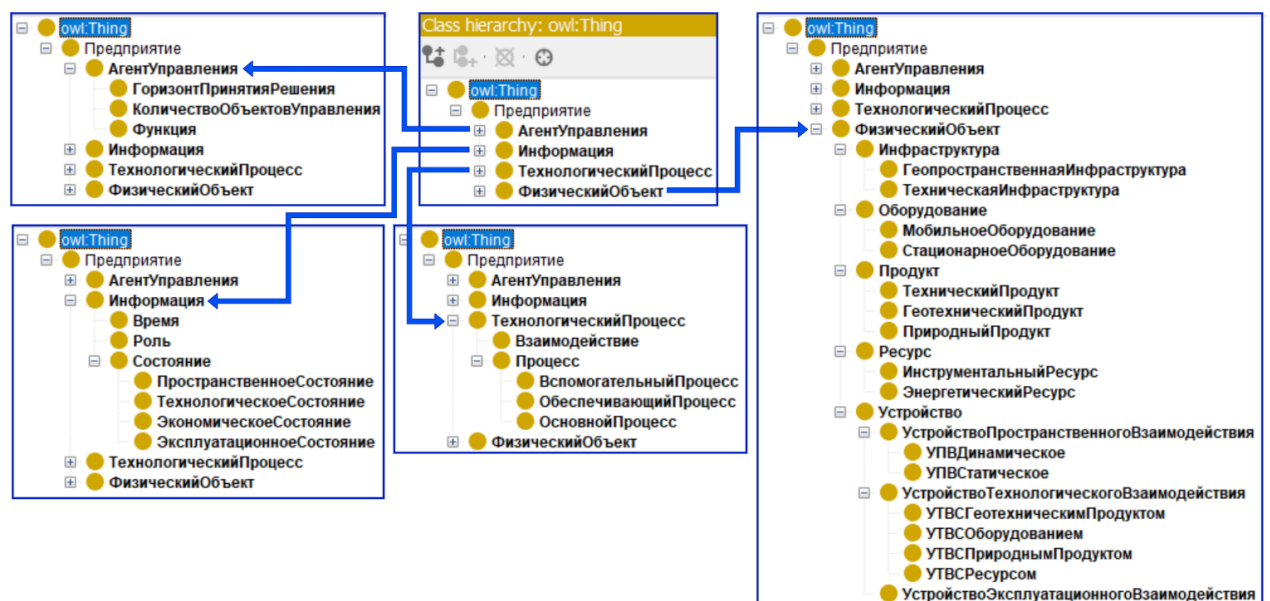


Рисунок 25. Иерархия классов концептов (понятий) онтологической модели архитектуры предприятия

На Рисунке 26 представлены фрагменты таблиц формирования ролей между концептами онтологии общего назначения, индивидов онтологии общего назначения и иерархия данных – наборы физико-математических и эконометрических величин, описывающих индивидов концепта «Состояния».

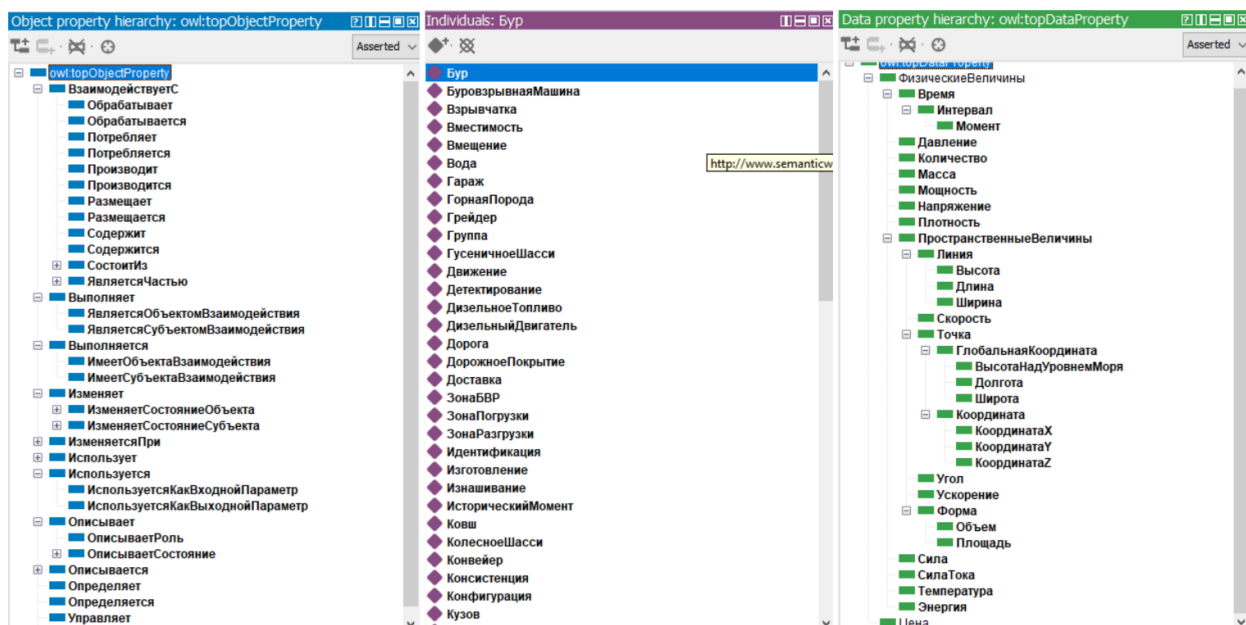


Рисунок 26. Фрагменты таблицы иерархии отношений между классами концептов (Object property hierarchy, синий цвет) и их экземплярами, таблицы экземпляров (Individuals, фиолетовый цвет) и таблицы иерархии данных (Data property hierarchy, зеленый цвет)

На Рисунке 27 представлен пример присвоения концептам иерархии наборов априорных аксиоматических ролей в виде предикативных отношений с другими концептами и индивидами, определяющими логико-семантическую конструкцию модели.

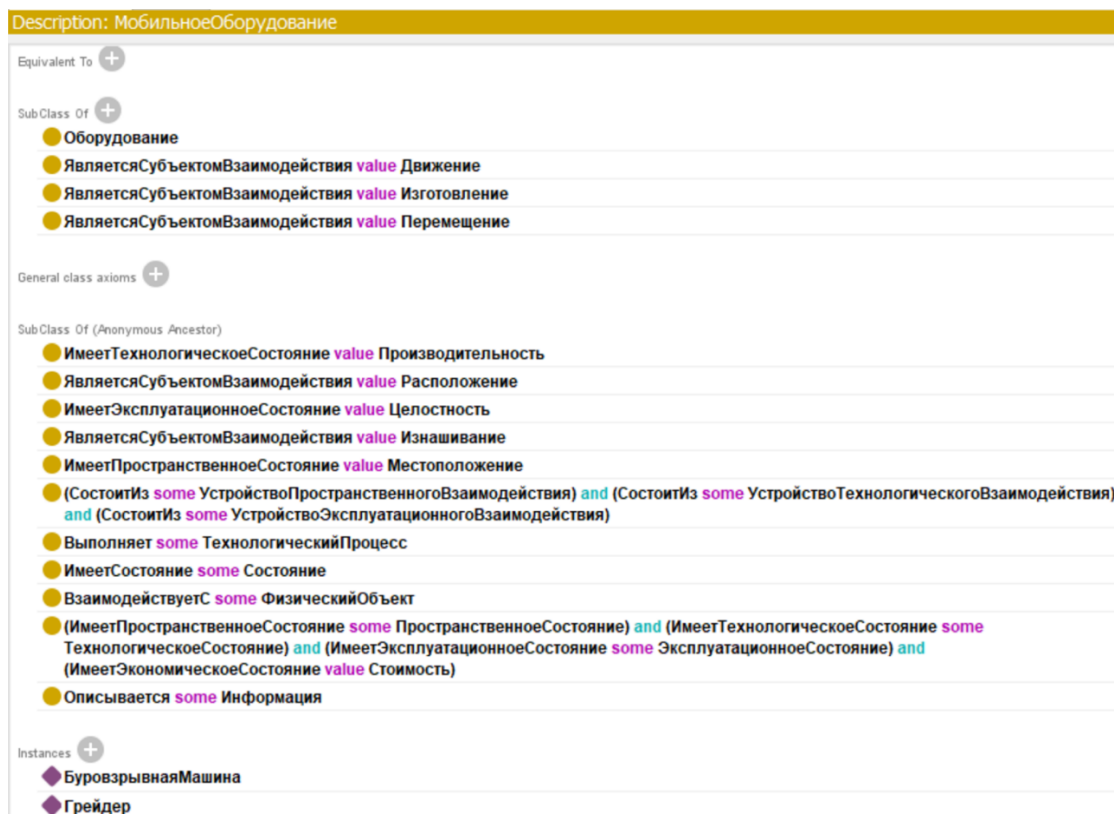


Рисунок 27. Пример формирования ролей в качестве априорных логико-семантических аксиом

Diagram illustrating the structure of an Enterprise (Предприятие) as a complex system, showing relationships between various components:

- Технологический Процесс** (Technological Process)
- Информация** (Information)
- Предприятие** (Enterprise)
- Физический Объект** (Physical Object)
- Агент Управления** (Control Agent)

The diagram shows numerous dashed lines with arrows representing relationships and flows between these components. A red box highlights a specific relationship:

Информация -- Описывает (Subclass some) --> Физический Объект

124

Для подтверждения консистентности и когерентности полученной онтологической модели был использован один из наиболее традиционных и развитых рассуждателей дескрипционной логики – HermiT (ver. 1.4.3.456). Доказательство истинности и непротиворечивости аксиом онтологической модели представлено на Рисунке 30.

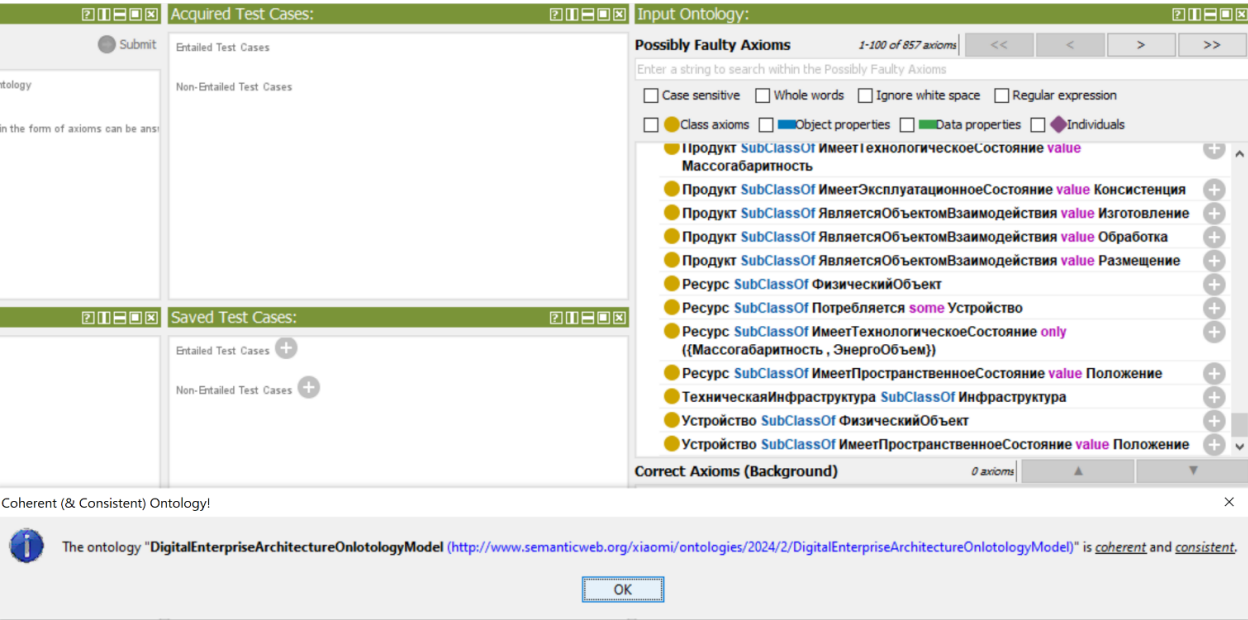


Рисунок 30. Подтверждение когерентности и консистентности онтологии с использованием HermiT

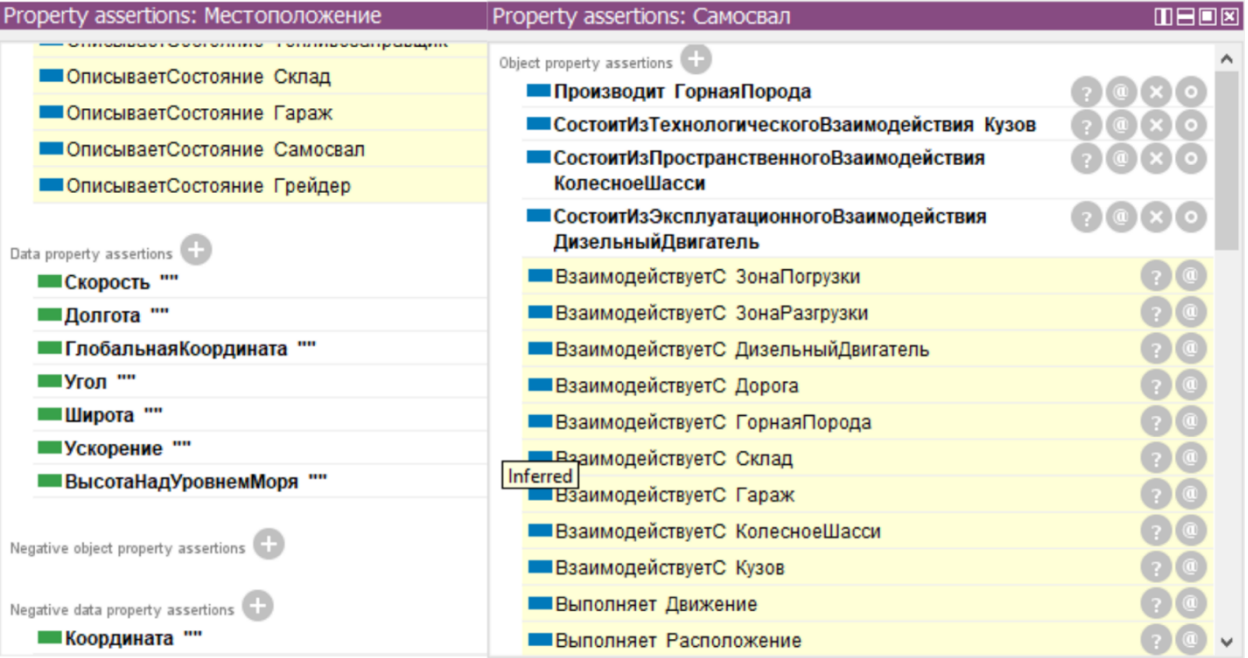
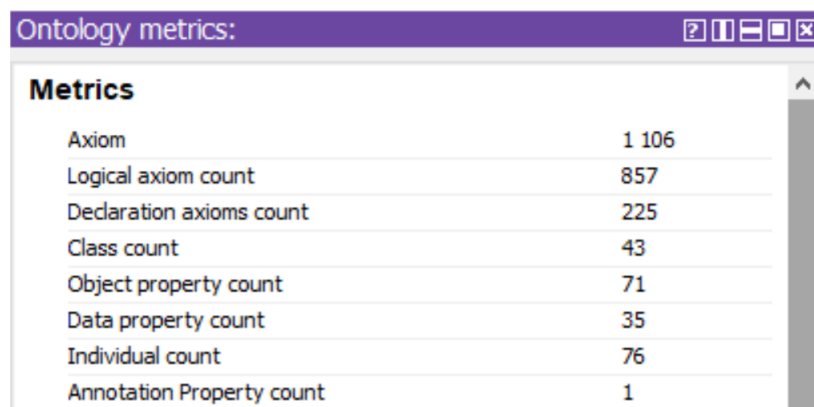


Рисунок 31. Визуализация результаты работы HermiT для вывода скрытых аксиом

На Рисунке 31 представлена визуализация результата работы HermiT по выводу скрытых (апостериорных) аксиоматических связей индивидов, не задекларированных «в ручном» режиме на стадии формирования онтологии.



Ontology metrics:	
Metrics	
Axiom	1 106
Logical axiom count	857
Declaration axioms count	225
Class count	43
Object property count	71
Data property count	35
Individual count	76
Annotation Property count	1

Рисунок 32. Метрики онтологической модели

На Рисунке 32 приведены метрические оценки количества аксиоматических высказываний (в т.ч. априорно задекларированных, и логически выведенных) онтологической модели.

Таким образом была сформирована онтологическая модель архитектуры автономно функционирующей кибер-физической геопространственной индустриальной системы на примере горнопромышленного предприятия по открытой добычи полезного ископаемого. Необходимо отметить, что в онтологию не введено исчерпывающее множество индивидов, являющихся базовыми компонентами архитектуры структурно-функциональной организации предприятий данного типа. Данная ситуация объясняется тем, что предлагаемая онтология направлена на демонстрацию способов унификации и стандартизации модельного представления архитектур, а следовательно, для обеспечения исчерпывающих глубины, полноты и единообразия таксономии, необходимо участие широкого круга отраслевых экспертов. В связи с этим текущую модель предлагается рассматривать как базовую основу для проведения дальнейших экспериментальных работ в рамках настоящего исследования и иллюстративный пример формирования комплексной методики цифровой трансформации индустриальных систем.

4.1.2 Результаты моделирования работы логико-семантического вывода

В рамках экспериментального моделирования предложенный механизм логико-семантического вывода реализован на языке Python 3.2 с использованием библиотек xml.etree.ElementTree, Pandas, Matplotlib, Pathlib.

Программная реализация механизма логико-семантического вывода осуществляет:

- Загрузку OWL-файла онтологии с расширением *.xml (Рисунок 33) из установленного локального хранилища.
- Считывание и парсинг структуры *.xml-файла по иерархии концептов и ролей с присвоением индексов.
- Перебор допустимых комбинаций в соответствии с заданной логико-семантической моделью.
- Создание в локальном хранилище файла с расширением *.csv по структуре Таблиц 15-16.
- Выгрузку в файл таблицы-каталога атомарных функциональных задач управления, таблицы структурно-функциональной организации атомарных функциональных задач управления, таблицы-каталога физических узлов, таблицы-матрицы смежности графа связи атомарных функциональных задач управления и физических узлов.

Рисунок 33. Фрагмент OWL-файла онтологии с расширением *.xml

Каталог представляет из себя перечень уникальных наименований, семантически различимых для отраслевых экспертов и разработчиков программного обеспечения, что, в целом, способствует удобству восприятия информации при анализе архитектур систем.

1	Ф	Наименование	ГПР	КО	Функция	Взаимодействие	Субъект	Объект	Вх. состояния суб.	Вх. состояния об.	Вх. Время	Вых. роль	Вых. время
2	ф1	Оперативное групповое детектирование со	1	2	1	5	610001	410001	2003;3001;4001	1002;2002	2	1	2
3	ф2	Оперативное групповое детектирование со	1	2	1	5	610001	410001	2003;3001;4001	1002;2002	2	2	2
4	ф3	Оперативное индивидуальное детектирова	1	1	1	5	610001	410001	2003;3001;4001	1002;2002	2	1	2
5	ф4	Оперативное индивидуальное детектирова	1	1	1	5	610001	410001	2003;3001;4001	1002;2002	2	2	2
6	ф5	Стратегическое групповое детектирование	3	2	1	5	610001	410001	2003;3001;4001	1002;2002	2	1	2
7	ф6	Стратегическое групповое детектирование	3	2	1	5	610001	410001	2003;3001;4001	1002;2002	2	2	2
8	ф7	Стратегическое индивидуальное детектиро	3	1	1	5	610001	410001	2003;3001;4001	1002;2002	2	1	2
9	ф8	Стратегическое индивидуальное детектиро	3	1	1	5	610001	410001	2003;3001;4001	1002;2002	2	2	2
10	ф9	Тактическое групповое детектирование сос	2	2	1	5	610001	410001	2003;3001;4001	1002;2002	2	1	2
11	ф10	Тактическое групповое детектирование сос	2	2	1	5	610001	410001	2003;3001;4001	1002;2002	2	2	2
12	ф11	Тактическое индивидуальное детектирован	2	1	1	5	610001	410001	2003;3001;4001	1002;2002	2	1	2
13	ф12	Тактическое индивидуальное детектирован	2	1	1	5	610001	410001	2003;3001;4001	1002;2002	2	2	2
14	ф13	Оперативное групповое идентификация сос	1	2	2	5	610001	410001	2003;3001;4001	1002;2002	4;2	1	2
15	ф14	Оперативное групповое идентификация сос	1	2	2	5	610001	410001	2003;3001;4001	1002;2002	4;2	2	2
16	ф15	Оперативное индивидуальное идентифика	1	1	2	5	610001	410001	2003;3001;4001	1002;2002	4;2	1	2
17	ф16	Оперативное индивидуальное идентифика	1	1	2	5	610001	410001	2003;3001;4001	1002;2002	4;2	2	2
18	ф17	Стратегическое групповое идентификация	3	2	2	5	610001	410001	2003;3001;4001	1002;2002	4;2	1	2
19	ф18	Стратегическое групповое идентификация	3	2	2	5	610001	410001	2003;3001;4001	1002;2002	4;2	2	2
20	ф19	Стратегическое индивидуальное идентифи	3	1	2	5	610001	410001	2003;3001;4001	1002;2002	4;2	1	2
21	ф20	Стратегическое индивидуальное идентифи	3	1	2	5	610001	410001	2003;3001;4001	1002;2002	4;2	2	2
22	ф21	Тактическое групповое идентификация сос	2	2	2	5	610001	410001	2003;3001;4001	1002;2002	4;2	1	2
23	ф22	Тактическое групповое идентификация сос	2	2	2	5	610001	410001	2003;3001;4001	1002;2002	4;2	2	2
24	ф23	Тактическое индивидуальное идентификац	2	1	2	5	610001	410001	2003;3001;4001	1002;2002	4;2	1	2
25	ф24	Тактическое индивидуальное идентификац	2	1	2	5	610001	410001	2003;3001;4001	1002;2002	4;2	2	2
26	ф25	Оперативное групповое планирование сос	1	2	4	5	610001	410001	2003;3001;4001	1002;2002	3;2	1	4
27	ф26	Оперативное групповое планирование сос	1	2	4	5	610001	410001	2003;3001;4001	1002;2002	3;2	2	4
28	ф27	Оперативное индивидуальное планирован	1	1	4	5	610001	410001	2003;3001;4001	1002;2002	3;2	1	4
29	ф28	Оперативное индивидуальное планирован	1	1	4	5	610001	410001	2003;3001;4001	1002;2002	3;2	2	4
30	ф29	Стратегическое групповое планирование	3	2	4	5	610001	410001	2003;3001;4001	1002;2002	3;2	1	4
31	ф30	Стратегическое групповое планирование	3	2	4	5	610001	410001	2003;3001;4001	1002;2002	3;2	2	4

Рисунок 36. Фрагмент выведенной структурно-функциональной организации атомарных функциональных задач управления

На Рисунке 36 представлен скрин-шот фрагмента выведенной структурно-функциональной организации атомарных функциональных задач управления, характеризующей их логическую различимость и предоставляющей вычислительную программную основу в целях обеспечения возможностей по самоорганизации и саморегуляции автономно функционирующих кибер-физических пространственно-распределенных геотехнических промышленных систем.

