

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего профессионального образования
«Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики»
(НИУ ИТМО)

На правах рукописи

АФАНАСЬЕВ МАКСИМ ЯКОВЛЕВИЧ

**Разработка и исследование
многоагентной системы для решения задач
технологической подготовки производства**

Специальность 05.11.14 – Технология приборостроения

ДИССЕРТАЦИЯ
на соискание ученой степени
кандидата технических наук

Научный руководитель
к. т. н., доц.
Филиппов Александр Николаевич

Санкт-Петербург – 2016

Оглавление

Список используемых сокращений	6
Введение	7
Глава 1. Анализ состояния вопроса и постановка задачи исследования	13
1.1. Основные задачи технологической подготовки современного научоёмкого производства	13
1.2. Автоматизированные системы технологической подготовки производства и принципы их построения	15
1.3. Проблемы создания комплексной автоматизированной системы технологической подготовки производства	18
1.4. Используемые методы интеграции автоматизированных систем технологической подготовки производства	22
1.5. Пути совершенствования методов интеграции автоматизированных систем технологической подготовки производства . .	25
1.6. Выводы к первой главе и постановка задачи исследования . .	28
Глава 2. Теоретические основы построения многоагентных систем	31
2.1. Архитектура агентов	31
2.1.1. Основные определения	31
2.1.2. Реактивная архитектура	35
2.1.3. Архитектура, управляемая целями	36
2.1.4. Гибридная архитектура	39
2.2. Описание математической модели функционирования многоагентных систем	40

2.2.1.	Многоагентные среды	41
2.2.2.	Многоагентные системы	46
2.2.3.	Абстрактные ресурсы и их роль в MAC	50
2.3.	Холонический подход к решению задачи технологической подготовки сложных производств	54
2.3.1.	Холон	56
2.3.2.	Холонические производственные системы	57
2.3.3.	Современное состояние ХПС	60
2.4.	Выводы ко второй главе	78
Глава 3.	Программное и аппаратное обеспечение многоагентной системы технологического назначения	80
3.1.	Программное обеспечение MAC	81
3.1.1.	Универсальный