1	Индекс	Наименование
2	000001	Сервер
3	110001	Самосвал
4	110002	Экскаватор
5	110003	Погрузчик
6	110004	Грейдер
7	110005	Буровзрывная машина
8	110006	Топливозаправщик
9	120001	Конвейер

Рисунок 37. Каталог физических узлов размещения

На Рисунке 37 представлен скрин-шот каталога физических узлов размещения в виде проиндексированного перечня всех введенных в онтологическую модель индивидов физических объектов класса «Оборудование» и централизованного серверного физического узла. Всего получено 8 физических узлов: 1 серверный, и 7 локальных, из которых 6 относятся к мобильному оборудованию и 1 к стационарному.

В соответствии со структурно-функциональной организацией атомарных функциональных задач управления и правила логико-семантического вывода, для полученных каталогов сформирована матрица смежности графа связи, фрагмент которой представлен на Рисунке 38.

1	φ	000001	110001	110002	110003	110004	110005	110006	120001
2	φ1	1	0	0	0	0	0	0	0
3	φ2	1	0	0	0	0	0	0	0
4	φ3	1	0	0	0	0	0	0	0
5	φ4	1	0	0	0	0	0	0	0
6	φ5	1	0	0	0	0	0	0	0
7	φ6	1	0	0	0	0	0	0	0
8	φ7	1	0	0	0	0	0	0	0
9	φ8	1	0	0	0	0	0	0	0
10	φ9	1	0	0	0	0	0	0	0
11	φ10	1	0	0	0	0	0	0	0
12	φ11	1	0	0	0	0	0	0	0
13	φ12	1	0	0	0	0	0	0	0
14	φ13	1	0	0	0	0	0	0	0
15	φ14	1	0	0	0	0	0	0	0
16	φ15	1	0	0	0	0	0	0	0
17	φ16	1	0	0	0	0	0	0	0
18	φ17	1	0	0	0	0	0	0	0
19	φ18	1	0	0	0	0	0	0	0
20	φ19	1	0	0	0	0	0	0	0
21	φ20	1	0	0	0	0	0	0	0
22	φ21	1	0	0	0	0	0	0	0
23	φ22	1	0	0	0	0	0	0	0
24	φ23	1	0	0	0	0	0	0	0
25	φ24	1	0	0	0	0	0	0	0
26	φ25	1	0	0	0	0	0	0	0
27	φ26	1	0	0	0	0	0	0	0
28	φ27	1	0	0	0	0	0	0	0
29	φ28	1	0	0	0	0	0	0	0
30	φ29	1	0	0	0	0	0	0	0
31	φ30	1	0	0	0	0	0	0	0

Рисунок 38. Матрица смежности графа связи выведенных каталогов

На основе полученной матрицы построен двудольный граф связи атомарных функциональных задач управления и физических узлов размещения, представленный на Рисунке 39.

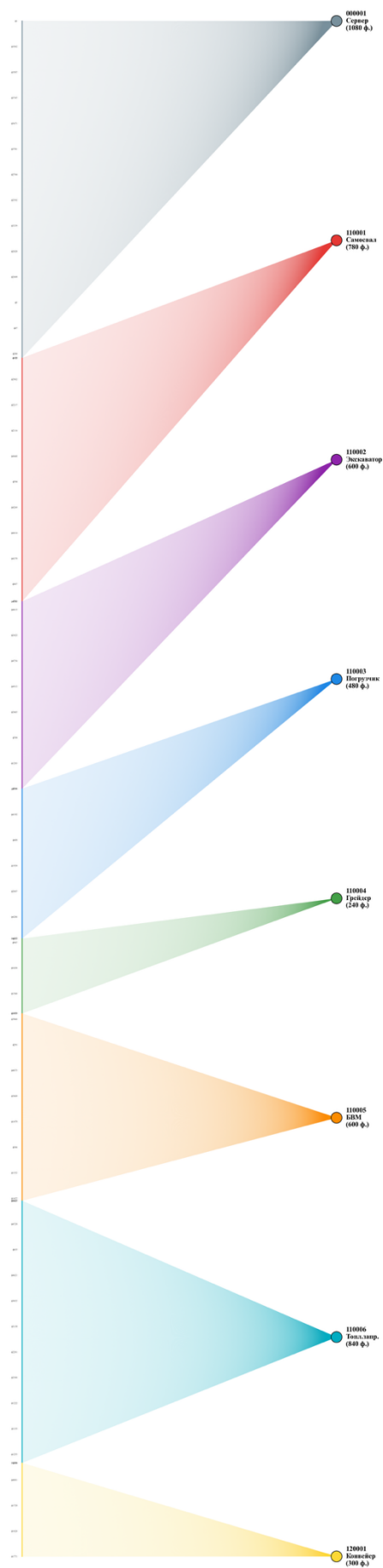


Рисунок 39. Граф связи атомарных функциональных задач управления и физических узлов размещения

Определим качество полученного решения в части полноты и точности логико-семантического вывода. Исходя из итоговых метрических оценок заданной онтологической модели (Рисунок 32), а также с учётом предложенных формальных определений понятию атомарной функциональной задачи управления, можно дать следующие теоретические оценки предельной степени полноты логико-семантического вывода:

Теоретический максимум «слепого» перебора всех возможных комбинаций, задекларированных в виде непротиворечивых логических аксиом без учёта ограничений предикативных ролей и семантического содержания можно оценить как:

$$n_{max}(\Phi) = \epsilon^2(E^2) * \epsilon(E^{1,1}) * \epsilon(E^{3,1}) * \epsilon(E^{3,2}) * \epsilon(E^{3,3}) * \epsilon(E^{4,3}) = 634\,260, \quad (99)$$

где n_{max} – максимальное число атомарных функциональных задач управления Φ , $\epsilon^2(E^2)$ – количество индивидов концепта «Физический объект», $\epsilon(E^{1,1})$ – количество индивидов концепта «Взаимодействие», $\epsilon(E^{3,1})$ – количество индивидов концепта «Функция», $\epsilon(E^{3,2})$ – количество индивидов концепта «Горизонт принятия решения», $\epsilon(E^{3,3})$ – количество индивидов концепта «Количество объектов», $\epsilon(E^{4,3})$ – количество индивидов концепта «Роль».

Однако, полный слепой перебор приводит к семантически бессмысленным конструкциям, противоречащим заданным ролевым отношениям, регламентирующих структурно-функциональную организацию горнопромышленной системы. С учётом аксиоматических ограничений в виде недопустимости взаимодействий для отдельных индивидов физических объектов, полный и семантически точный теоретический предел числа атомарных функциональных задач управления должен составить:

$$n_{max}^*(\Phi) = n_{max}(F_{Rol}: \Phi) = 4\,920, \quad (100)$$

Таким образом можно заключить, что итоговое число выведенных атомарных функциональных задач управления соответствует теоретической оценке их предельного числа, а следовательно механизм логико-семантического вывода удовлетворяет критериям точности и полноты.

4.2 Результаты моделирования работы алгоритмов конфигурирования и анализа архитектур

4.2.1 Результаты моделирования работы алгоритма конфигурирования и анализа архитектуры «as is»

Для дальнейшего экспериментального моделирования в части работы алгоритмов конфигурирования и анализа архитектур в соответствии с типовым ландшафтом предприятий по открытой добычи полезных ископаемых (Рисунок 11) экспертно-аналитическим путём был задан каталог агентов управления, включая набор типовых программных систем и должностей (Рисунок 40).

Индекс	Наименование	Тип
0001	АСУ ГТК	Программн
0002	ГГИС	Программн
0003	ERP	Программн
1001	Водитель самосвала	Персонал
1002	Водитель топливозаправщика	Персонал
1003	Оператор экскаватора	Персонал
1004	Оператор погрузчика	Персонал
1005	Оператор грейдера	Персонал
1006	Оператор БВМ	Персонал
1007	Оператор конвейера	Персонал
1008	Диспетчер	Персонал
1009	Маркшейдер	Персонал
1010	Геолог	Персонал
1011	Главный инженер	Персонал

Рисунок 40. Экспериментальный каталог агентов управления

Для полученного каталога агентов управления было задано экспериментальное распределение с целью формирования графов связи с атомарными функциональными задачами управления и физическими узлами размещения с учётом следующих экспертно-аналитических эвристик:

1. Для задач оперативного детектирования состояний физических объектов, в которых субъектом взаимодействия выступает индивид концепта «Оборудование» или «Устройство», назначить агента «АСУ ГТК»
2. Для всех остальных задач оперативного горизонта принятия решения, в которых субъектом взаимодействия выступает индивид концепта «Оборудование» или

- «Устройство», назначить агентами управления тех, чье наименование соответствует наименованию такого оборудования.
3. Для задач тактического управления с субъектом «Оборудование» назначить агентом управления «Диспетчер».
 4. Для задач стратегического управления с субъектом «Оборудование» назначить агентом управления «Главный инженер».
 5. Для всех остальных задач с субъектом «Оборудование» назначить агентом АСУ «ГТК».
 6. Для задач оперативного ГПП с субъектом «Геотехническая инфраструктура» и выходной ролью «Субъект» назначить агентом управления «Маркшейдер».
 7. Для задач оперативного ГПП с субъектом «Инфраструктура», объектом «Природный продукт» и выходной ролью «Объект» назначить агентом управления «Геолог».
 8. Для задач стратегического управления с субъектом «Инфраструктура» назначить агентом управления «Главный инженер».
 9. Для всех остальных задач с субъектом «Геотехническая инфраструктура», объектом «Продукт» назначить агента «ГГИС»
 10. Для задач стратегического планирования с объектом «Ресурс» назначить агента «ERP».
 11. Для всех задач с субъектом «Оборудование» или «Устройство» (принадлежащее конкретному оборудованию) назначить физическим узлом такое оборудование.
 12. Для всех остальных задач назначить физическим узлом «Сервер».

Сформулированные эвристики были реализованы в виде набора программных процедур на языке Python 3.2, обеспечивающих построение необходимого начального распределения каталогов и графов связи, формирующих метамодель архитектуры автономно функционирующей кибер-физической индустриальной системы.

На Рисунке 41 и Рисунке 42 показаны скриншоты фрагментов матриц смежности для графов связи каталогов атомарных функциональных задач управления – агентов управления и агентов управления – физических узлов размещения.

1	Ф	0001	0002	0003	1001	1002	1003	1004	1005	1006	1007	1008	1009	1010	1011
2	Ф1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
3	Ф2	1	0	0	0	0	0	0	0	0	0	0	0	0	0
4	Ф3	1	0	0	0	0	0	0	0	0	0	0	0	0	0
5	Ф4	1	0	0	0	0	0	0	0	0	0	0	0	0	0
6	Ф5	1	0	0	0	0	0	0	0	0	0	0	0	0	0
7	Ф6	1	0	0	0	0	0	0	0	0	0	0	0	0	0
8	Ф7	1	0	0	0	0	0	0	0	0	0	0	0	0	0
9	Ф8	1	0	0	0	0	0	0	0	0	0	0	0	0	0
10	Ф9	1	0	0	0	0	0	0	0	0	0	0	0	0	0
11	Ф10	1	0	0	0	0	0	0	0	0	0	0	0	0	0
12	Ф11	1	0	0	0	0	0	0	0	0	0	0	0	0	0
13	Ф12	1	0	0	0	0	0	0	0	0	0	0	0	0	0
14	Ф13	0	0	0	0	0	1	0	0	0	0	0	0	0	0
15	Ф14	0	0	0	0	0	1	0	0	0	0	0	0	0	0
16	Ф15	0	0	0	0	0	1	0	0	0	0	0	0	0	0
17	Ф16	0	0	0	0	0	1	0	0	0	0	0	0	0	0
18	Ф17	1	0	0	0	0	0	0	0	0	0	0	0	0	0
19	Ф18	1	0	0	0	0	0	0	0	0	0	0	0	0	0
20	Ф19	1	0	0	0	0	0	0	0	0	0	0	0	0	0
21	Ф20	1	0	0	0	0	0	0	0	0	0	0	0	0	0
22	Ф21	1	0	0	0	0	0	0	0	0	0	0	0	0	0
23	Ф22	1	0	0	0	0	0	0	0	0	0	0	0	0	0
24	Ф23	1	0	0	0	0	0	0	0	0	0	0	0	0	0
25	Ф24	1	0	0	0	0	0	0	0	0	0	0	0	0	0
26	Ф25	0	0	0	0	0	1	0	0	0	0	0	0	0	0
27	Ф26	0	0	0	0	0	1	0	0	0	0	0	0	0	0
28	Ф27	0	0	0	0	0	1	0	0	0	0	0	0	0	0
29	Ф28	0	0	0	0	0	1	0	0	0	0	0	0	0	0
30	Ф29	1	0	0	0	0	0	0	0	0	0	0	0	0	0
31	Ф30	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Рисунок 41. Матрица смежности графа связи каталога атомарных функциональных задач управления и каталога агентов управления

1	Агент управления	000001	110001	110002	110003	110004	110005	110006	120001
2	0001	0	1	1	1	1	1	1	1
3	0002	1	0	0	0	0	0	0	0
4	0003	1	0	1	0	0	0	1	0
5	1001	0	1	0	0	0	0	0	0
6	1002	0	0	0	0	0	0	1	0
7	1003	0	0	1	0	0	0	0	0
8	1004	0	0	0	1	0	0	0	0
9	1005	0	0	0	0	1	0	0	0
10	1006	0	0	0	0	0	1	0	0
11	1007	0	0	0	0	0	0	0	1
12	1008	0	1	1	1	1	1	1	1
13	1009	1	0	0	0	0	0	0	0
14	1010	1	0	0	0	0	0	0	0
15	1011	1	1	1	1	1	1	1	1

Рисунок 42. Матрица смежности графа связи каталога агентов управления и каталога физических узлов размещения

На Рисунке 43 и Рисунке 44 показаны графы связи на основе полученных матриц смежности начального распределения.

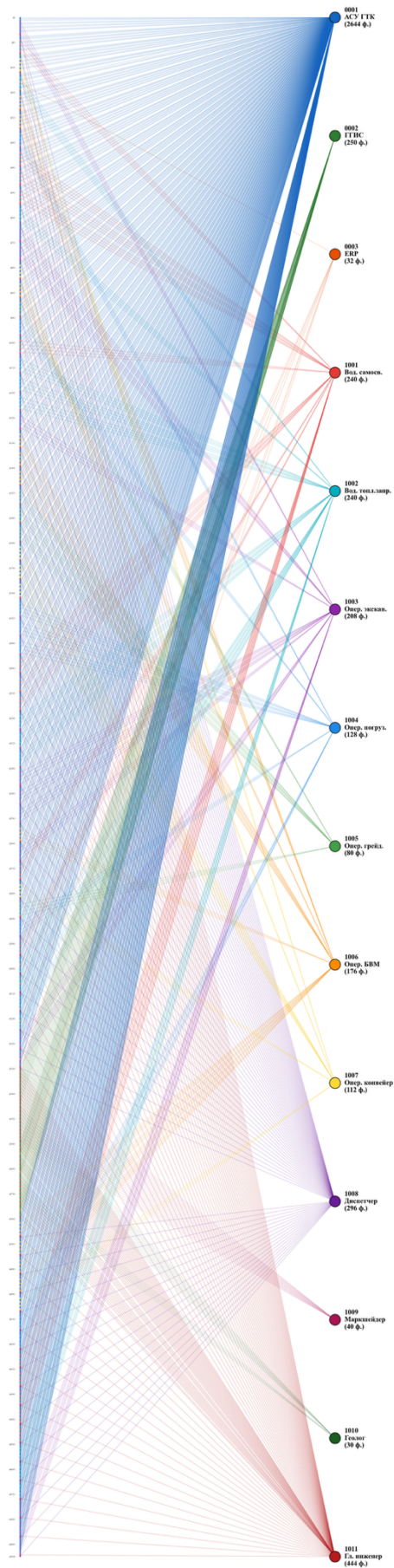


Рисунок 43. Граф связи атомарных функциональных задач управления и агентов управления

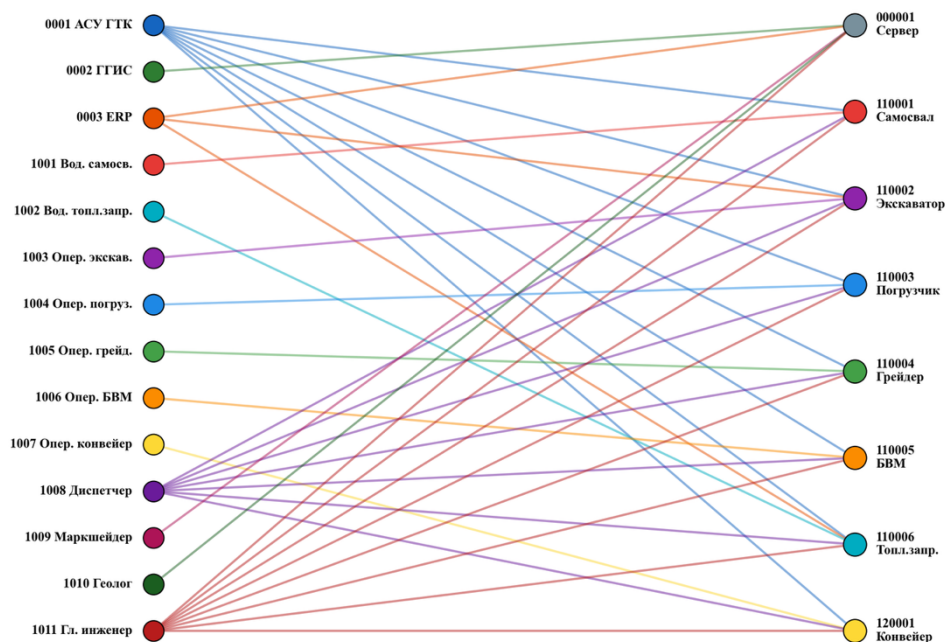


Рисунок 44. Граф связи агентов управления и физических узлов размещения

На Рисунке 45 показана гистограмма начального распределения экспериментальных атомарных функциональных задач управления между агентами управления.

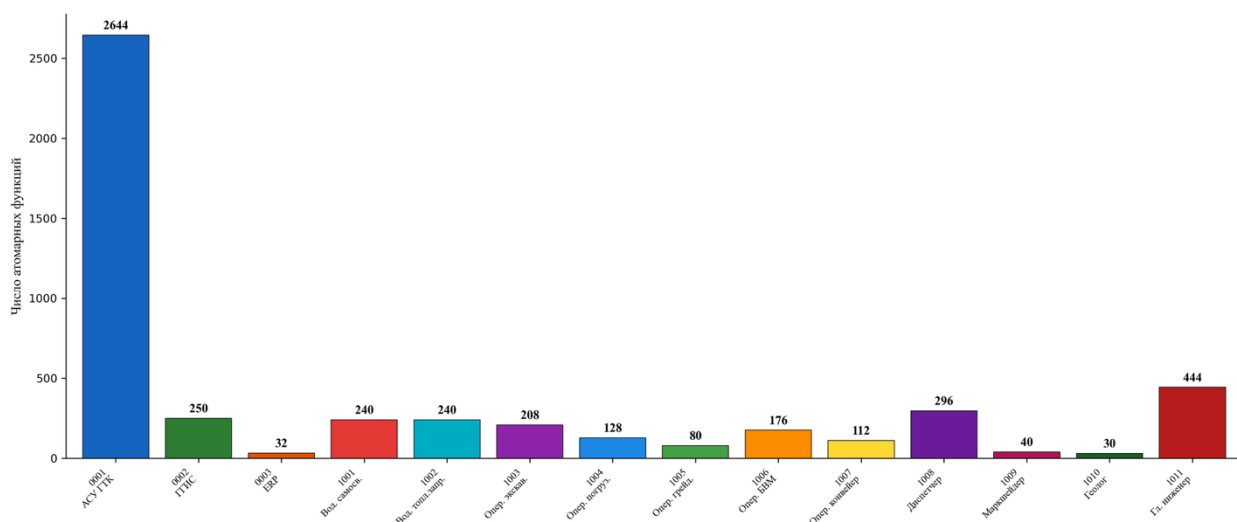


Рисунок 45. Гистограмма начального распределения атомарных функциональных задач управления между агентами управления

На основании полученного начального экспериментального распределения в соответствии с алгоритмом конфигурирования архитектуры «as is» была осуществлена программная реализация создания логических классов для каждой атомарной

функциональной задачи управления и их размещение на физических узлах в соответствии с заданными правилами конфигурирования и модификации.

На Рисунке 46 продемонстрирован фрагмент результата формирования таблицы начальной популяции логических классов атомарных функциональных задач управления.

1	φ	x^{lp}	x^{sp}	x^{lb}	x^{sb}	x^{lc}	x^{sc}	x^{ls}	x^{ss}
2	$\varphi 1$	0	1	0	1	0	1	0	1
3	$\varphi 2$	0	1	0	1	0	1	0	1
4	$\varphi 3$	0	1	0	1	0	1	0	1
5	$\varphi 4$	0	1	0	1	0	1	0	1
6	$\varphi 5$	0	1	0	1	0	1	0	1
7	$\varphi 6$	0	1	0	1	0	1	0	1
8	$\varphi 7$	0	1	0	1	0	1	0	1
9	$\varphi 8$	0	1	0	1	0	1	0	1
10	$\varphi 9$	0	1	0	1	0	1	0	1
11	$\varphi 10$	0	1	0	1	0	1	0	1
12	$\varphi 11$	0	1	0	1	0	1	0	1
13	$\varphi 12$	0	1	0	1	0	1	0	1
14	$\varphi 13$	0	1	0	1	0	1	0	1
15	$\varphi 14$	0	1	0	1	0	1	0	1
16	$\varphi 15$	0	1	0	1	0	1	0	1
17	$\varphi 16$	0	1	0	1	0	1	0	1
18	$\varphi 17$	0	1	0	1	0	1	0	1
19	$\varphi 18$	0	1	0	1	0	1	0	1
20	$\varphi 19$	0	1	0	1	0	1	0	1
21	$\varphi 20$	0	1	0	1	0	1	0	1

Рисунок 46. Начальная популяция логических классов атомарных функциональных задач управления

В Таблице 20 приведены оценка численности созданных логических классов начальной популяции.

Таблица 20

Численность логических классов начальной популяции

x^{lp}	x^{sp}	x^{lb}	x^{sb}	x^{lc}	x^{sc}	x^{ls}	x^{ss}
3840	3640	3840	3640	3840	3640	3840	3640
$x^p = 7480$		$x^b = 7480$		$x^c = 7480$		$x^s = 7480$	

Так в начальной популяции сформировано классов каждого типа больше, чем число атомарных функциональных задач управления – 4920, что является недопустимым по правилам конфигурирования архитектуры. Следуя алгоритму конфигурирования и правилам модификации для всех классов определено их размещение на физических узлах в

соответствии с метамоделью архитектуры Θ^0 . Результат сокращения популяции с учётом размещения на физических узлах представлен в Таблице 21.

Таблица 21

Результат сокращения начальной популяции логических классов по физическим узлам

x^{lp}	x^{sp}	x^{lb}	x^{sb}	x^{lc}	x^{sc}	x^{ls}	x^{ss}
3840	1080	1280	3640	1280	3640	1280	3640
$x^p = 4920$		$x^b = 4920$		$x^c = 4920$		$x^s = 4920$	

Далее каждому логическому классу присвоен сквозной индекс и применено правило слияния однотипных классов, принадлежащих одним и тем же агентам управления $C_i^{3,j}$ и размещаемых на одних и тех же физических узлах $u_m^{[l,s]}$ (Рисунок 47 и Таблица 22).

1	φ	x^{lp}	x^{sp}	x^{lb}	x^{sb}	x^{lc}	x^{sc}	x^{ls}	x^{ss}
2	$\varphi 1$	0	200001	0	400001	0	600001	0	800001
3	$\varphi 2$	0	200001	0	400001	0	600002	0	800001
4	$\varphi 3$	0	200001	0	400001	0	600003	0	800001
5	$\varphi 4$	0	200001	0	400001	0	600004	0	800001
6	$\varphi 5$	0	200001	0	400001	0	600005	0	800001
7	$\varphi 6$	0	200001	0	400001	0	600006	0	800001
8	$\varphi 7$	0	200001	0	400001	0	600007	0	800001
9	$\varphi 8$	0	200001	0	400001	0	600008	0	800001
10	$\varphi 9$	0	200001	0	400001	0	600009	0	800001
11	$\varphi 10$	0	200001	0	400001	0	600010	0	800001
12	$\varphi 11$	0	200001	0	400001	0	600011	0	800001
13	$\varphi 12$	0	200001	0	400001	0	600012	0	800001
14	$\varphi 13$	0	200013	0	400013	0	600013	0	800013
15	$\varphi 14$	0	200014	0	400014	0	600014	0	800014
16	$\varphi 15$	0	200015	0	400015	0	600015	0	800015
17	$\varphi 16$	0	200016	0	400016	0	600016	0	800016
18	$\varphi 17$	0	200001	0	400001	0	600017	0	800001
19	$\varphi 18$	0	200001	0	400001	0	600018	0	800001
20	$\varphi 19$	0	200001	0	400001	0	600019	0	800001

Рисунок 47. Фрагмент итоговой популяции логических классов атомарных функциональных задач управления

Таблица 22

Результат сокращения начальной популяции логических классов по физическим узлам

x^{lp}	x^{sp}	x^{lb}	x^{sb}	x^{lc}	x^{sc}	x^{ls}	x^{ss}
1544	461	1031	973	1280	3640	1031	973
$x^p = 2005$		$x^b = 2004$		$x^c = 4920$		$x^s = 2004$	

Из полученной итоговой популяции логических классов атомарных функциональных задач управления сформирована матрица смежности графа логической конфигурации архитектуры $G^0(A^0)$ в соответствии с разработанной моделью представления и правил конфигурирования $F_{cnfg}(A^0)$.

На Рисунке 48 представлен фрагмент графа архитектуры для агента «Диспетчер», демонстрирующий логическую конфигурацию классов и интерфейсов. Визуализация полного графа архитектуры автономно функционирующей кибер-физической пространственно-распределенной геотехнической индустриальной системы в состоянии «as is», включающая все типы агентов управления, представлена на Рисунке 49.

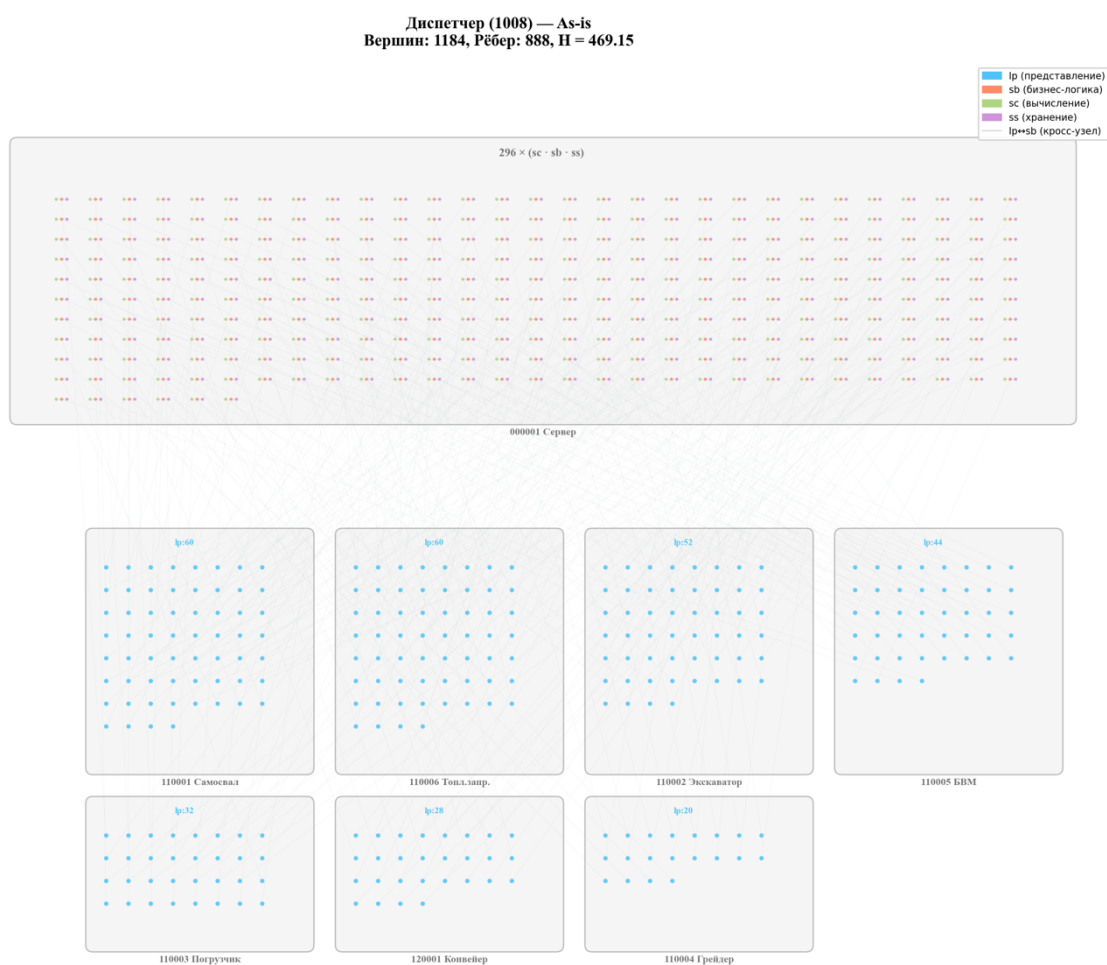


Рисунок 48. Фрагмент графа логической конфигурации архитектуры для агента управления «Диспетчер»

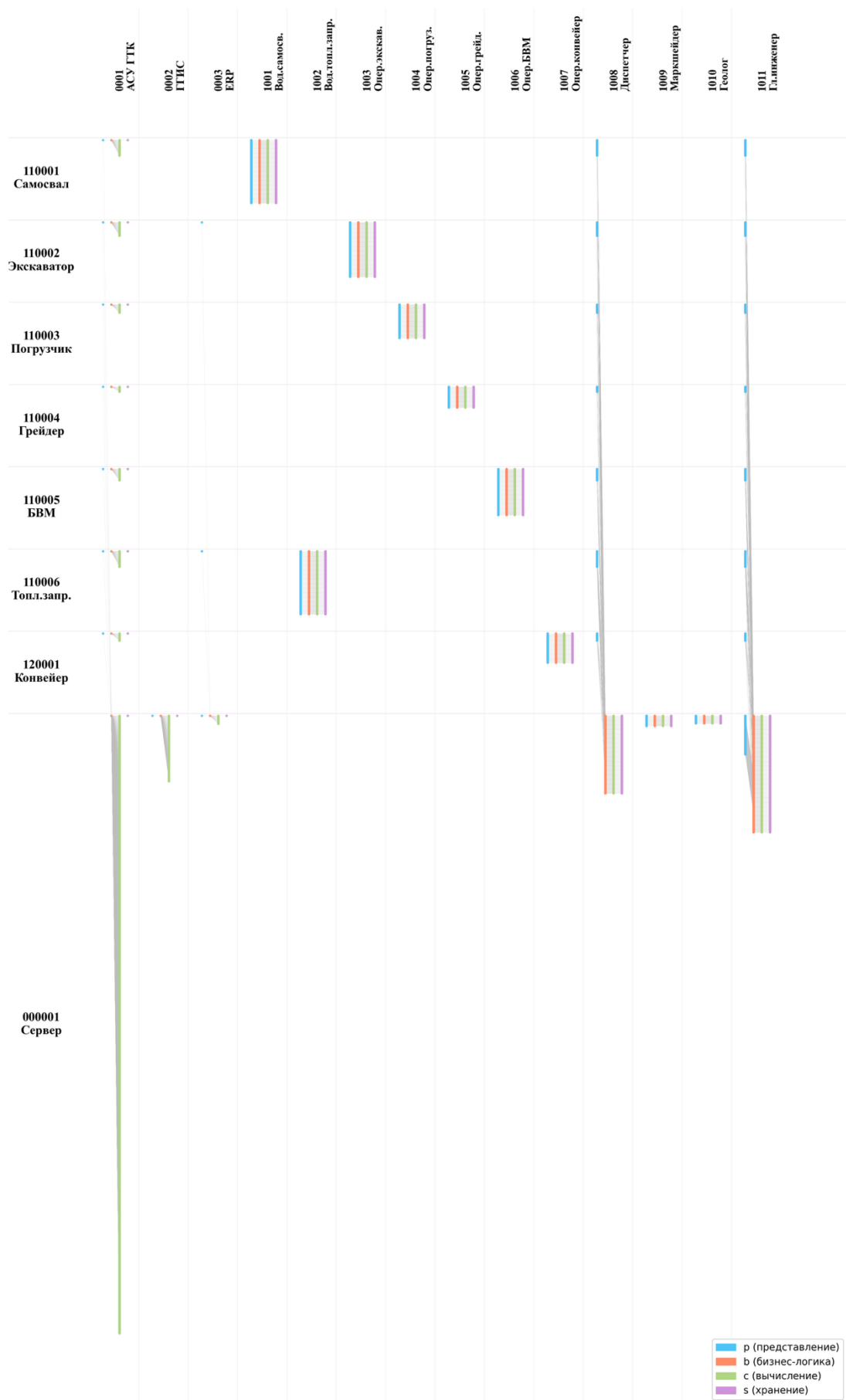


Рисунок 49. Граф логической конфигурации архитектуры автономно функционирующей кибер-физической пространственно-распределенной геотехнической индустриальной системы в состоянии «as is»

Итогом работы алгоритма конфигурирования и анализа архитектуры в состоянии «as is» является оценка структурной сложности логической конфигурации, отраженная в Таблице 23.

Таблица 23

Результат оценки структурной сложности логической конфигурации архитектуры в состоянии «as is»

$H^0(G^0(A^0))$	$N(\Phi)$	$I(C^{3j})$	$M(v)$	$\alpha(x)$	$\iota(y)$	$deg(min)$	$deg(max)$	$deg(mean)$
3 229,41	4 920	14	8	10 933	8 936	1	2 397	1,63

Таким образом продемонстрирован полный цикл работы алгоритма конфигурирования и анализа архитектуры автономно функционирующей киберфизической пространственно-распределенной геотехнической системы в состоянии «as is» на примере горнопромышленного предприятия открытой добычи полезных ископаемых.

4.2.2 Результаты моделирования работы алгоритма конфигурирования и анализа архитектуры «to be»

Для моделирования работы алгоритма конфигурирования и анализа архитектуры в состоянии «to be» были заданы следующие дополнительные условия экспериментальной модификации в целях определения возможностей сравнения получаемых результатов по аналогии с экспертно-аналитическим способом решения задач цифровой трансформации:

1. Перераспределение всех атомарных функциональных задач управления, реализуемых агентами управления группы «Персонал», между существующими агентами управления группы «Программные системы» (без введения в архитектуру новых агентов управления в состоянии «to be»).
2. Создание нового агента управления группы «Программные системы» $C_{i+1}^{3,j=0}$ и назначением ему всех атомарных функциональных задач управления, ранее реализуемых агента управления группы «Персонал».
3. Создание новых агентов управления группы «Программные системы» и переназначение всех атомарных функциональных задач управления таким агентам для приведения архитектуры системы к микросервисному виду со связями в графе g_1 один к одному (1:1).

При этом, следуя правилам конфигурирования архитектуры в состоянии «to be» $F_{cnfg}(A^*)$, для всех трёх вариантов должна быть введена атомарная функциональная задача управления интеграционным взаимодействием φ^ϕ , реализуемая новым соответствующим

программным агентом управления $C_{i+1}^{3,j=0}$ с размещением на серверном физическом узле u_m^s .

В рамках экспериментального моделирования на языке Python 3.2 реализован программный алгоритм конфигурирования и модификации архитектуры, результаты работы которого для Вариантов 1-3 представлены далее:

На Рисунках 50-51 показаны гистограммы итогового перераспределения атомарных функциональных задач управления между агентами управления для Вариантов 1 и 2 (для Варианта 3 с отношением перераспределения 1:1 не приводится в силу низкой информативности гистограммы и высокой размерности рисунка).

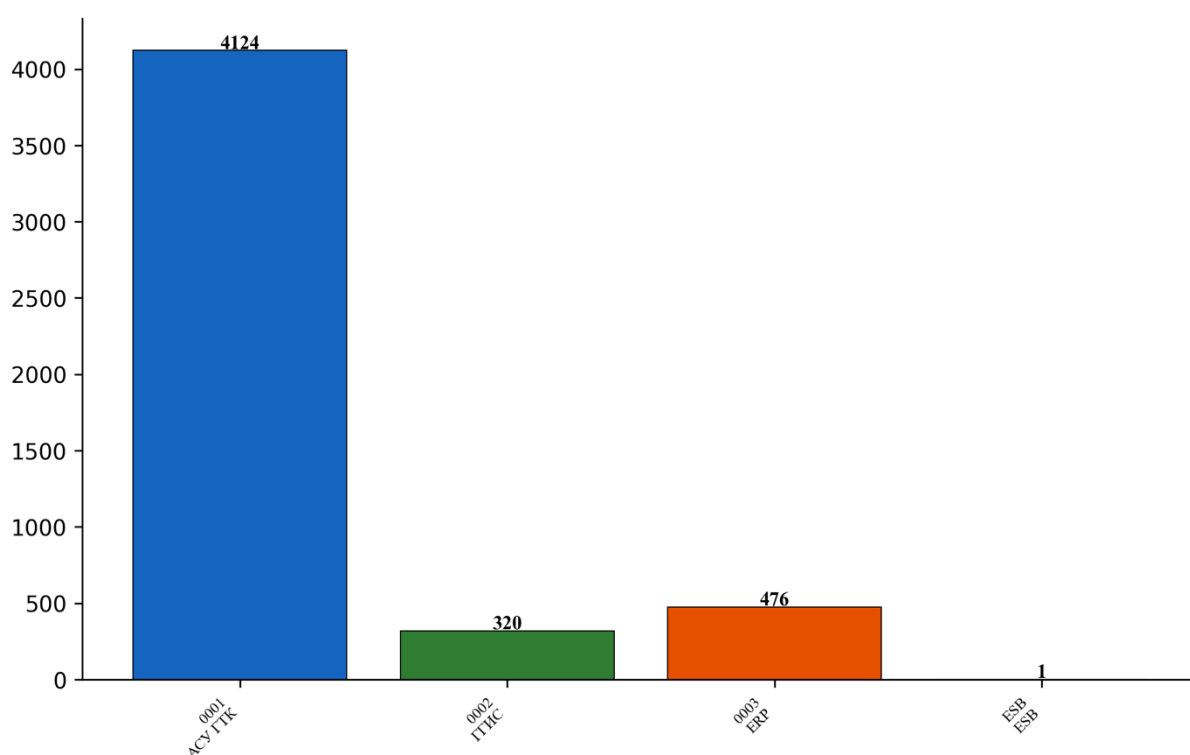


Рисунок 50. Итоговое распределение атомарных функциональных задач управления для Варианта 1 (по оси ординат – количество)

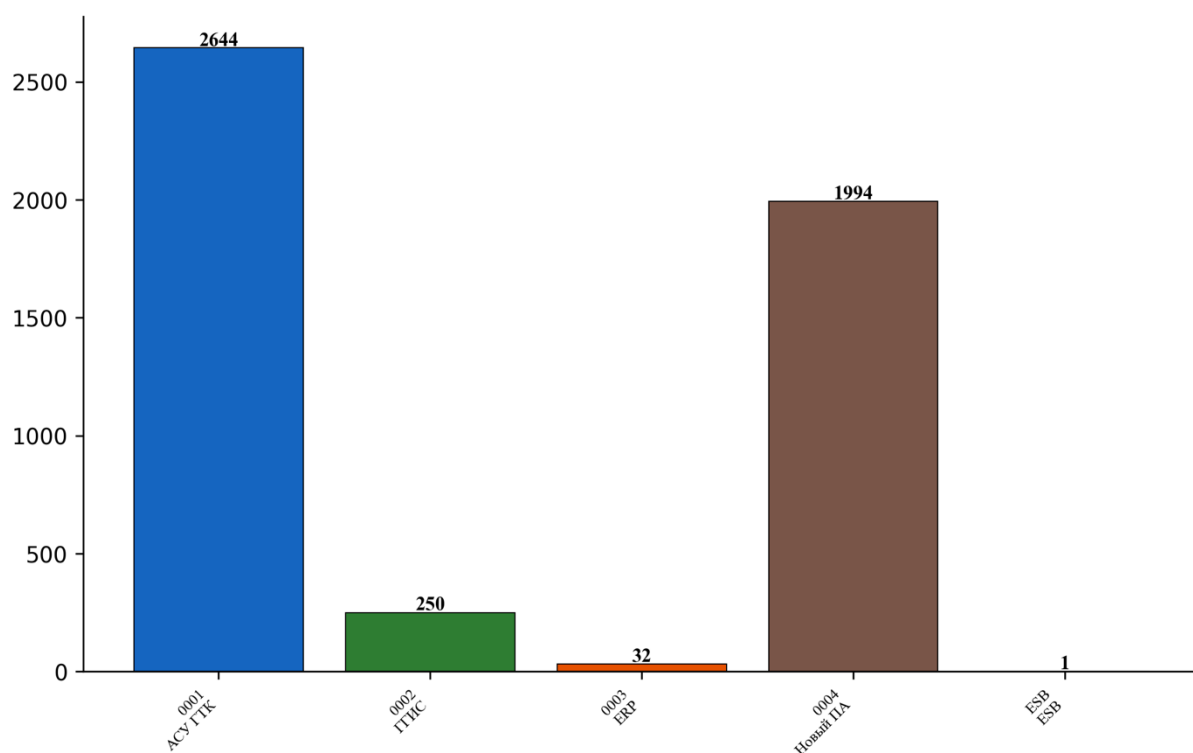


Рисунок 51. Итоговое распределение атомарных функциональных задач управления для Варианта 2 (по оси ординат – количество)

На Рисунках 52-54 представлены трехдольные графы $g_1^0 \rightarrow g_1^*$ перераспределения атомарных функциональных задач управления от агентов состояния «as is» к агентам состояния «to be». На Рисунках 55-57 представлены итоговые графы логической конфигурации архитектур в состоянии «to be» для Вариантов модификации 1-3, а на Рисунках 58-60 детализация перераспределения атомарных функциональных задач управления агента «Диспетчер» для Вариантов 1-3.

В Таблице 24 в приведены различные сравнительные оценки сложности итоговых вариантов логической конфигурации архитектур $H^*(G^*(A^*))$, включая оценку итоговой структурной сложности – $deg(x^*_\alpha)$, оценку сложности модификации – $deg^*(x^*_\alpha)$ и сложности только процедур модификации – $deg^{mod}(x^*_\alpha)$, рассчитанные в соответствии с предложенной методикой (91).

Таблица 24

Сравнительные оценки сложности модификации логической конфигурации архитектуры

$H^*(G^*(A^*))$	A^0	A_1^*	A_2^*	A_3^*
$deg(x^*_\alpha)$	3 229,41	94,2	143,11	9 853,85
$deg^*(x^*_\alpha)$	-	5 058,49	2 164,77	27 350,04
$deg^{mod}(x^*_\alpha)$	-	59,05	42,31	15 084,01

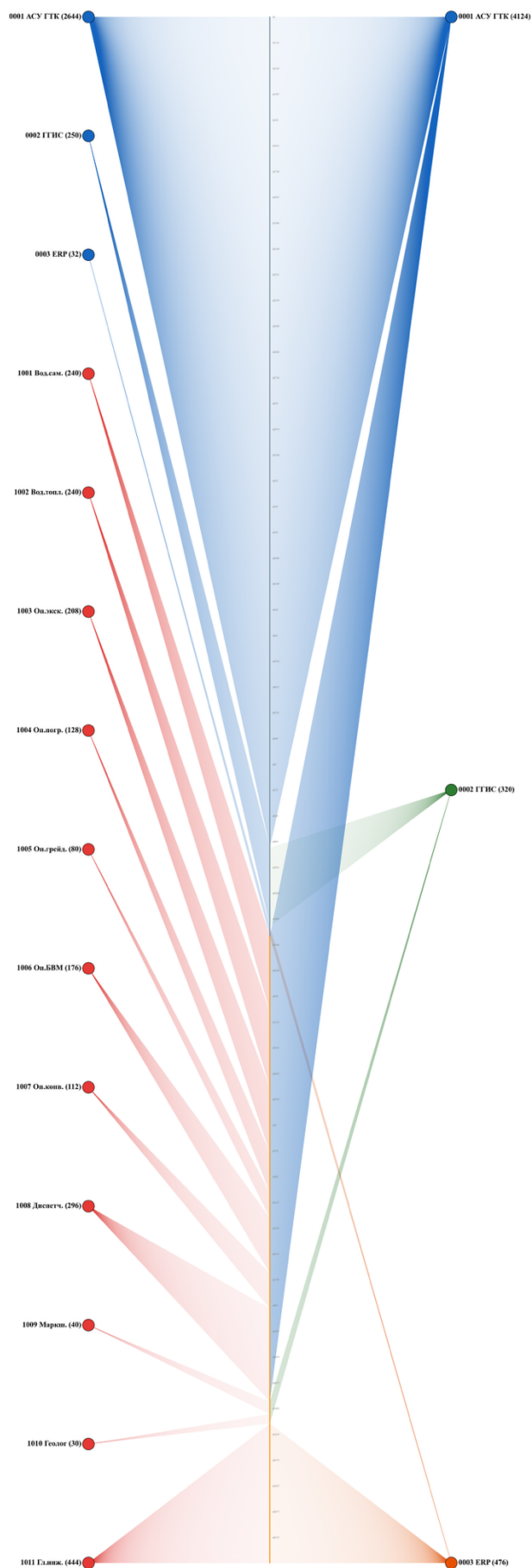


Рисунок 52. Граф перераспределения атомарных функциональных задач управления между агентами состояния «as is» и состояния «to be» для Варианта 1

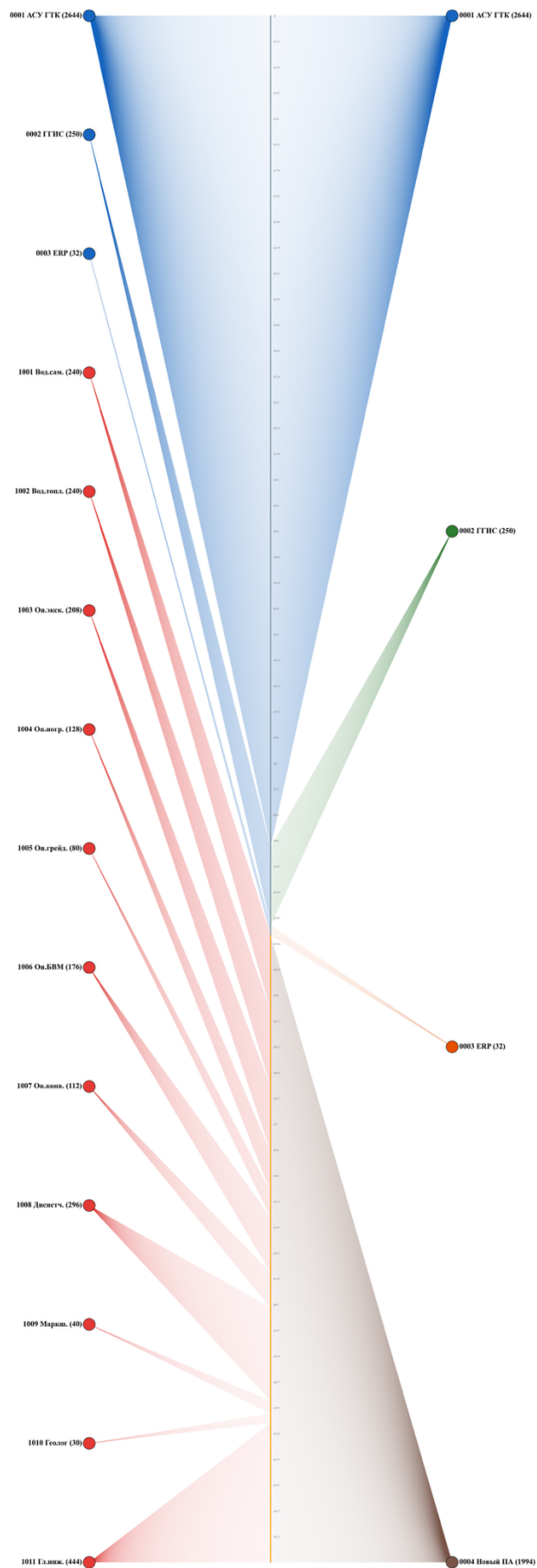


Рисунок 53. Граф перераспределения атомарных функциональных задач управления между агентами состояния «as is» и состояния «to be» для Варианта 2

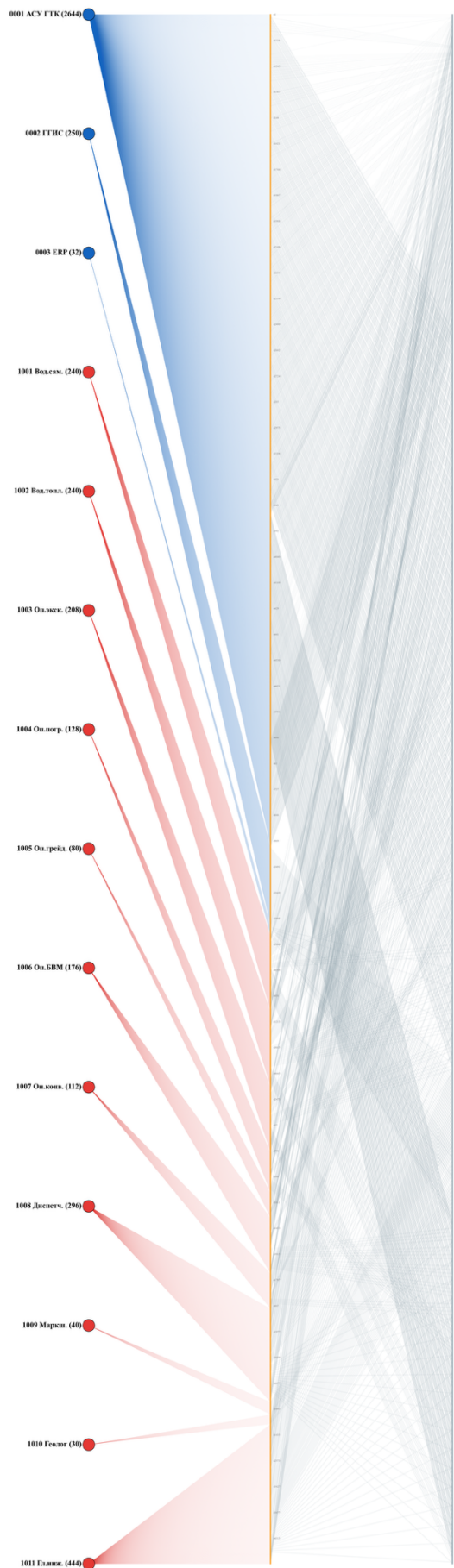


Рисунок 54. Граф перераспределения атомарных функциональных задач управления между агентами состояния «as is» и состояния «to be» для Варианта 3

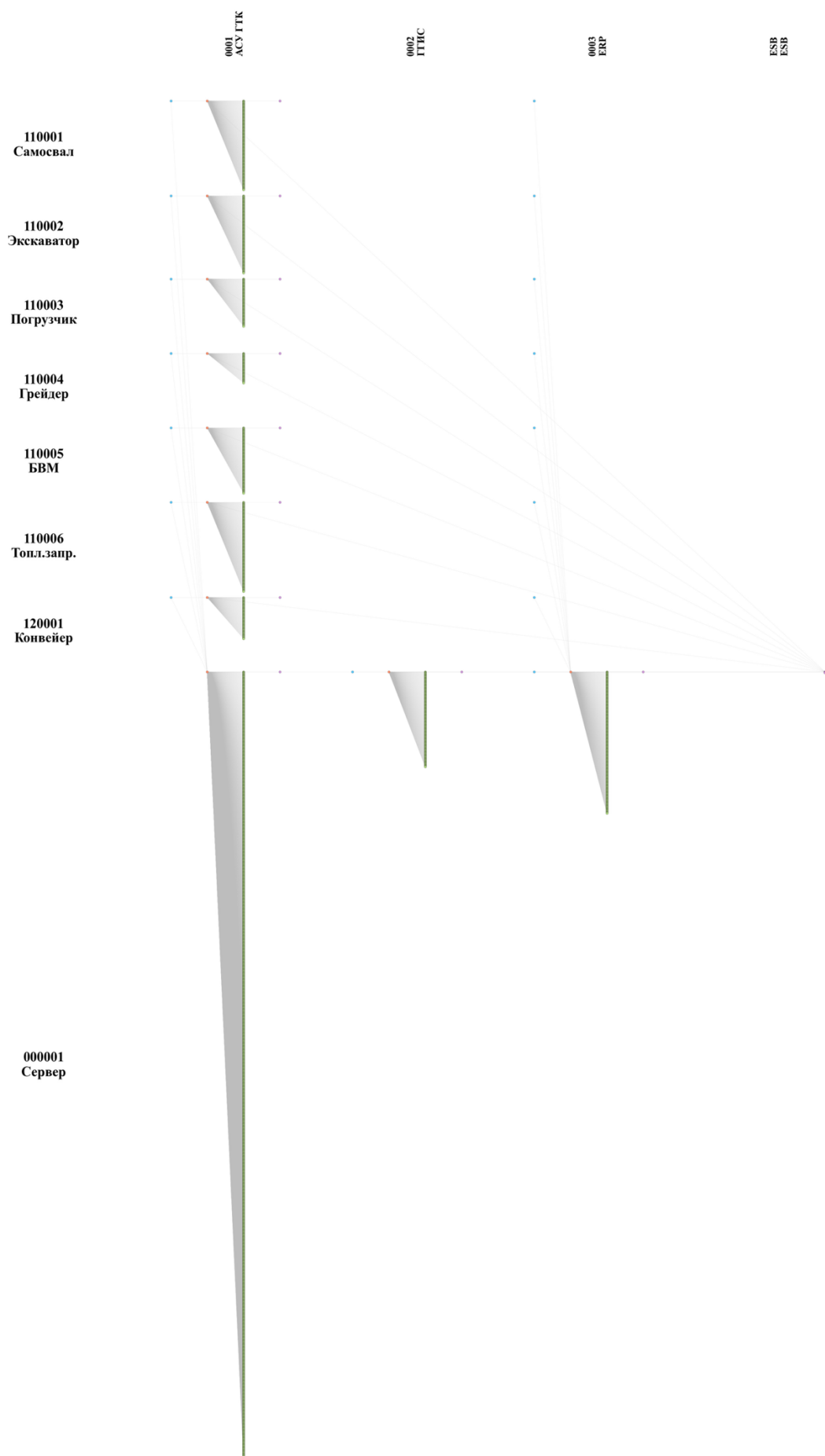


Рисунок 55. Граф логической конфигурации архитектуры в состоянии «to be» для Варианта 1

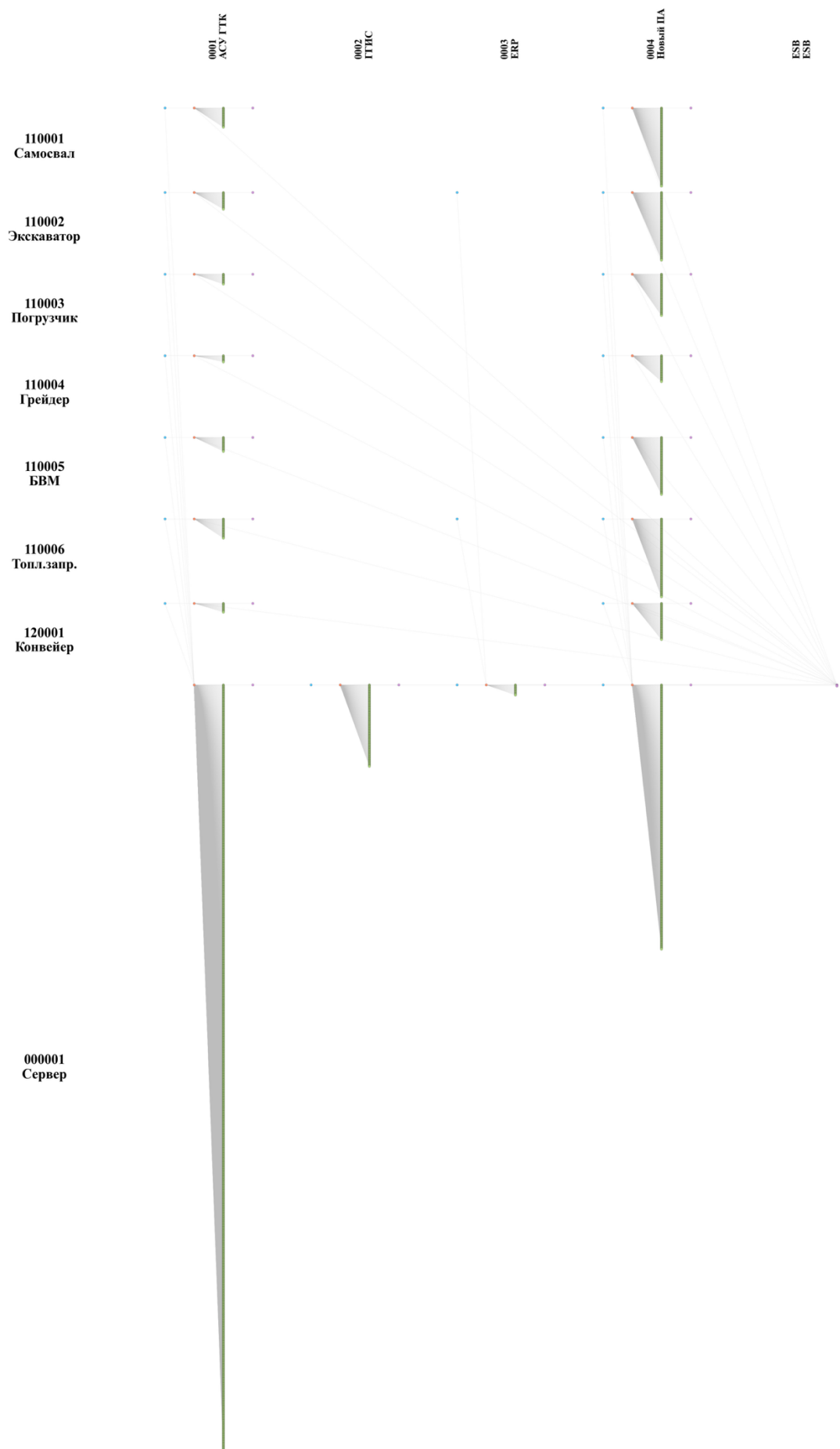


Рисунок 56. Граф логической конфигурации архитектуры в состоянии «to be» для Варианта 2

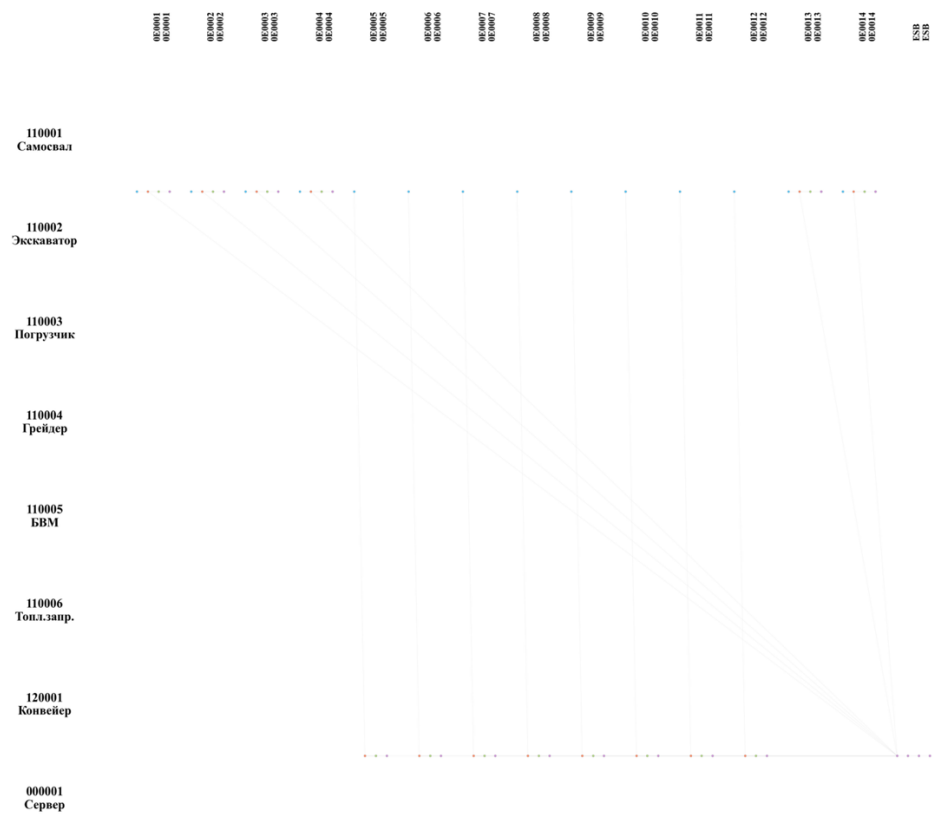


Рисунок 57. Фрагмент графа логической конфигурации архитектуры в состоянии «to be» для Варианта 3

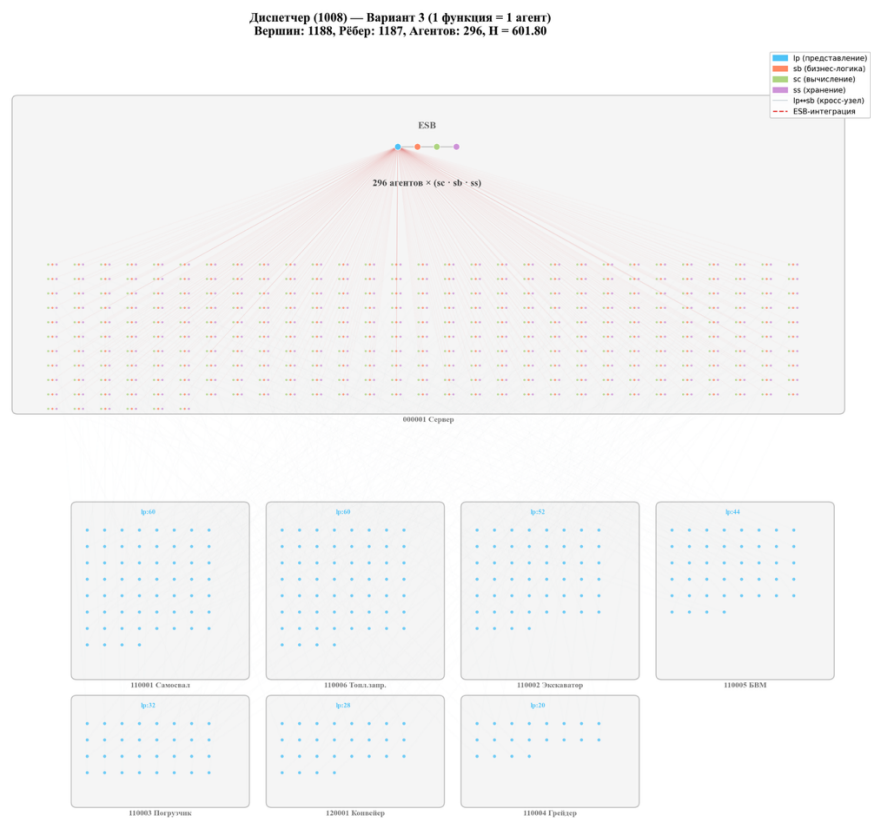


Рисунок 58. Фрагмент графа логической конфигурации архитектуры для перераспределенных атомарных функциональных задач управления агента «Диспетчер» в состоянии «to be» для Варианта 3

Диспетчер (1008) — Вариант 1 (→ АСУ ГТК)
Вершин: 309, Рёбер: 308, H = 10.84

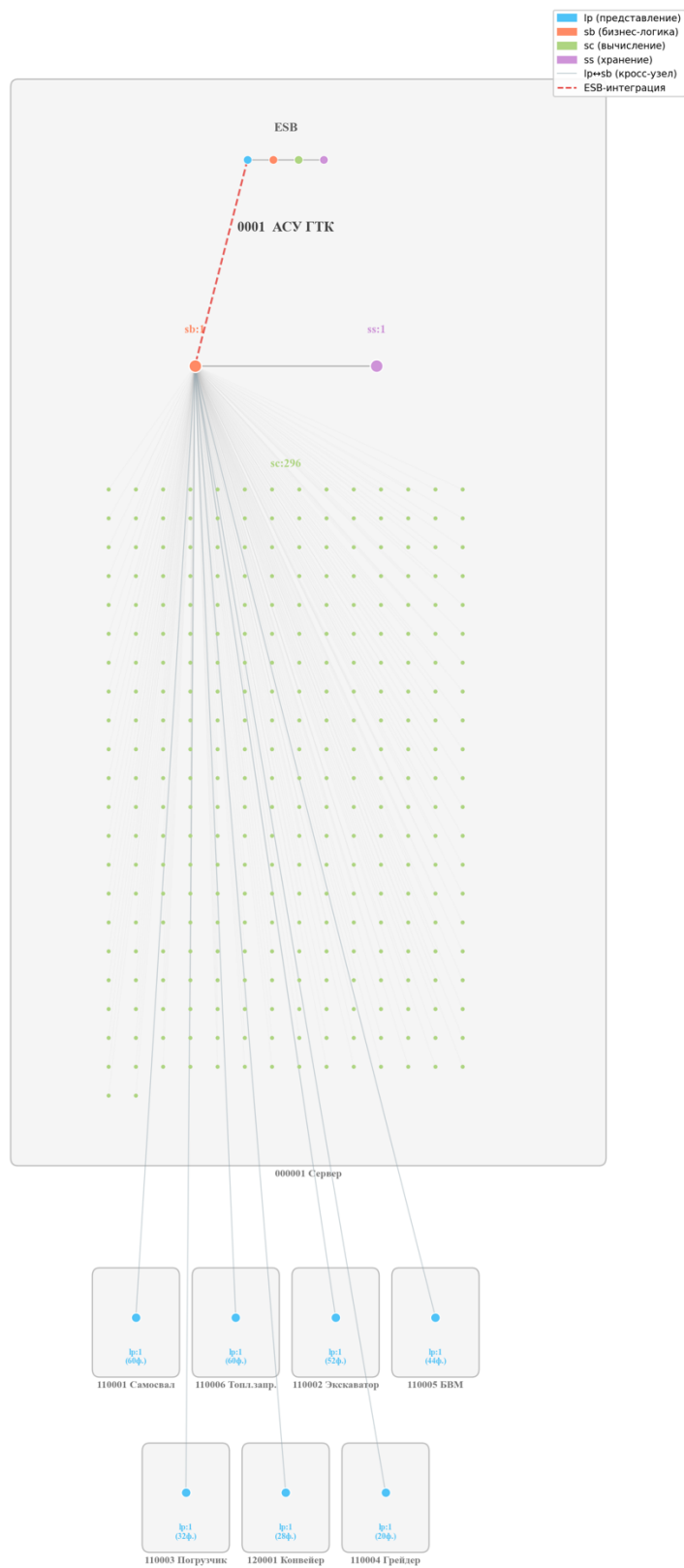


Рисунок 59. Фрагмент графа логической конфигурации архитектуры для перераспределенных атомарных функциональных задач управления агента «Диспетчер» в состоянии «to be» для Варианта 1

Диспетчер (1008) — Вариант 2 (→ Новый ПА)
Вершин: 309, Рёбер: 308, H = 10.84



Рисунок 60. Фрагмент графа логической конфигурации архитектуры для перераспределенных атомарных функциональных задач управления агента «Диспетчер» в состоянии «to be» для Варианта 2

Таким образом в соответствии с Таблицей 24 можно определить, что Вариант 1 имеет меньшую структурную сложность, что объясняется сокращением логических классов и интерфейсов связи. В то же время, сложность модификации и самих процедур модификации меньше для Варианта 2, что, вероятно, можно назвать более предпочтительным в условиях экспертно-аналитического анализа при решении задач управления жизненным циклом цифровой трансформации архитектур промышленных систем. Вариант 3, при котором реализуется микросервисная архитектура, напротив, показала худшие результаты при различных оценках, что, в некотором роде коррелирует с экспертно-аналитической оценкой стоимости трудозатрат на проведение подобной модификации.

Таблица 25

Параметры логических конфигураций архитектур

Параметр	A^0	A_1^*	A_2^*	A_3^*
$N(\varphi)$	4 920	4 920 + 1	4920 + 1	4920 + 1
$I(C^{3j})$	14	3 + 1	4 + 1	4920 + 1
$M(v)$	8	8	8	8
$\alpha(x)$	10 937	4 960	4 979	19 684
$\iota(y)$	8 939	4 966	4 992	19 683
$deg(min)$	1	1	1	1
$deg(max)$	2 356	2 653	2 357	4 921
$deg(mean)$	1,63	2	2,01	2
$deg(median)$	1	1	1	1
$\iota(y^{mod})$	0	5 050	2 096	34 989
$y^{mod}(min)$	0	1	0	0
$y^{mod}(max)$	0	12	11	2 916
$y^{mod}(mean)$	0	1,02	0,42	1,78
$y^{mod}(median)$	0	1	0	1

В Таблице 25 приведены параметры логических конфигураций архитектур в целях наглядного сопоставления и возможности проведения глубинного анализа результатов.

4.3 Анализ результатов

На основании результатов экспериментального моделирования можно сделать заключение, что предложенные модели, критерии и алгоритмы прошли верификацию и валидацию, а также доказали свою потенциальную применимость для решения реальных задач цифровой трансформации горнопромышленных систем.

Результаты моделирования работы механизма логико-семантического вывода показали возможность получения из онтологической модели архитектуры индустриальной системы каркасо-образующих каталогов и графа связи, необходимых для формирования начальной метамодели на примере горнопромышленного предприятия по открытой добычи полезных ископаемых. Разработанный механизм в достаточной мере удовлетворяет критериям точности и полноты вывода, сочетая возможности получения семантически корректных и уникальных наименований атомарных функциональных задач управления и комбинаторику перебора всех допустимых высказываний, определяющих структурно-функциональную организацию индустриальной системы. Обоснованием удовлетворения точности и полноты является фактическое сокращение результирующего вывода относительно потенциального теоретического предела слепого перебора в рамках декларируемых аксиом онтологии. Следует отметить, что степень полноты самой онтологической модели в настоящий момент зависит от ширины и глубины структуры знаний, закладываемых экспертами, однако их истинность и непротиворечивость может быть проверена формальными методами дескрипционной логики на базе современных инструментов проектирования онтологий (OWL, Protégé) и механизмов рассуждений (HermiT и др.). Таким образом предложенный подход обеспечивает сокращение рисков возникновения ошибок, характерных в традиционном экспертно-аналитическом подходе при формировании каталогов и матриц связи метамodelей архитектур систем. А комплексное использование современных инструментальных средств онтологического моделирования и предложенных способов построения механизмов логико-семантического вывода позволяет заключить об их высоком потенциале применимости для решения научно-практических задач в области управления цифровой трансформации сложных крупномасштабных пространственно-распределенных геотехнических индустриальных систем для их приведения к виду автономно функционирующих производств.

Моделирование работы алгоритма конфигурирования и анализа архитектуры в состоянии «as is» продемонстрировало практическую реализацию предложенного подход по формальному представлению логической конфигурации архитектур индустриальных систем и возможность получения численной оценки качества такой архитектуры с

использованием энтропийного критерия. Полученный граф логической конфигурации архитектуры наглядно демонстрирует практическую значимость предложенного критерия анализа, который позволяет количественно выявлять «узкие» места архитектуры – вершины с аномально высокой степенью, модификация которых затрагивает большое число смежных компонентов. В контексте задач цифровой трансформации это означает, что замена или обновление программных агентов, реализующих большое число функций, сопряжена с наибольшей стоимостью модификации, что может быть учтено при планировании этапов жизненного цикла архитектур. Также следует отметить преимущество данного подхода, обеспечивающего универсальность и равнозначность описания гетерогенных агентов управления через их единообразное представление в виде логической конфигурации атомарных функциональных задач управления.

Экспериментальное моделирование работы алгоритма конфигурирования и анализа архитектуры в состоянии «to be» показало работоспособность предложенных способов формальной модификации архитектур систем на базе операций редактирования графов, а также вариативность оценок качества в зависимости от выбранной стратегии цифровой трансформации. Так, исходя из результатов, приведенных в Таблицах 24 и 25 можно дать следующие выводы:

- Вариант 1 обеспечивает минимальную итоговую структурную сложность за счёт максимального слияния классов при сведении архитектуры к трём программным агентам. Наибольшее сокращение числа логических классов с 10 937 до 4 960 (в 2,2 раза) и рёбер с 8 939 до 4 966 обусловлено слиянием однотипных логических классов на общих физических узлах для агентов управления, что, однако, приводит к большей сложности модификации вследствие итерационного применения операций удаления (5 977 классов агентов группы «Персонал») и слияния, порождающих соответствующие петли модификации на смежных нодах.
- Вариант 2 демонстрирует компромиссный результат: структурная сложность несколько выше из-за появления нового программного агента, а, как следствие, увеличения числа классов и интерфейсов. В то же время сложность модификации существенно ниже, что объясняется меньшим числом петель за операции слияния с затрагиванием смежных нод.
- Вариант 3 показал наихудшие результаты по всем критериям. Резкое увеличение сложности объясняется принципиальным отсутствием процедур слияния: каждая из 4 920 функций реализуется собственным агентом, что порождает 19 684 уникальных логических класса (рост в 1,8 раза по сравнению с исходной архитектурой). Максимальная степень вершины (4 921) отражает то, что интеграционный агент ESB

связан со всеми функциями, а максимальное число петель на вершине (2 916) свидетельствует о чрезвычайно высокой стоимости модификации центральных компонентов.

Данные результаты количественно подтверждают неформализованный тезис в области практики программной инженерии о высокой интеграционной сложности микросервисной архитектуры. При этом необходимо отметить, что контртезис о преимуществе микросервисной архитектуры на этапе поддержки в рамках полученных результатов не подтвержден и не опровергнут, но может быть дополнительно проверен с использованием моделирования на базе критерия (98).

Результаты экспериментального моделирования наглядно выявляют фундаментальную проблему в области решения научно-исследовательских и прикладных задач управления архитектурами сложных крупномасштабных систем: поиск компромисса между конкурентными критериями минимизации структурной сложности и минимизации сложности модификации. При этом предложенный в рамках настоящего исследования подход формализует единый критерий оптимизации (96), согласовывающий достижение архитектуры с минимальной структурной сложностью при минимальной сложности модификации для её достижения. Однако для действительного поиска оптимального пути цифровой трансформации архитектуры необходимо разработать соответствующие алгоритмы на базе предложенного критерия, что будет реализовано в последующих исследованиях.

Полученные результаты в целом доказывают, что разработанные модели и алгоритмы обеспечивают:

- количественную и воспроизводимую оценку качества архитектур, инвариантную к состоянию архитектуры индустриальной системы и способам её структурно-функциональной организации;
- возможность формального анализа и сопоставления различных стратегий цифровой трансформации архитектур по единому критерию качества на базе энтропийной оценки сложности.

ЗАКЛЮЧЕНИЕ

По результатам настоящего исследования решена актуальная задача повышения эффективности процессов цифровой трансформации горнопромышленных предприятий посредством разработки моделей и методов оценки и анализа архитектур для построения автономно функционирующих кибер-физических пространственно-распределенных геотехнических промышленных систем, включая:

1. Формализованы критерии и задачи управления цифровой трансформацией архитектур сложных крупномасштабных пространственно-распределенных геотехнических предприятий. Архитектура представлена как совокупность четырёх групп гетерогенных компонент – технологических процессов, физических объектов, агентов управления и информации, обуславливающих структурно-функциональную организацию предприятий. Сформулированы условия проведения ЦТ предприятия как процесса замещения персонала программными агентами при сохранении функциональной пригодности.
2. Предложен подход формализации априорных знаний о структурно-функциональной конфигурации предприятия для составления метамодели архитектуры посредством онтологического моделирования. В соответствии с предложенным подходом разработана двухкомпонентная онтологическая модель архитектуры на языке дескрипционной логики OWL, объединяющая универсальную таксономию концептов пространственно-распределенной геотехнической промышленной системы с конкретизированным набором индивидов для типового горнопромышленного предприятия по открытой добычи полезных ископаемых.
3. Предложен оригинальный механизм логико-семантического вывода ключевых компонентов метамодели, обуславливающих архитектуру промышленной системы инвариантно этапам жизненного цикла цифровой трансформации. Механизм реализует строгий комбинаторный перебор допустимых аксиоматических высказываний для получения семантически точных и полных каталогов атомарных функциональных задач управления и физических узлов размещения, а также граф их взаимосвязи.
4. Разработан алгоритм формирования метамодели архитектуры промышленной системы, обеспечивающий минимизацию рисков возникновения ошибок традиционного экспертно-аналитического подхода посредством комплексного применения средств дескрипционной логики и механизмов строгого

комбинаторного перебора с доказательством непротиворечивости, полноты и точности знаний.

5. Разработаны модель представления архитектуры в виде графа логической конфигурации, согласовывающая понятия архитектуры предприятия с архитектурами программных систем через единообразное описание атомарных функциональных задач управления, и унифицированные информационно-энтропийные критерии количественной оценки качества архитектур, обеспечивающие возможность сопоставления и анализа её состояний инвариантно этапам жизненного цикла.
6. Разработан набор формальных операций модификации логической конфигурации архитектуры с эвристической метрикой стоимости в виде петель модификации и правилами допустимости, позволяющими количественно оценивать и сопоставлять варианты структурных преобразований.
7. Сформулирован критерий оптимизации в виде минимизации сложности по множеству допустимых путей редактирования графа архитектуры на базе различных вариантов энтропийных оценок, что обеспечивает возможность обоснованного выбора стратегии цифровой трансформации промышленной системы.
8. Разработаны алгоритмы процедур конфигурирования и анализа архитектур, обеспечивающих формальную инструментальную основу решения задач управления жизненным циклом цифровой трансформации для построения автономно функционирующих кибер-физических пространственно-распределенных геотехнических промышленных систем.

Теоретическое значение и практическая значимость достигнутых результатов обусловлена воспроизводимостью экспериментального моделирования на примере горнопромышленного предприятия по открытой добыче полезных ископаемых, в ходе которого верифицированы полнота и точность логико-семантического вывода, продемонстрирована работоспособность предложенных моделей, методов и алгоритмов конфигурирования и получены количественные оценки для различных вариантов модификации архитектуры.

Предложенный в настоящем исследовании подход и полученные результаты имеют ряд ограничений, а именно:

1. Критерии сложности рассчитаны только для логической конфигурации архитектуры и не учитывают технические аспекты функционирования промышленной системы в части: технологического стека реализации агентов управления, определяющих допустимость слияния однотипных классов; программно-аппаратных требований

агентов управления и характеристик физических узлов, требующих сопоставления при их взаимном размещении; структурно-функциональной организации каталогов экземпляров агентов управления и экземпляров физических узлов, обуславливающих масштаб индустриальной системы в виде множественного воспроизведения однотипных атомарных функциональных задач управления.

2. Модель оценки стоимости модификации на основе петель является эвристической метрикой, коррелирующей, но явно не обуславливающей фактическую технико-экономическую стоимость проведения цифровой трансформации архитектур сложных крупномасштабных индустриальных систем. Для получения подобных оценок могут быть использованы дополнительные количественные критерии стоимости процедур модификации, в том числе учитывающие технические аспекты структурно-функциональной конфигурации архитектуры.
3. В настоящей работе не реализованы алгоритмы конфигурационной оптимизации архитектуры на базе предложенных критериев, а также механизмы управления динамическими изменениями, вызванными недетерминированными во времени возможностями имплементации в архитектуре новых программных агентов управления, замещающих персонал в реализации атомарных функциональных задач управления.

Таким образом перспективными направлениями дальнейших исследований, развивающих полученные результаты являются:

- разработка алгоритмов конфигурационной оптимизации архитектуры на базе жадных, эвристических или метаэвристических подходов минимизации сложности жизненного цикла ЦТ;
- расширение моделей и проведение экспериментальных вычислений с учётом технических аспектов функционирования архитектур индустриальных систем;
- обобщение результатов на другие типы пространственно-распределённых геотехнических индустриальных систем (нефтегазовый сектор, лесная промышленность и др.) путём формирования соответствующих онтологий специального назначения;
- интеграция механизма логико-семантического вывода с рассуждателями дескриптивной логики для автоматизации процедур построения и обновления метамоделей архитектур;
- реализация СППР управления жизненным циклом ЦТ как необходимой инструментальной вычислительной основы самоорганизации и саморегуляции автономно функционирующей кибер-физической индустриальной системы.

СПИСОК ИСТОЧНИКОВ

1. Alkhalaf T., Durrah O., Alkhalaf M. Digital Transformation, Business Model, and Performance in the Context of Digital SMEs: Evidence From France. *Digital Technologies for Sustainability and Quality Control*, IGI Global Scientific Publishing, 2025, 101-120, DOI: 10.4018/979-8-3693-4373-9.ch005
2. Макаренко Я.В., Соловьева И.А. Цифровая трансформация предприятия: ключевые дефиниции, принципы и подходы. *Вестник Южно-Уральского государственного университета, Серия «Экономика и менеджмент»*, 2025, 19(1), 112-123, DOI: 10.14529/em250109
3. Litvinenko V.S. Digital Economy as a Factor in the Technological Development of the Mineral Sector. *Natural Resources Research*, 2019, 29(3), 1521-1541, DOI: 10.1007/s11053-019-09568-4
4. Лукичев С.В., Наговицын О.В. Цифровая трансформация горно-добывающей промышленности: прошлое, настоящее, будущее. *Горный журнал*, 2020, 9, 13-18, DOI: 10.17580/gzh.2020.09.01
5. Владимиров Д.Я., Клебанов А.Ф., Кузнецов И.В. Цифровая трансформация открытых горных работ и новое поколение карьерной техники. *Горная промышленность*, 2020, 6, 10-12, DOI: 10.30686/1609-9192-2020-6-10-12
6. Thoben K.-D., Wiesner S., Wuest T. “Industrie 4.0” and smart manufacturing—a review of research issues and application examples. *Int. J. Autom. Technol*, 2017, 11, 4–16, DOI: 10.20965/ijat.2017.p0004
7. Borangiu T., Morariu O., Răileanu S., Trentesaux D., Leitão P., Barata J. Digital transformation of manufacturing. Industry of the future with cyber-physical production systems. *Rom. J. Inf. Sci. Technol*, 2020, 23, 3–37
8. Soomro M., Hizam-Hanafiah M., Abdullah N., Ali M., Jusoh M. Embracing industry 4.0: Empirical insights from Malaysia. *Informatics*, 2021, 8, 30 DOI: 10.3390/informatics8020030
9. Ржевский В.В., Трубецкой К.Н. Задачи горной науки в области открытой разработки месторождений полезных ископаемых. *Горный журнал*, 1988, 1, 21
10. Пучков Л.А., Федунец Н.И., Потресов Д.К. Автоматизированные системы управления в горнодобывающей промышленности. 1987
11. Klebanov, A.F. Information systems of mining production and the main directions of development of automation of open pit mining. *Min. Ind.*, 2015, 2, 93–95

12. Клебанов А.Ф., Бондаренко А.В., Жуковский Ю.Л., Клебанов Д.А. Организация удаленных центров управления горным предприятием: стратегические предпосылки и этапы реализации. Горная промышленность, 2024, 4, 174-183, DOI: 10.30686/1609-9192-2024-4-174-183
13. Захаров В.Н., Каплунов Д.Р., Клебанов Д.А., Радченко Д.Н. Методические подходы к стандартизации сбора, хранения и анализа данных при управлении горнотехническими системами. Горный журнал, 2022, 12, 55-61, DOI: 10.17580/gzh.2022.12.10
14. Kiziroglou M.E., Boyle D.E., Yeatman E.M., Cilliers J.J. Opportunities for sensing systems in mining. IEEE Transactions on Industrial Informatics, 2016, 13(1), 278-286, DOI: 10.1109/TII.2016.2636131
15. Park S., Choi Y. Development and Utilization of Mine Management Software: A Review. Tunnel and Underground Space, 2015, 25(3), 221-230, DOI: 10.7474/TUS.2015.25.3.221
16. Temkin I. O., Klebanov D. A., Deryabin S. A., Konov I. S. Construction of intelligent geoinformation system for a mine using forecasting analytics techniques. Mining Inf. Anal. Bull., 2020, 3, 114-125, DOI: 10.25018/0236-1493-2020-3-0-114-125
17. Клебанов А.Ф., Сиземов Д.Н., Кадочников М.В. Комплексный подход к удаленному мониторингу технического состояния и режимов эксплуатации карьерного автосамосвала. Горная промышленность. 2020, 2, 75-81, DOI: 10.30686/1609-9192-2020-2-75-81
18. Koteleva N., Valnev V. Automatic Detection of Maintenance Scenarios for Equipment and Control Systems in Industry. Applied Sciences, 2023, 13(24), 12997, DOI: 10.3390/app132412997
19. Lel Yu. I., Arefev S. A., Stenin Yu. V., Sandrigaylo I. N., Glebov I. A. et al. Method of assessment of mining-technical and road conditions of exploitation of open-pit dump trucks. Innovation geotechnologies during ore and non-metallic deposits mining: collection of reports of the III International scientific and technical conference. Ekaterinburg: Izdatelstvo UGGU, 2014, 19– 22.
20. Odintsev N. V., Klebanov D. A. Roadway quality control for mining companies. Gornaya promyshlennost, 2009, 3, 24–26
21. Temkin I., Deryabin S., Konov I., Klebanov D. Haul road condition determination under controlled interaction of robotic elements in open pit mining and transport system. Gornyi Zhurnal, 2018, DOI:10.17580/gzh.2018.01.14
22. Dadonov M., Kulpin A., Ostanin O., Suleimenov E. Distribution of static normal reactions to wheels of open-pit dump trucks depending on the longitudinal and cross sections of the

- open-pit road. E3S Web of Conferences, International Innovative Mining Symposium, 2019, 105, 03009, DOI: 10.1051/e3sconf/201910503009
23. Jhavian R., Brito L., Behzadan A. Integrated mobile sensor-based activity recognition of construction equipment and human crews. Proceedings of the 2015 Conference on Autonomous and Robotic Construction of Infrastructure, 2015, 1-20
 24. Temkin I., Klebaniv D., Deryabin S., Konov I. Predictive Analytics in Mining. Dispatch System Is the Core Element of Creating Intelligent Digital Mine. Modern Information Technology and IT Education, 2020, 365-374
 25. Varichev A.V., Kretov S.I., Ismagilov R.I., Badtiev B.P., Vladimirov D.Ya. An integrated approach to intelligent systems of mining management. Gornaya promyshlennost', 2016, 3(127), 4-7
 26. Kammoun H.M., Kallel I., Casillas J., Abraham A., Alimi A.M. Adapt-Traf: An adaptive multiagent road traffic management system based on hybrid ant-hierarchical fuzzy model. Transportation Research Part C: Emerging Technologies, 2014, 42, 147–167, DOI: 10.1016/j.trc.2014.03.003
 27. Ghandriz T., Jacobson B., Laine L., Hellgren J. Impact of automated driving systems on road freight transport and electrified propulsion of heavy vehicles. Transportation Research Part C: Emerging Technologies, 2020, 115, 102610, DOI: 10.1016/j.trc.2020.102610
 28. Dadhich S., Bodin U., Anderson U. Key challenges in automation of earth-moving machines. Autom. Constr., 2016, 68, 212–222, DOI: 10.1016/j.autcon.2016.05.009
 29. Yan Z., Jouandeau N., Cherif A.A. A survey and analysis of multi-robot coordination. Int. J. Adv. Robot. Syst., 2013, 10, 399, DOI: 10.5772/57313
 30. Marshall J.A., Bonchis A., Nebot E., Scheduling S. Robotics in mining. Springer handbook of robotics. Siciliano B. et al. (Ed.), 2016, 1549-1576, DOI: 10.1007/978-3-319-32552-1_59
 31. Trubetskoy K.N., Vladimirov D.Ya., Pytalev I.A., Popova T.M. Robotic systems for open pit mineral mining. Gornyy zhurnal, 2016, 5, 21-27, DOI: 10.17580/gzh.2016.05.01
 32. Nagovitsyn O.V., Voznyak M.G. Robotic mine management. Mining Inf. Anal. Bull., 2021, 5(1), 326-335, DOI: 10.25018/0236_1493_2021_51_0_326
 33. Oxborrow J. R. Implementation of an Autonomous Haulage System on retrofitted haul trucks at Nevada Gold Mines Goldstrike operations. SME Annual Conference and Expo2020, 2020, 159172
 34. Ishimoto H., Hamada T. Safety Concept and Architecture for Autonomous Haulage System in Mining. Proceedings of the 37th International Symposium on Automation and Robotics in Construction, 2020, 377–384, DOI: 10.22260/isarc2020/0054

35. Gaber T., Jazouli Y. E., Eldesouky E., Ali A. Autonomous haulage systems in the mining industry: cybersecurity, communication and safety issues and challenges. *Electronics*, 2021, 10(11), DOI: 10.3390/electronics10111357
36. Durrant-White H., Henderson T. *Multisensor Data Fusion*. Springer Handbook of Robotics, 867–896, DOI: 10.1007/978-3-540-30301-5_26
37. Lee C., Suh J., Baek J., Choi Y. Review of Collision Avoidance Systems for Mine Safety Management: Development Status and Applications. *Tunnelling and Underground Space Technology*, 2017, 27(5), DOI: 10.7474/TUS.2017.27.5.282
38. Leitão P., Colombo A.W., Karnouskos S. Industrial automation based on cyber-physical systems technologies: Prototype implementations and challenges. *Computers in Industry*, 2016, 81, 11-25, DOI: 10.1016/j.compind.2015.08.004
39. Lee J., Bagheri B., Kao H.-A.A. Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems. *Manufacturing Letters*, 2015, 3, 18-23, DOI: 10.1016/j.mfglet.2014.12.001
40. Kunath M., Winkler H. Integrating the Digital Twin of the manufacturing system into a decision support system for improving the order management process. *Procedia CIRP*, 2018, 72, 225-231, DOI: 10.1016/j.procir.2018.03.192
41. Negria E., Fumagallia L., Macchia M. A review of the roles of Digital Twin in CPS-based production systems. *Procedia Manufacturing*, 2017, 11, 939-948, DOI: 10.1016/j.promfg.2017.07.198
42. Kritzinger W., Karner M., Traar G., Henjes J., Sihn W. Digital Twin in manufacturing: A categorical literature review and classification. *IFAC-PapersOnLine*, 2018, 51(11), 1016-1022, DOI: 10.1016/j.ifacol.2018.08.474
43. Temkin I.O., Myaskov A.V., Deryabin S.A., Rzazade U.A. Digital Twins and modeling of the transporting-technological processes for the on-line dispatch control in open pit mining. *Eurasian mining*, 2020, 2, 55-58, DOI: 10.17580/em.2020.02.13
44. Deryabin S., Temkin I., Zykov S. About some issues of developing Digital Twins for the intelligent process control in quarries. *Procedia Computer Science*, 2020, 176, 3210-3216, DOI: 10.1016/j.procs.2020.09.128
45. Hu L., Nguyen N.-T., Tao W., Leu M.C. and others Modeling of Cloud-Based Digital Twins for Smart Manufacturing with MT Connect. *Procedia Manufacturing*, 2018, 26, 1193-1203, DOI: 10.1016/j.promfg.2018.07.155
46. Cheng J., Zhang H., Tao F., Juang C.-F. DT-II: Digital twin enhanced Industrial Internet reference framework towards smart manufacturing. *Robotics and Computer-Integrated Manufacturing*, 2020, 62, 101881, DOI: 10.1016/j.rcim.2019.101881

47. Junliang W., Chuqiao X., Jie Z., Jingsong B., Ray Z. A collaborative architecture of the industrial internet platform for manufacturing system. *Robotics and Computer-Integrated Manufacturing*, 2020, 61, 101854, DOI: 10.1016/j.rcim.2019.101854
48. Martínez P.L., Dintén R., Drake J.M., Zorrilla M. A big data-centric architecture metamodel for Industry 4.0. *Future Generation Computer Systems*, 2021, 125, 263-284, DOI: 10.1016/j.future.2021.06.020
49. Petrasch R., Hentschke R. Process Modeling for Industry 4.0 Applications: Towards an Industry 4.0 Process Modeling Language and Method. *13th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, 2016, DOI: 10.1109/JCSSE.2016.7748885
50. Deryabin S.A., Temkin I., Rzazade U., Kondratev E. Models and Methods of Designing Data-Centric Microservice Architectures of Digital Enterprises. *informatics*, 2023, 10(1), 4, DOI: 10.3390/informatics10010004
51. Al-Osta M., Ahmed B., Abdelouahed G. A Lightweight Semantic Web-based Approach for Data Annotation on IoT Gateways. *Procedia Computer Science*, 2017, 113, 186-193, DOI: 10.1016/j.procs.2017.08.339
52. Velásquez N., Estevez E., Pesado P. Cloud computing, big data and the industry 4.0 reference architectures. *J. Comput. Sci. Tech.*, 2018, 18(03), e29, DOI: 10.24215/16666038.18.e29
53. Kamalakannan K., Parthipan P. Survey of Data-Centric Machine Learning Approaches in IoT-Enabled Self-Evolving Smart Environments. *2025 2nd International Conference on Electronic Circuits and Signaling Technologies (ICECST)*, 2025, DOI: 10.1109/ICECST66106.2025.11307637
54. Quintanilla F.G., Cardin O., L'Anton A., Castagna P. A modeling framework for manufacturing services in Service-oriented Holonic Manufacturing Systems. *Eng. Appl. Artif. Intell.*, 2016, 55, 26–36, DOI: 10.1016/j.engappai.2016.06.004
55. Havard V., Sahnoun M., Bettayeb B., Duval F., Baudry D. Data architecture and model design for Industry 4.0 components integration in cyber-physical production systems. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 2021, 235(14), 2338-2349, DOI: 10.1177/0954405420979463
56. Resende C., Folgado D., Oliveira J., Franco B., Moreira W., Oliveira A.Jr., Cavaleiro A., Carvalho R. TIP4.0: Industrial Internet of Things Platform for Predictive Maintenance. *Sensors*, 2021, 21, 4676, DOI: 10.3390/s21144676
57. Cardin O., Trentesaux D., Thomas A., Castagna P., Berger T., El-Haouzi H.B. Coupling predictive scheduling and reactive control in manufacturing hybrid control architectures:

- State of the art and future challenges. *J. Intell. Manuf.*, 2017, 28, 1503–1517, DOI: 10.1007/s10845-015-1139-0
58. Kozma D., Varga P., Larrinaga F. Dynamic Multilevel Workflow Management Concept for Industrial IoT Systems. *IEEE Trans. Autom. Sci. Eng*, 2021, 18, 1354–1366, DOI: 10.1109/TASE.2020.3004313
 59. Al-Gumaei K., Schuba K., Friesen A., Heymann S., Pieper C., Pethig F., Schriegel S. A survey of internet of things and big data integrated solutions for industrie 4.0. 2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA), 2018, 1, 1417-1424, DOI: 10.1109/ETFA.2018.8502484
 60. Sahal R., Breslin, J.G., Ali M.I. Big data and stream processing platforms for industry 4.0 requirements mapping for a predictive maintenance use case. *J. Manuf. Syst*, 2020, 54, 138–151, DOI: 10.1016/j.jmsy.2019.11.004
 61. Liu C., Su Z., Xu X., Lu Y. Service-oriented industrial internet of things gateway for cloud manufacturing. *Robot. Comput. -Integr. Manuf.*, 2022, 73, 102217, DOI: 10.1016/j.rcim.2021.102217
 62. Tuli S., Poojara S. R., Srirama S. N., Casale G. COSCO: Container Orchestration Using Co-Simulation and Gradient Based Optimization for Fog Computing Environments. *IEEE Transactions on Parallel and Distributed Systems*, 2021, 99, 1-1, DOI: 10.1109/TPDS.2021.3087349
 63. Nagovitsyn O.V., Lukichev S.V. Engineering geological block models as an informational framework for digital management in mining. *Journal of Mining Science*, 2025, 61(6), 1056-1065, DOI: 10.1134/S1062739125060201
 64. Анистратов К.Ю., Наговицын О.В., Наговицын Г.О., Васильева М.О. Формирование цифровой модели угольного месторождения в горно-геологической информационной системе Майнфрэйм. *Горная промышленность*, 2024, 3, 64-69, DOI: 10.30686/1609-9192-2024-3-64-69
 65. Trubetskoy K.N., Kuleshov A.A., Klebanov A.F., Vladimirov D.Ya. Modern control systems for mining and transport complexes. Saint Petersburg: Nauka, 2007, 312
 66. Villar E., Martín Toral I., Calvo I. et al. Architectures for Industrial AIoT Applications. *Sensors*, 2024, 24(15), 4929, DOI: 10.3390/s24154929
 67. Shirbazo A., Binghao Li, Ata S. et al. A Guideline for the Standardization of Smart Manufacturing and the Role of RAMI 4.0 in Digitising the Industrial Sector. *IEEE Internet of Things Journal*, 2025, 12(12), 19090-19118, DOI: 10.1109/IIOT.2025.3559929
 68. Esposito G., Romagnoli G. A Reference Model for SMEs understanding of Industry 4.0. *IFAC-PapersOnLine*. 2021, 54(1), 510-515, DOI: 10.1016/j.ifacol.2021.08.166

69. Safitri S.R., Mulyana R., Fajrillah A.A.N. Developing enterprise architecture for BPRACo SMEs digital transformation by using TOGAF 10. *Jurnal Informatika dan Komputer*, 2024, 7(3), 165-174, DOI: 10.33387/jiko.v7i3.8629
70. Afarah S.F., Hindarto D., Wahyuddin M.I. Optimizing Automotive Manufacturing Systems through TOGAF Modelling. *Sinkron*, 2024, 8(1), 414-425, DOI: 10.33395/sinkron.v9i1.13256
71. Khai Min Chew, Lee Wah Pheng. Product Life Cycle Data Management and Analytics in RAMI4.0 using the Manufacturing Chain Management Platform. *Journal of The Institution of Engineers*, 2021, 82(3), 101-108, DOI: 10.54552/v82i3.120
72. Дерябин С.А., Рзазаде У.А., Кондратьев Е.И., Темкин И.О. Мета модель архитектуры системы автономного управления транспортно-технологическими процессами в карьере. *Горный информационно-аналитический бюллетень*, 2022, 3, 117-129, DOI: 10.25018/0236_1493_2022_3_0_117
73. Kitchenham B., Brereton O.P., Budgen D., Turner M., Bailey J., Linkman S. Systematic literature reviews in software engineering-A systematic literature review. *Inf. Softw. Technol.*, 2009, 51, 7–15, DOI: 10.1016/j.infsof.2008.09.009
74. Kruchten P. The 4+1 View Model of Architecture. *IEEE Softw.*, 1995, 12, 45–50, DOI: 10.1109/52.469759
75. Vazquez-Ingelmo A., Garcia-Holgado A., Garcia-Penalvo F.J. C4 model in a software engineering subject to ease the comprehension of UML and the software. *IEEE Glob. Eng. Educ. Conf.*, 2020, 9125335, 919–924, DOI: 10.1109/EDUCON45650.2020.9125335
76. Zykov S.V., Temkin I.O., Deryabin S.A. Tradeoff-based architecting of the software system for autonomous robotized open pit mining. *Procedia Comput. Sci.*, 2019, 159, 1740–1746, DOI: 10.1016/j.procs.2019.09.345
77. Chen H.-M., Kazman R., Haziyevev S. Agile big data analytics development: An architecture-centric approach. *Proceedings of the Annual Hawaii International Conference on System Sciences*, 2016, 5378-5387, DOI: 10.1109/HICSS.2016.665
78. Tanovic A. Improvement of the eTOM standard through the comparison with ITIL V3 best practices. *Telecommunications Forum (TELFOR)*, 2012, DOI: 10.1109/TELFOR.2012.6419141
79. Zachman J. A Framework for Information Systems Architecture. *IBM Systems Journal*, 1987, 26, 276-292, DOI: 10.1147/sj.263.0276
80. Shaw M., Garlan D. An Introduction to Software Architecture. *Advances in Software Engineering and Knowledge Engineering*, 1993, 1, DOI: 10.1142/9789812798039_0001

81. Newman S. Monolith to Microservices: Evolutionary Patterns to Transform Your Monolith. Sebastopol, CA, USA: O'Reilly Media, 2019a
82. Velepucha V., Flores P. A Survey on Microservices Architecture: Principles, Patterns and Migration Challenges. *IEEE Access*, 2023, 99, DOI: 10.1109/ACCESS.2023.3305687
83. Kazman R., Klein M., Clements P. Evaluating Software Architectures for Real-Time Systems. *Annals of Software Engineering*, 1999, 7(1), 71-93, DOI: 10.1023/A:1018902725104
84. Sobhy D., Bahsoon R., Minku L., Kazman R. Evaluation of Software Architectures under Uncertainty: A Systematic Literature Review. *ACM Transactions on Software Engineering and Methodology*, 2021, 30(4), 1-50, DOI: 10.1145/3464305
85. Дерябин С.А., Темкин И.О. Онтологическое моделирование и управление цифровой трансформацией архитектуры горно-добывающих предприятий. *Записки Горного Института*, 2025, 275, 130-144
86. Zhang B., Wang X., Li H., Jiang M. Knowledge modeling of coal mining equipments based on ontology. *IOP Conference Series: Earth and Environmental Science*, 2017, 69, 012136, DOI: 10.1088/1755-1315/69/1/012136
87. Stanković R.M., Obradović I., Kitanović O., Kolonja L. Towards a Mining Equipment Ontology. 12th International Conference "Research and Development in Mechanical Industry", 14-17 September 2012, Vrnjačka Banja, Serbia. *SaTCIP*, 2012, 10 p.
88. Kolonja L., Stanković R., Obradović I. et al. Development of terminological resources for expert knowledge: a case study in mining. *Knowledge Management Research & Practice*. 2016, 14(4), 445-456, DOI: 10.1057/kmrp.2015.10
89. Vlasenko L., Lutska N., Zaiets N. et al. Core Ontology for Describing Production Equipment According to Intelligent Production. *Applied System Innovation*, 2022, 5(5), 98, DOI: 10.3390/asi5050098
90. Studer R., Benjamins V.R., Fensel D. Knowledge engineering: Principles and methods. *Data & Knowledge Engineering*, 1998, 25, 1-2, 161-197, DOI: 10.1016/S0169-023X(97)00056-6
91. Guarino N. Formal ontology, conceptual analysis and knowledge representation. *International Journal of Human-Computer Studies*, 1995, 43(5-6), 625-640, DOI: 10.1006/ijhc.1995.1066
92. Berners-Lee T., Hendler J., Lassila O. The Semantic Web: A New Form of Web Content That is Meaningful to Computers Will Unleash a Revolution of New Possibilities. *Scientific American*, 2001

93. Kleshchev A.S., Artemjeva I.L. A mathematical apparatus for domain ontology simulation. An extadable language of applied logic. *International Journal on Information Theory and Applications*, 2005, 12(2), 149-157
94. Пospelов Д.А. Прикладная семиотика и искусственный интеллект. Программные продукты и системы, 1996, 1, 3
95. Herron D., Jiménez-Ruiz E., Weyde T. On the Potential of Logic and Reasoning in Neurosymbolic Systems Using OWL-Based Knowledge Graphs. *Neurosymbolic Artificial Intelligence*, 2025, 1, 15 p., DOI: 10.1177/29498732251320043
96. Chen J., Hu P., Jimenez-Ruiz E. et al. OWL2Vec*: embedding of OWL ontologies. *Machine Learning*, 2021, 110(7), 1813-1845, DOI: 10.1007/s10994-021-05997-6
97. Baader F., Calvanese D., McGuinness D., Nardi D., Patel-Schneider P. *The Description Logic Handbook*. Cambridge University Press, 2003
98. Motik B., Shearer R., Horrocks I. Hypertableau Reasoning for Description Logics. *Journal of Artificial Intelligence Research*, 36, 165-228, DOI: 10.1613/jair.2811
99. Stevens W.P., Myers G.L., Constantive L.L. Structured design. *IBM Systems Journal*, 1999, 38(2.3), 231-256, DOI: 10.1147/sj.382.0231
100. Yourdon E., Constantine L. *Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design*, Prentice Hall, 1979
101. Chidamber S.R., Kemerer C.F. A Metrics Suite for Object-Oriented Design. *IEEE Trans. Soft-ware Eng.*, 1994, 20(6), 476-493, DOI: 10.1109/32.295895
102. Basili V.R., Briand L.C., Melo W.L. A Validation of Object-Oriented Design Metrics as Quality Indicators. *IEEE Transactions on Software Engineering*, 1996, 22(10), 751-761, DOI: 10.1109/32.544352
103. McCabe T.J. A Complexity Measure. *IEEE Transactions on Software Engineering*, 1976, SE-2(4), 308-320, DOI: 10.1109/TSE.1976.233837
104. Halstead M.H. *Elements of Software Science*. Elsevier, 1977, 127
105. Shen V.Y., Conte S.D., Dunsmore H.E. Software Science Revisited: A Critical Analysis of the Theory and Its Empirical Support. *IEEE Transactions on Software Engineering*, 1983, SE-9(2), 155-165, DOI: 10.1109/TSE.1983.236460
106. Shannon C.E. A Mathematical Theory of Communication. *The Bell System Technical Journal*, 1948, 27(3), 379-423, DOI: 10.1002/j.1538-7305.1948.tb01338.x
107. Rashevsky N. Life, information theory and topology. *Bull. Math. Biophys.*, 1955, 17, 229–235, DOI: 10.1007/BF02477860

- 108.Mowshowitz A. Entropy and the complexity of the graphs I: an index of the relative complexity of a graph. *Bull. Math. Biophys.*, 1968, 30, 175–204, DOI: 10.1007/BF02476948
- 109.Mowshowitz A. Entropy and the complexity of graphs II: the information content of digraphs and infinite graphs. *Bull. Math. Biophys.*, 1968, 30, 225–240, DOI: 10.1007/BF02476692
- 110.Mowshowitz A. Entropy and the complexity of graphs III: graphs with prescribed information content. *Bull. Math. Biophys.*, 1968, 30, 387–414, DOI: 10.1007/BF02476603
- 111.Mowshowitz A. Entropy and the complexity of graphs IV: entropy measures and graphical structure. *Bull. Math. Biophys.*, 1968, 30, 533–546, DOI: 10.1007/BF02476673
- 112.Dehmer M., Mowshowitz A. A History of Graph Entropy Measures, *Information Sciences*, 2011, 1(1), 57-78, DOI: 10.1016/j.ins.2010.08.041
- 113.Dehmer M., Barbarini N., Varmuza K., Graber A. A large scale analysis of information-theoretic network complexity measures using chemical structures. *PLOS ONE*, 2009, 4(12), e8057, DOI: 10.1371/journal.pone.0008057
- 114.Dehmer M., Mowshowitz A. Generalized graph entropies. *Complexity*, 2011, 17, 45–50, DOI: 10.1002/cplx.20379
- 115.Dehmer M., Barbarini N., Varmuza K., Graber A. A large scale analysis of information-theoretic network complexity measures using chemical structures. *PLOS ONE*, 2009, 4(12), e8057, DOI: 10.1371/journal.pone.0008057
- 116.Belady L., Lehman M. A Model of Large Program Development. *IBM Systems Journal*, 1976, 1, 13440611, DOI: 10.1147/sj.153.0225
- 117.Hassan A.E. Predicting faults using the complexity of code changes. *Proceedings - International Conference on Software Engineering*, 2009, DOI: 10.1109/ICSE.2009.5070510
- 118.Bunke H., Allermann G. Inexact graph matching for structural pattern recognition. *Pattern Recognition Letters*, 1983, 1(4), 245-253, DOI: 10.1016/0167-8655(83)90033-8
- 119.Bunke H. On a relation between graph edit distance and maximum common subgrap. *Pattern Recognition Letters*, 1997, 18(8), 689-694, DOI: 10.1016/S0167-8655(97)00060-3
- 120.Sanfeliu A., Fu K. A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man and Cybernetisc*, 1983, 1087693, DOI: 10.1109/TSMC.1983.6313167
- 121.Zeng Z., Tung A.K.H., Wang J., Feng J., Zhou L. Comparing stars: On approximating graph edit distance. *PVLDB*, 2009, 2(1), 25–36, DOI: 10.14778/1687627.1687631

122. Zheng W., Zou L., Lian X., Wang D. Efficient Graph Similarity Search Over Large Graph Databases. *IEEE Transactions on Knowledge and Data Engineering*, 2015, 27(4), 964-978, DOI: 10.1109/TKDE.2014.2349924
123. Zheng W., Zou L., Lian X., Wang D., Zhao D. Graph Similarity Search with Edit Distance Constraint in Large Graph Databases. *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management (CIKM '13)*, 2013, 1595-1600, DOI: 10.1145/2505515.2505723
124. Blumenthal D., B., Gamper J. On the exact computation of the graph edit distance. *Pattern Recognition Letters*, 2020, 134, 46-57, DOI: 10.1016/j.patrec.2018.05.002
125. Reisen k., Ferrer M., Fischer A., Bunke H. Approximation of Graph Edit Distance in Quadratic Time. *Lecture Notes in Computer Science*, 2015, 9069, 3-12, DOI: 10.1007/978-3-319-18224-7_1
126. Stary C. Digital Twin Generation: Re-Conceptualizing Agent Systems for Behavior-Centered Cyber-Physical System Development. *Sensors*, 2021, 21, 1096, DOI: 10.3390/s21041096
127. Savolainen J., Urbani M. Maintenance optimization for a multi-unit system with digital twin simulation. *Intell. Manuf.*, 2021, 32, 1953–1973, DOI: 10.1007/s10845-021-01740-z
128. Laryukhin V., Skobelev P., Lakhin O., Grachev S., Yalovenko V., Yalovenko O. The multi-agent approach for developing a cyber-physical system for managing precise farms with digital twins of plants. *Cybern. Phys.*, 2019, 8, 257–261, DOI: 10.1016/j.compind.2018.08.004
129. Kostromin R., Feoktistov A. Agent-based DevOps of software and hardware resources for digital twins of infrastructural objects. In *Proceedings of the 4th International Conference on Future Networks and Distributed Systems (ICFNDS)*, St. Petersburg, Russia, 26–27 November 2020, 1–6, DOI: 10.1145/3440749.3442599
130. Jung T., Shah P., Weyrich M. Dynamic Co-Simulation of Internet-of-Things-Components using a Multi-Agent-System. *Procedia CIRP*, 2018, 72, 874–879, DOI: 10.1016/j.procir.2018.03.084
131. Latsou C., Farsi M., Erkoyuncu J.A., Morris G. Digital twin integration in multi-agent cyber physical manufacturing systems. *FAC-Pap.*, 2021, 54, 811–816, DOI: 10.1016/j.ifacol.2021.08.096
132. Волкова Е.А., Дружинин А.В., Полузадов В.Н., Копанев А.А., Управление горным предприятием на основе мультиагентного подхода. *Универсум: технические науки*, 2016, 10(31)

133. Temkin I., Myaskov A., Deryabin S., Konov I., Ivannikov A. Design of a Digital 3D Model of Transport–Technological Environment of Open-Pit Mines Based on the Common Use of Telemetric and Geospatial Information. *Sensors*, 2021, 21(18), 6277, DOI: 10.3390/s21186277
134. Темкин И.О., Куляница А.Л., Дерябин С.А. Вычислительные модели взаимодействия автономных мобильных агентов транспортного комплекса горных предприятий. *Информация и космос*, 2017, 2, 65-71
135. Темкин И.О., Дерябин С.А., Конов И.С. Нечеткие модели управления взаимодействием мобильных объектов горно-транспортного комплекса (ГТК). Шаг в будущее: Искусственный Интеллект и Цифровая Экономика, 2017, 1, 246-253
136. Федунец Н.И., Приходько М.А. Алгоритмы формирования управляющих воздействий в распределенных мультиагентных системах. программные продукты и системы, 2010, 4, 18
137. Федунец Н.И., Кубрин С.С. Перспективы и проблемы построения автоматизированных радиотелеметрических систем управления технологическими процессами в шахтах и рудниках. Горный информационно-аналитический бюллетень, 2010, S5, 137-142
138. Федунец Н.И., Гончаренко А.Н. Структурно-параметрический синтез оптимальной архитектуры программно-аппаратных средств на промышленном предприятии. Прикаспийский журнал: управление и высокие технологии, 2012, 1(17), 152-158
139. Федунец Н.И., Гончаренко А.Н. Инновационный подход к определению структуры программных решений для бизнес-процессов промышленного предприятия. Программные продукты и системы, 2012, 1, 18
140. Ahmad A., Babar M.A. Software architectures for robotic systems: A systematic mapping study. *Journal of Systems and Software*, 2016, 122, 16-39, DOI: 10.1016/j.jss.2016.08.039
141. Chen D., Feng L., Qureshi T.N., Lönn H., Hagl F. An architectural approach to the analysis, verification and validation of software intensive embedded systems. *Computing*, 2013, 95(8), 649-688, DOI: 10.1007/s00607-013-0314-4
142. Patel J., Lee R., Kim H.-K. Architectural View in Software Development Life-Cycle Practices. 6th IEEE/ACIS International Conference on Computer and Information Science (ICIS 2007), 2007, DOI: 10.1109/ICIS.2007.64

ПРИЛОЖЕНИЯ

Приложение А.

Листинг исходного кода алгоритма формирования метамодели архитектуры автономно функционирующей кибер-физической геопространственной индустриальной системы

```
import sys
from pathlib import Path
import xml.etree.ElementTree as ET
import pandas as pd
import numpy as np

INPUT_FILE = Path(__file__).resolve().parent.parent / "input" / "inferred"
OUTPUT_DIR = Path(__file__).resolve().parent.parent / "output"

SERVER_CODE = "000001"
NS_RDF = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
NS_OWL = "http://www.w3.org/2002/07/owl#"
NS_RDFS = "http://www.w3.org/2000/01/rdf-schema#"

GPR_ADJ = {"ОперативныйГПР": "Оперативное", "ТактическийГПР": "Тактическое",
           "СтратегическийГПР": "Стратегическое"}
COUNT_ADJ = {"Экземпляр": "индивидуальное", "Группа": "групповое"}
STATE_NAME = {
    "Местоположение": "местоположение", "Положение": "положение",
    "Производительность": "производительность", "Массогабаритность":
"массогабаритность",
    "Вместимость": "вместимость", "ЭнергоОбъем": "энергообъем",
    "Энергоемкость": "энергоемкость", "Работоспособность": "работоспособность",
    "Консистенция": "консистенция", "Конфигурация": "конфигурация",
    "Целостность": "целостность", "ЭлектрическийТок": "электрический ток",
    "Стоимость": "стоимость",
}
ENTITY_NAME = {
    "Самосвал": "самосвал", "Экскаватор": "экскаватор", "Погрузчик": "погрузчик",
    "Грейдер": "грейдер", "БузовзрывнаяМашина": "бузовзрывная машина",
    "Топливозаправщик": "топливозаправщик", "Конвейер": "конвейер",
    "Дорога": "дорога", "ЗонаПогрузки": "зона погрузки",
    "ЗонаРазгрузки": "зона разгрузки", "ЗонаБВР": "зона БВР",
    "Гараж": "гараж", "Склад": "склад",
    "ГорнаяПорода": "горная порода", "ДорожноеПокрытие": "дорожное покрытие",
    "Скважина": "скважина", "ДизельноеТопливо": "дизельное топливо",
    "Электричество": "электричество", "Взрывчатка": "взрывчатка", "Вода": "вода",
    "Ковш": "ковш", "Кузов": "кузов", "Лента": "лента", "Бур": "бур",
    "ОтвалСНожом": "отвал с ножом", "Цистерна": "цистерна",
    "ДизельныйДвигатель": "дизельный двигатель",
    "ЭлектрическийДвигатель": "электрический двигатель",
    "Платформа": "платформа", "КолесноеШасси": "колесное шасси",
    "ГусеничноеШасси": "гусеничное шасси",
}
```



```

INTER_NAME = {
    "Движение": "движение", "Доставка": "доставка", "Перемещение": "перемещение",
    "Размещение": "размещение", "Вмещение": "вмещение", "Расположение":
"расположение",
    "Обработка": "обработка", "Изготовление": "изготовление",
    "Изнашивание": "изнашивание", "Потребление": "потребление",
    "Расходование": "расходование",
}
FUNC_NAME_LC = {
    "Детектирование": "детектирование", "Идентификация": "идентификация",
    "Прогнозирование": "прогнозирование", "Планирование": "планирование",
    "Управление": "управление",
}
STATE_CODE = {
    "Местоположение": "1001", "Положение": "1002", "Производительность": "2001",
    "Массогабаритность": "2002", "Вместимость": "2003", "ЭнергоОбъем": "2004",
    "Энергоемкость": "2005", "Работоспособность": "3001", "Консистенция": "3002",
    "Конфигурация": "3003", "Целостность": "3004", "ЭлектрическийТок": "3005",
    "Стоимость": "4001",
}
TIME_CODE = {"ИсторическийМомент": 1, "ТекущийМомент": 2,
    "ПрогнозныйМомент": 3, "ПлановыйМомент": 4, "ФиктивныйМомент": 5}
GPR_CODE = {"ОперативныйГПР": 1, "ТактическийГПР": 2, "СтратегическийГПР": 3}
COUNT_CODE = {"Экземпляр": 1, "Группа": 2}
FUNC_CODE = {"Детектирование": 1, "Идентификация": 2, "Прогнозирование": 3,
    "Планирование": 4, "Управление": 5}
INTER_CODE = {"Доставка": 1, "Движение": 2, "Перемещение": 3, "Размещение": 4,
    "Вмещение": 5, "Расположение": 6, "Обработка": 7, "Изготовление": 8,
    "Изнашивание": 9, "Потребление": 10, "Расходование": 11}
PHYS_CODE = {
    "Самосвал": "110001", "Экскаватор": "110002", "Погрузчик": "110003",
    "Грейдер": "110004", "БуровзрывнаяМашина": "110005",
    "Топливозаправщик": "110006", "Конвейер": "120001",
    "Дорога": "210001", "ЗонаПогрузки": "210002", "ЗонаРазгрузки": "210003",
    "ЗонаБВР": "210004", "Гараж": "220001", "Склад": "220002",
    "ГорнаяПорода": "410001", "ДорожноеПокрытие": "420001", "Скважина": "420002",
    "ДизельноеТопливо": "510001", "Электричество": "510002",
    "Взрывчатка": "520001", "Вода": "520002",
    "Ковш": "610001", "Кузов": "610002", "Лента": "610003", "Бур": "610004",
    "ОтвалСНожом": "610005", "Цистерна": "610006",
    "ДизельныйДвигатель": "620001", "ЭлектрическийДвигатель": "620002",
    "Платформа": "630001", "КолесноеШасси": "630002", "ГусеничноеШасси": "630003",
}
_PROP_MAP = {
    "ИзменяетПространственноеCO": ("CO", "пространственное"),
    "ИзменяетПространственноеСС": ("СС", "пространственное"),
    "ИзменяетТехнологическоеCO": ("CO", "технологическое"),
    "ИзменяетТехнологическоеСС": ("СС", "технологическое"),
    "ИзменяетЭксплуатационноеCO": ("CO", "эксплуатационное"),
    "ИзменяетЭксплуатационноеСС": ("СС", "эксплуатационное"),
    "ИзменяетЭкономическоеCO": ("CO", "экономическое"),
    "ИзменяетЭкономическоеСС": ("СС", "экономическое"),
}

```

```

def _ln(uri):
    if not uri: return ""
    if "}" in uri: return uri.rsplit("}", 1)[-1]
    if "#" in uri: return uri.rsplit("#", 1)[-1]
    return uri.rsplit("/", 1)[-1]

def parse_owl(path):
    tree = ET.parse(path)
    root = tree.getroot()
    individuals = {}
    for ind in root.findall(f"{{{NS_OWL}}}NamedIndividual"):
        name = _ln(ind.get(f"{{{NS_RDF}}}about", ""))
        if not name: continue
        types = set()
        props = {}
        for child in ind:
            tag = _ln(child.tag)
            res = _ln(child.get(f"{{{NS_RDF}}}resource", ""))
            if not res: continue
            if tag == "type":
                if res != "NamedIndividual": types.add(res)
            else:
                props.setdefault(tag, []).append(res)
        individuals[name] = {"types": types, "props": props}
    class_hierarchy = {}
    has_value_restrictions = []
    for cls in root.findall(f"{{{NS_OWL}}}Class"):
        cls_name = _ln(cls.get(f"{{{NS_RDF}}}about", ""))
        if not cls_name: continue
        for sub in cls.findall(f"{{{NS_RDFS}}}subClassOf"):
            parent_ref = sub.get(f"{{{NS_RDF}}}resource", "")
            if parent_ref:
                class_hierarchy.setdefault(cls_name, set()).add(_ln(parent_ref))
            continue
        restriction = sub.find(f"{{{NS_OWL}}}Restriction")
        if restriction is None: continue
        on_prop = restriction.find(f"{{{NS_OWL}}}onProperty")
        has_val = restriction.find(f"{{{NS_OWL}}}hasValue")
        if on_prop is not None and has_val is not None:
            prop_name = _ln(on_prop.get(f"{{{NS_RDF}}}resource", ""))
            val_name = _ln(has_val.get(f"{{{NS_RDF}}}resource", ""))
            if prop_name and val_name:
                has_value_restrictions.append((cls_name, prop_name, val_name))
    return individuals, class_hierarchy, has_value_restrictions

def get_all_subclasses(cls_name, hierarchy):
    result = {cls_name}
    children = [c for c, parents in hierarchy.items() if cls_name in parents]
    for child in children:

```

```

        result |= get_all_subclasses(child, hierarchy)
    return result

def individuals_of_class(cls_name, hierarchy, individuals):
    all_classes = get_all_subclasses(cls_name, hierarchy)
    result = set()
    for ind_name, ind_data in individuals.items():
        if ind_data["types"] & all_classes:
            result.add(ind_name)
    return sorted(result)

def _is_valid_pair(subj, inter, obj, individuals):
    s_props = individuals.get(subj, {}).get("props", {})
    o_props = individuals.get(obj, {}).get("props", {})
    s_types = individuals.get(subj, {}).get("types", set())
    o_types = individuals.get(obj, {}).get("types", set())
    explicit_inter = o_props.get("ЯвляетсяОбъектомВзаимодействия", [])
    if explicit_inter and inter not in explicit_inter:
        return False
    if "УстройствоТехнологическогоВзаимодействия" in s_types or \
        any(t.startswith("УТВС") for t in s_types):
        targets = s_props.get("Обрабатывает", []) + s_props.get("Содержит", [])
        if targets and obj not in targets:
            return False
    if "УстройствоЭксплуатационногоВзаимодействия" in s_types:
        targets = s_props.get("Потребляет", [])
        if targets and obj not in targets:
            return False
    is_equipment = s_types & {"Оборудование", "МобильноеОборудование",
                              "СтационарноеОборудование"}
    obj_is_prod_res = o_types & {"Продукт", "ПриродныйПродукт",
                              "ГеотехническийПродукт",
                              "ТехническийПродукт", "Ресурс",
                              "ЭнергетическийРесурс",
                              "ИнструментальныйРесурс"}
    if is_equipment and obj_is_prod_res:
        produced = s_props.get("Производит", [])
        if produced and obj not in produced:
            return False
    if ("Устройство" in o_types or
        "УстройствоПространственногоВзаимодействия" in o_types or
        "УстройствоТехнологическогоВзаимодействия" in o_types or
        "УстройствоЭксплуатационногоВзаимодействия" in o_types or
        any(t.startswith("УПВ") or t.startswith("УТВС") for t in o_types)):
        own = (s_props.get("СостоитИзПространственногоВзаимодействия", []) +
              s_props.get("СостоитИзТехнологическогоВзаимодействия", []) +
              s_props.get("СостоитИзЭксплуатационногоВзаимодействия", []))
        if own and obj not in own:
            return False
    if s_types & {"Инфраструктура", "ГеопространственнаяИнфраструктура",
                  "ТехническаяИнфраструктура"}:

```

```

        placed = s_props.get("Размещает", [])
        if placed and obj not in placed:
            return False
    if o_types & {"Инфраструктура", "ГеопространственнаяИнфраструктура",
"ТехническаяИнфраструктура"}:
        hosts = o_props.get("Размещает", [])
        if hosts and subj not in hosts:
            return False
    return True

def infer_triples(individuals, hierarchy, restrictions):
    ind_subj, ind_obj = {}, {}
    for name, data in individuals.items():
        for val in data["props"].get("ЯвляетсяСубъектомВзаимодействия", []):
            ind_subj.setdefault(val, set()).add(name)
        for val in data["props"].get("ЯвляетсяОбъектомВзаимодействия", []):
            ind_obj.setdefault(val, set()).add(name)
    cls_subj, cls_obj = {}, {}
    for cls_name, prop_name, val_name in restrictions:
        if prop_name == "ЯвляетсяСубъектомВзаимодействия":
            cls_subj.setdefault(val_name, set()).add(cls_name)
        elif prop_name == "ЯвляетсяОбъектомВзаимодействия":
            cls_obj.setdefault(val_name, set()).add(cls_name)
    all_interactions = set(cls_subj) | set(cls_obj) | set(ind_subj) | set(ind_obj)
    triples = []
    for interaction in sorted(all_interactions):
        subjects = set()
        for cls in cls_subj.get(interaction, set()):
            subjects.update(individuals_of_class(cls, hierarchy, individuals))
        subjects.update(ind_subj.get(interaction, set()))
        objects = set()
        for cls in cls_obj.get(interaction, set()):
            objects.update(individuals_of_class(cls, hierarchy, individuals))
        objects.update(ind_obj.get(interaction, set()))
        for s in sorted(subjects):
            for o in sorted(objects):
                if s != o and _is_valid_pair(s, interaction, o, individuals):
                    triples.append((s, interaction, o))
    return triples

def get_states(individuals, interaction):
    result = {"C0": [], "CC": []}
    props = individuals[interaction]["props"]
    for prop_name, (role, _) in _PROP_MAP.items():
        for val in props.get(prop_name, []):
            if val not in result[role]:
                result[role].append(val)
    return result

def get_func_times(individuals):

```

```

result = {}
funcs = [n for n, d in individuals.items()
          if "Функция" in d["types"]
          and "ГоризонтПринятияРешения" not in d["types"]
          and "КоличествоОбъектовУправления" not in d["types"]]
for f in funcs:
    props = individuals[f]["props"]
    inp = props.get("ИспользуетКакВходнойПараметр", [])
    out = props.get("ИспользуетКакВыходнойПараметр", [])
    result[f] = (inp, out[0] if out else "")
return result

def _build_device_to_equipment(individuals):
    equip_types = {"Оборудование", "МобильноеОборудование",
                  "СтационарноеОборудование"}
    result = {}
    for name, data in individuals.items():
        if not (data["types"] & equip_types): continue
        for prop in ["СостоитИзПространственногоВзаимодействия",
                     "СостоитИзТехнологическогоВзаимодействия",
                     "СостоитИзЭксплуатационногоВзаимодействия"]:
            for device in data["props"].get(prop, []):
                result[device] = name
    return result

def build_description(gpr, count, func, out_states, out_entity, inter, other):
    gpr_a = GPR_ADJ.get(gpr, gpr)
    cnt_a = COUNT_ADJ.get(count, count)
    func_l = FUNC_NAME_LC.get(func, func)
    states_s = " и ".join(STATE_NAME.get(s, s) for s in out_states) if out_states else ""
    return (f"{gpr_a} {cnt_a} {func_l} "
            f"состояний {states_s} {ENTITY_NAME.get(out_entity, out_entity)} "
            f"при {INTER_NAME.get(inter, inter)} {ENTITY_NAME.get(other, other)}")

def generate_catalog(individuals, func_times, triples, gprs, counts):
    active_funcs = [f for f in func_times if f != "Моделирование"]
    device_to Equip = _build_device_to_equipment(individuals)
    rows = []
    phi_n = 1
    for subj, inter, obj in triples:
        states = get_states(individuals, inter)
        subj_states, obj_states = states["CC"], states["C0"]
        for func in active_funcs:
            for gpr in gprs:
                for count in counts:
                    for out_idx, (out_role, out_st, out_ent, other_ent) in enumerate([
                        ("субъект", subj_states, subj, obj),
                        ("объект", obj_states, obj, subj),
                    ], start=1):

```

```

inp_times, out_time = func_times[func]
gpr_code = GPR_CODE[gpr]
subj_code = PHYS_CODE.get(subj, "")
owner = device_to Equip.get(subj)
effective_code = subj_code
if owner and not subj_code.startswith("1"):
    effective_code = PHYS_CODE.get(owner, "")
is_equip_or_device = effective_code.startswith("1")
if gpr_code == 1 and is_equip_or_device:
    node, placement = effective_code, 1
elif gpr_code in (2, 3) and is_equip_or_device:
    node, placement = effective_code, 2
else:
    node, placement = SERVER_CODE, 1
rows.append({
    "phi": f"φ{phi_n}",
    "description": build_description(gpr, count, func, out_st,
out_ent, inter, other_ent),
    "gpr": gpr_code, "phys_obj_count": COUNT_CODE[count],
    "function_type": FUNC_CODE[func],
    "interaction": INTER_CODE.get(inter, inter),
    "subject": PHYS_CODE.get(subj, subj),
    "object": PHYS_CODE.get(obj, obj),
    "input_states_subject": ";".join(STATE_CODE.get(s, s) for
s in subj_states),
    "input_states_object": ";".join(STATE_CODE.get(s, s) for s
in obj_states),
    "input_time": ";".join(str(TIME_CODE.get(t, t)) for t in
inp_times),
    "output_state": out_idx,
    "output_time": TIME_CODE.get(out_time, ""),
    "node": node, "placement": placement,
    "agent": "", "integrated": 0,
    "субъект": subj, "взаимодействие": inter, "объект": obj,
    "функция": func, "горизонт": gpr, "количество": count,
    "выходная_роль": out_role, "выходные_состояния": ";
".join(out_st),
    })
    phi_n += 1
rows.append({"phi": f"φ{phi_n}", "description": "Интеграция систем и функций",
    "gpr": "", "phys_obj_count": "", "function_type": "", "interaction":
    "",
    "subject": "", "object": "", "input_states_subject": "",
    "input_states_object": "", "input_time": "", "output_state": "",
    "output_time": "", "node": SERVER_CODE, "placement": 1,
    "agent": "", "integrated": 1,
    "субъект": "", "взаимодействие": "", "объект": "",
    "функция": "Интеграция", "горизонт": "", "количество": "",
    "выходная_роль": "", "выходные_состояния": ""})
return pd.DataFrame(rows)

if __name__ == "__main__":

```

```

individuals, hierarchy, restrictions = parse_owl(INPUT_FILE)
triples = infer_triples(individuals, hierarchy, restrictions)
func_times = get_func_times(individuals)
gprs = sorted([n for n, d in individuals.items() if "ГоризонтПринятияРешения" in
d["types"]])
counts = sorted([n for n, d in individuals.items() if
"КоличествоОбъектовУправления" in d["types"]])
phi = generate_catalog(individuals, func_times, triples, gprs, counts)
n_main = len(phi[phi.integrated != 1])
print(f"Каталог Φ: {len(phi)} функций ({n_main} основных + 1 интеграционная)")
print(f"Троек: {len(triples)}")
nodes = phi[phi.node != ""].value_counts().sort_index()
print(f"Каталог Ξ: {len(nodes)} физических узлов")
for node, cnt in nodes.items():
    print(f"    {node}: {cnt}")
out = OUTPUT_DIR / "algorithm_1_metamodel.xlsx"
main_cols = [c for c in phi.columns if c not in
              {"субъект", "взаимодействие", "объект", "функция",
               "горизонт", "количество", "выходная_роль", "выходные_состояния"}]
phi[main_cols].to_excel(out, sheet_name="Каталог Φ", index=False)
print(f"Сохранено: {out}")

```

Приложение Б.

Листинг исходного кода алгоритма конфигурирования и анализа архитектуры «as is»

```
import math
import sys
from pathlib import Path
import numpy as np
import pandas as pd

OUTPUT_DIR = Path(__file__).resolve().parent

SERVER_CODE = "000001"
CLASS_COLUMNS = ["x^lp", "x^sp", "x^lb", "x^sb", "x^lc", "x^sc", "x^ls", "x^ss"]
LOCAL_CLASSES = ["x^lp", "x^lb", "x^lc", "x^ls"]
SERVER_CLASSES = ["x^sp", "x^sb", "x^sc", "x^ss"]
CLASS_IDS = {"x^lp": 1, "x^sp": 2, "x^lb": 3, "x^sb": 4,
             "x^lc": 5, "x^sc": 6, "x^ls": 7, "x^ss": 8}
MERGE_AGENTS = {"0001", "0002", "0003"}
MERGE_COLUMNS = ["x^lp", "x^sp", "x^lb", "x^sb", "x^ls", "x^ss"]
CLASS_PAIRS = {"p": ("x^lp", "x^sp"), "b": ("x^lb", "x^sb"),
               "c": ("x^lc", "x^sc"), "s": ("x^ls", "x^ss")}

def classify_index(idx):
    return {1:"p", 2:"p", 3:"b", 4:"b", 5:"c", 6:"c", 7:"s", 8:"s"}.get(idx // 100000,
    "?")

def build_initial_population(phi):
    matrix = pd.DataFrame(0, index=phi["phi"], columns=CLASS_COLUMNS)
    matrix.index.name = "phi"
    for _, row in phi.iterrows():
        p = row["phi"]
        node = row["node_str"]
        placement = int(row["placement"])
        if node == SERVER_CODE:
            for c in SERVER_CLASSES:
                matrix.loc[p, c] = 1
        elif placement == 1:
            for c in LOCAL_CLASSES:
                matrix.loc[p, c] = 1
        elif placement == 2:
            for c in CLASS_COLUMNS:
                matrix.loc[p, c] = 1
    return matrix

def build_reduced_population(phi):
    matrix = pd.DataFrame(0, index=phi["phi"], columns=CLASS_COLUMNS)
    matrix.index.name = "phi"
    for _, row in phi.iterrows():
```



```

p = row["phi"]
node = row["node_str"]
placement = int(row["placement"])
is_local = node != SERVER_CODE
is_hard_bound = is_local and placement == 1
if is_hard_bound:
    matrix.loc[p, "x^lc"] = 1
else:
    matrix.loc[p, "x^sc"] = 1
if matrix.loc[p, "x^lc"] == 1:
    matrix.loc[p, "x^lb"] = 1
    matrix.loc[p, "x^ls"] = 1
if matrix.loc[p, "x^sc"] == 1:
    matrix.loc[p, "x^sb"] = 1
    matrix.loc[p, "x^ss"] = 1
if is_local:
    matrix.loc[p, "x^lp"] = 1
else:
    matrix.loc[p, "x^sp"] = 1
return matrix

def build_indexed_classes(reduced):
    indexed = pd.DataFrame(0, index=reduced.index, columns=reduced.columns)
    indexed.index.name = reduced.index.name
    for col, class_id in CLASS_IDS.items():
        counter = 1
        for p in reduced.index:
            if reduced.loc[p, col] == 1:
                indexed.loc[p, col] = class_id * 100000 + counter
                counter += 1
    return indexed

def build_merged_classes(indexed, phi):
    phi_info = {}
    for _, row in phi.iterrows():
        phi_info[row["phi"]] = (str(row["agent"]), row["node_str"])
    merged = indexed.copy()
    for col in MERGE_COLUMNS:
        groups = {}
        for p in merged.index:
            if merged.loc[p, col] != 0:
                agent, node = phi_info[p]
                actual_node = SERVER_CODE if col.startswith("x^s") else node
                groups.setdefault((agent, actual_node), []).append(p)
        for (agent, _), phis in groups.items():
            if agent in MERGE_AGENTS and len(phis) > 1:
                min_idx = min(merged.loc[p, col] for p in phis)
                for p in phis:
                    merged.loc[p, col] = min_idx
    return merged

```

```

def build_architecture_graph(merged):
    all_indices = set()
    for col in merged.columns:
        all_indices.update(merged[col][merged[col] != 0].unique())
    all_indices = sorted(all_indices)
    idx_to_pos = {idx: i for i, idx in enumerate(all_indices)}
    n = len(all_indices)
    matrix = np.zeros((n, n), dtype=int)
    for phi in merged.index:
        classes = {}
        for ct, (lc, sc) in CLASS_PAIRS.items():
            lv, sv = int(merged.loc[phi, lc]), int(merged.loc[phi, sc])
            classes[ct] = lv if lv != 0 else sv
        p, b, c, s = idx_to_pos[classes["p"]], idx_to_pos[classes["b"]], \
            idx_to_pos[classes["c"]], idx_to_pos[classes["s"]]
        matrix[p, b] = matrix[b, p] = 1
        matrix[b, c] = matrix[c, b] = 1
        matrix[b, s] = matrix[s, b] = 1
    return matrix, all_indices

def compute_H(matrix):
    degrees = matrix.sum(axis=1)
    return sum(math.log2(d) for d in degrees if d > 1)

if __name__ == "__main__":
    phi = pd.read_excel(OUTPUT_DIR / "task0_ontology_inference.xlsx",
        sheet_name="Каталог Φ")
    phi = phi[phi["integrated"] != 1].copy()
    phi["_sort"] = phi["phi"].str.replace("φ", "", regex=False).astype(int)
    phi = phi.sort_values("_sort").reset_index(drop=True)
    fa = pd.read_csv(OUTPUT_DIR / "task0_adjacency_phi_agents.csv", index_col=0)
    agent_codes = ["0001", "0002", "0003", "1001", "1002", "1003", "1004",
        "1005", "1006", "1007", "1008", "1009", "1010", "1011"]
    phi_agents = {}
    for i in range(len(fa)):
        for j, code in enumerate(agent_codes):
            if fa.iloc[i].iloc[j] == 1:
                idx = fa.index[i]
                phi_agents[idx if isinstance(idx, str) else f"φ{idx}"] = code
                break
    phi["agent"] = phi["phi"].map(phi_agents)
    phi["node_str"] = phi["node"].apply(lambda x: str(int(x)).zfill(6))

    initial = build_initial_population(phi)
    print(f"Начальная популяция: {int(initial.values.sum())} классов")

    reduced = build_reduced_population(phi)
    print(f"После сокращения: {int(reduced.values.sum())} классов")

    indexed = build_indexed_classes(reduced)
    merged = build_merged_classes(indexed, phi)

```

```

total_before = sum(indexed[col][indexed[col] != 0].nunique() for col in
indexed.columns)
total_after = sum(merged[col][merged[col] != 0].nunique() for col in
merged.columns)
print(f"После слияния: {total_after} уникальных (было {total_before})")

matrix, all_indices = build_architecture_graph(merged)
n_vertices = len(all_indices)
n_edges = int(matrix.sum()) // 2
H = compute_H(matrix)
print(f"Граф  $G(A^0)$ : {n_vertices} вершин, {n_edges} рёбер, H = {H:.4f} бит")

```

Приложение В.

Листинг исходного кода алгоритма конфигурирования и анализа архитектуры «to be»

```
import math
import sys
from pathlib import Path
import numpy as np
import pandas as pd

OUTPUT_DIR = Path(__file__).resolve().parent

SERVER_CODE = "000001"
CLASS_PAIRS = {"p": ("x^lp", "x^sp"), "b": ("x^lb", "x^sb"),
               "c": ("x^lc", "x^sc"), "s": ("x^ls", "x^ss")}
MERGE_COLUMNS = ["x^lp", "x^sp", "x^lb", "x^sb", "x^ls", "x^ss"]
PERSONNEL = {"1001", "1002", "1003", "1004", "1005", "1006", "1007",
             "1008", "1009", "1010", "1011"}

def classify_index(idx):
    return {1:"p",2:"p",3:"b",4:"b",5:"c",6:"c",7:"s",8:"s"}.get(idx//100000,"?")

def get_phi_classes(merged, phi):
    cl = {}
    for ct, (lc, sc) in CLASS_PAIRS.items():
        lv, sv = int(merged.loc[phi, lc]), int(merged.loc[phi, sc])
        cl[ct] = lv if lv != 0 else sv
    return cl

def compute_H(matrix):
    degrees = matrix.sum(axis=1)
    return sum(math.log2(d) for d in degrees if d > 1)

def rebuild_merged(indexed, phi_info, merge_agents):
    merged = indexed.copy()
    for col in MERGE_COLUMNS:
        groups = {}
        for phi in merged.index:
            if merged.loc[phi, col] != 0:
                agent = phi_info.get(phi, ("?", "?"))[0]
                node = phi_info.get(phi, ("?", "?"))[1]
                actual_node = SERVER_CODE if col.startswith("x^s") else node
                groups.setdefault((agent, actual_node), []).append(phi)
        for (agent, _), phis in groups.items():
            if agent in merge_agents and len(phis) > 1:
                min_idx = min(merged.loc[p, col] for p in phis)
                for p in phis:
                    merged.loc[p, col] = min_idx
```

```

return merged

def build_graph(merged, phi_info, integration_agents=None):
    all_indices = set()
    for col in merged.columns:
        all_indices.update(merged[col][merged[col] != 0].unique())
    max_idx = max(all_indices)
    integ_p, integ_b = max_idx + 1, max_idx + 2
    integ_c, integ_s = max_idx + 3, max_idx + 4
    all_indices.update({integ_p, integ_b, integ_c, integ_s})
    all_indices = sorted(all_indices)
    idx_to_pos = {idx: i for i, idx in enumerate(all_indices)}
    n = len(all_indices)
    matrix = np.zeros((n, n), dtype=int)
    for phi in merged.index:
        cl = get_phi_classes(merged, phi)
        p, b = idx_to_pos[cl["p"]], idx_to_pos[cl["b"]]
        c, s = idx_to_pos[cl["c"]], idx_to_pos[cl["s"]]
        matrix[p, b] = matrix[b, p] = 1
        matrix[b, c] = matrix[c, b] = 1
        matrix[b, s] = matrix[s, b] = 1
        ip, ib = idx_to_pos[integ_p], idx_to_pos[integ_b]
        ic, is_ = idx_to_pos[integ_c], idx_to_pos[integ_s]
        matrix[ip, ib] = matrix[ib, ip] = 1
        matrix[ib, ic] = matrix[ic, ib] = 1
        matrix[ib, is_] = matrix[is_, ib] = 1
    if integration_agents:
        for phi in merged.index:
            agent = phi_info.get(phi, ("?",))[0]
            if agent in integration_agents:
                b_idx = get_phi_classes(merged, phi)["b"]
                b_pos = idx_to_pos[b_idx]
                matrix[idx_to_pos[integ_p], b_pos] = 1
                matrix[b_pos, idx_to_pos[integ_p]] = 1
    return matrix, all_indices

def apply_ymod(matrix, all_indices, merged_before, merged_after, phi_info_before,
phi_info_after):
    mod_matrix = matrix.copy()
    idx_to_pos = {idx: i for i, idx in enumerate(all_indices)}
    before_classes, after_classes = set(), set()
    for col in merged_before.columns:
        before_classes.update(merged_before[col][merged_before[col] != 0].unique())
        after_classes.update(merged_after[col][merged_after[col] != 0].unique())
    removed = before_classes - after_classes
    for idx in removed:
        if idx in idx_to_pos:
            pos = idx_to_pos[idx]
            for j in range(len(all_indices)):
                if j != pos and matrix[pos, j] > 0:
                    mod_matrix[j, j] += 1

```

```

added = after_classes - before_classes
for idx in added:
    if idx in idx_to_pos:
        pos = idx_to_pos[idx]
        mod_matrix[pos, pos] += 2
        for j in range(len(all_indices)):
            if j != pos and mod_matrix[pos, j] > 0:
                mod_matrix[j, j] += 1
for col in MERGE_COLUMNS:
    after_vals = merged_after[col][merged_after[col] != 0]
    after_groups = after_vals.groupby(after_vals).groups
    for val, phis in after_groups.items():
        if val in idx_to_pos:
            orig_indices = set()
            for phi in phis:
                orig_val = merged_before.loc[phi, col]
                if orig_val != 0:
                    orig_indices.add(orig_val)
            if len(orig_indices) > 1:
                pos = idx_to_pos[val]
                mod_matrix[pos, pos] += 1
                for orig_idx in orig_indices:
                    if orig_idx in idx_to_pos and orig_idx != val:
                        mod_matrix[idx_to_pos[orig_idx], idx_to_pos[orig_idx]] +=
1
                        for j in range(len(all_indices)):
                            if j != pos and mod_matrix[pos, j] > 0:
                                mod_matrix[j, j] += 1
return mod_matrix

def process_variant(name, merged_base, indexed, phi_main, reassign_map=None,
phi_reassign_func=None):
    phi_info_before = {}
    for _, row in phi_main.iterrows():
        phi_info_before[row["phi"]] = (str(row["agent"]), row["node_str"])
    phi_info_after = {}
    if phi_reassign_func:
        phi_reassign = phi_reassign_func(phi_main)
        for _, row in phi_main.iterrows():
            phi = row["phi"]
            if phi in phi_reassign:
                phi_info_after[phi] = (phi_reassign[phi], row["node_str"])
            else:
                phi_info_after[phi] = (str(row["agent"]), row["node_str"])
    elif reassign_map:
        for _, row in phi_main.iterrows():
            phi = row["phi"]
            agent = str(row["agent"])
            phi_info_after[phi] = (reassign_map.get(agent, agent), row["node_str"])
    else:
        phi_info_after = dict(phi_info_before)

```

```

    new_agents = set(a for a, _ in phi_info_after.values() if a.startswith("0") or
a.startswith("0E"))
    merge_agents = new_agents - {"ESB"}
    merged_after = rebuild_merged(indexed, phi_info_after, merge_agents)
    integration_agents = set(a for a, _ in phi_info_after.values() if a in
merge_agents)
    matrix_after, indices_after = build_graph(merged_after, phi_info_after,
integration_agents)
    H_final = compute_H(matrix_after)
    matrix_before, indices_before = build_graph(merged_base, phi_info_before, set())
    mod_matrix = apply_ymod(matrix_after, indices_after, merged_base, merged_after,
        phi_info_before, phi_info_after)
    H_mod = compute_H(mod_matrix)
    H_base = compute_H(matrix_before)
    diag = np.diag(mod_matrix)
    H_ymod = sum(math.log2(d) for d in diag if d > 1)
    n_agents = len(set(a for a, _ in phi_info_after.values())) + 1
    print(f"\n {name}")
    print(f" Агентов: {n_agents}, Вершин: {len(indices_after)}, Рёбер:
{int(matrix_after.sum())//2}")
    print(f" H_base={H_base:.2f}, H_final={H_final:.2f}, H_mod={H_mod:.2f},
H_ymod={H_ymod:.2f}")
    return {"H_base": H_base, "H_final": H_final, "H_mod": H_mod, "H_ymod": H_ymod}

if __name__ == "__main__":
    merged_base = pd.read_excel(OUTPUT_DIR / "task0_population.xlsx",
        sheet_name="4. Слияние классов", index_col=0)
    indexed = pd.read_excel(OUTPUT_DIR / "task0_population.xlsx",
        sheet_name="3. Индексирование", index_col=0)
    phi_main = pd.read_excel(OUTPUT_DIR / "task0_ontology_inference.xlsx",
sheet_name="Каталог Φ")
    phi_main = phi_main[phi_main["integrated"] != 1].copy()
    phi_main["_sort"] = phi_main["phi"].str.replace("φ", "", regex=False).astype(int)
    phi_main = phi_main.sort_values("_sort").reset_index(drop=True)
    fa = pd.read_csv(OUTPUT_DIR / "task0_adjacency_phi_agents.csv", index_col=0)
    agent_codes = ["0001", "0002", "0003", "1001", "1002", "1003", "1004",
        "1005", "1006", "1007", "1008", "1009", "1010", "1011"]
    phi_agents = {}
    for i in range(len(fa)):
        for j, code in enumerate(agent_codes):
            if fa.iloc[i].iloc[j] == 1:
                phi_agents[fa.index[i] if isinstance(fa.index[i], str) else
f"φ{fa.index[i]}"] = code
                break
    phi_main["agent"] = phi_main["phi"].map(phi_agents)
    phi_main["node_str"] = phi_main["node"].apply(lambda x: str(int(x)).zfill(6))

    reassign_v1 = {
        "1009": "0002", "1010": "0002", "1011": "0003", "1008": "0001",
        "1001": "0001", "1002": "0001", "1003": "0001", "1004": "0001",
        "1005": "0001", "1006": "0001", "1007": "0001",
    }

```

```

reassign_v2 = {a: "0004" for a in PERSONNEL}
def v3_reassign(phi_df):
    result = {}
    counter = 1
    for _, row in phi_df.iterrows():
        result[row["phi"]] = f"0E{counter:04d}"
        counter += 1
    return result

r1 = process_variant("Вариант 1: Персонал → существующим ПА",
                    merged_base, indexed, phi_main, reassign_map=reassign_v1)
r2 = process_variant("Вариант 2: Персонал → новый ПА",
                    merged_base, indexed, phi_main, reassign_map=reassign_v2)
r3 = process_variant("Вариант 3: Микросервис (1:1)",
                    merged_base, indexed, phi_main,
phi_reassign_func=v3_reassign)

print("\nСВОДКА:")
for name, r in [("B1", r1), ("B2", r2), ("B3", r3)]:
    print(f" {name}: H_final={r['H_final']:.2f}, H_mod={r['H_mod']:.2f},
H_ymod={r['H_ymod']:.2f}")

```


Приложение Г

Акт внедрения результатов диссертационной работы в ООО «ПРАЙМ ГРУП»



ООО "ПРАЙМ ГРУП"

ул. Успенская, д. 3, пом. I, Московская область,
г. Красногорск, 143409

ТЕЛ.: +7 (929) 968-57-48 +7 (499) 579-7701/02

ФАКС: +7 (499) 579-7701/01

e-mail: VPankratov@primegroup.ru

www.primegroup.ru

АКТ

Внедрения в ООО «ПРАЙМ ГРУП» результатов диссертационной работы
Дерябина Сергея Андреевича на соискание ученой степени кандидата технических наук по
теме

«Модели и методы оценки и анализа архитектур цифровой трансформации
горнопромышленных систем»
по специальности 2.3.1. «Системный анализ, управление и обработка информации,
статистика»

Настоящий акт подтверждает, что результаты работы Дерябина Сергея Андреевича по
теме «Модели и методы оценки и анализа архитектур цифровой трансформации
горнопромышленных систем» использованы в деятельности общества с ограниченной
ответственностью «ПРАЙМ ГРУП».

Комиссия в составе:

Лукичев Максим Игоревич – председатель комиссии, генеральный директор;

Панкратов Вадим Александрович – член комиссии, заместитель директора
департамента – начальник Управления по развитию бизнеса;

Горячева Анна Юрьевна – член комиссии, администратор проектов;

**рассмотрела материалы диссертационной работы Дерябина С.А. и установила
следующее:**

1. Методика и наработки данной диссертации используются в процессе моделирования архитектуры информационных систем корпоративных Заказчиков.
2. В частности, разработанный в рамках диссертации набор формальных операций по модификации логической конфигурации архитектуры ИС, позволяет количественно оценивать и сопоставлять варианты структурных преобразований, в рамках процедур цифровой трансформации таких информационных систем.
3. А предложенный алгоритм процедур конфигурирования и анализа архитектур систем предлагает формальную инструментальную основу для решения задач управления жизненным циклом их цифровой трансформации.

Председатель комиссии:

Члены комиссии:



М.И. ЛУКИЧЕВ
В.А. ПАНКРАТОВ
А.Ю. ГОРЯЧЕВА

Приложение Д

Акт внедрения результатов диссертационной работы в ООО «Брайт Софт»



**Брайт
Софт**

Общество с ограниченной ответственностью «Брайт Софт»
614066 г. Пермь, Ш. Космонавтов 111д, офис 105
info@bright-soft.ru; www.bright-soft.ru
ОГРН 1195958009060; ИНН 5905059850; КПП 590501001

«2» июня 2026 г.

АКТ

внедрения в ООО «Брайт софт» результатов диссертационной работы Дерябина Сергея Андреевича на соискание ученой степени кандидата технических наук по теме

«Модели и методы оценки и анализа архитектур для построения автономно функционирующих горнопромышленных систем» (специальность 2.3.1. «Системный анализ, управление и обработка информации, статистика»)

Настоящий акт подтверждает, что результаты работы Дерябина Сергея Андреевича по теме «Модели и методы оценки и анализа архитектур для построения автономно функционирующих горнопромышленных систем» использованы в деятельности общества с ограниченной ответственностью «Брайт софт».

Комиссия в составе:

Штатнов Сергей Владимирович – председатель комиссии, директор по развитию бизнеса;

Рычков Артем Владимирович – член комиссии, директор проектов;

Свиридкина Ксения Александровна – член комиссии, администратор проектов;

рассмотрела материалы диссертационной работы Дерябина С.А. и заключила, что предложенный подход к проектированию и анализу архитектур геопространственных промышленных систем является важным инновационным решением, обеспечивающим ликвидацию методологического пробела в области автоматизации и цифровой трансформации сложных крупномасштабных объектов. В настоящий момент разработанные модели и алгоритмы имплементированы в рабочие процессы и продукты компании и используются при планировании и технико-экономическом обосновании проектов автоматизации производственных процессов клиентов.

Онтологическая модель геопространственной промышленной системы и алгоритмы конфигурирования и количественного анализа применены в составе продукта «Smart Geo-Data» для разрабатываемых архитектурных решений в области нефтедобычи, управления лесными и природными зонами, что позволяет говорить о потенциальном сокращении расходов на поддержку жизненного цикла разрабатываемых и эксплуатируемых продуктов компании.

Председатель комиссии:

Члены комиссии:

Штатнов С.В.

Рычков А.В.

Свиридкина К.А.



Приложение Е

Акт внедрения результатов диссертационной работы в ООО «Экспасофт»

expasoft*

ООО «Экспасофт»

expasoft.com | expasoft@expasoft.com | +7 (383) 281-94-92

АКТ

внедрения в Общество с ограниченной ответственностью «Экспасофт» результатов диссертационной работы Дерябина Сергея Андреевича на соискание ученой степени кандидата технических наук по теме «Модели и методы оценки и анализа архитектур для построения автономно функционирующих горнопромышленных систем» (специальность 2.3.1. «Системный анализ, управление и обработка информации, статистика»)

г. Новосибирск

08.06.2026 г.

Настоящий акт подтверждает, что результаты работы Дерябина Сергея Андреевича по теме «Модели и методы оценки и анализа архитектур для построения автономно функционирующих горнопромышленных систем» применены в деятельности общества с ограниченной ответственностью «Экспасофт».

Комиссия в составе:

Алямкин Сергей Анатольевич – председатель комиссии, заместитель директора по науке и технике;

Григорьев Евгений Владимирович – член комиссии, заместитель директора;

Генне Артур Александрович – член комиссии, руководитель проекта;

рассмотрела материалы диссертационной работы Дерябина С. А. и заключила, что предложенная методика оценки и анализа архитектур систем является ценным научно-практическим результатом и в настоящий момент имплементирована в процессы компании в области управления жизненным циклом программных продуктов.

С использованием данной методики осуществляется технико-экономическое обоснование проектов по автоматизации и цифровой трансформации для клиентов из промышленного и строительного секторов, что способствует существенному сокращению трудозатрат при подготовке технорабочей документации и разработке архитектурных решений.

Подход с онтологическим представлением знаний интегрирован в процессы формализации экспертной информации при создании диалоговых систем, что обеспечило сокращение ошибок интерпретаций и повысило скорость верификации и валидации моделей.

Председатель комиссии:

Члены комиссии:



Алямкин
Григорьев
Генне

Алямкин С. А.

Григорьев Е. В.

Генне А. А.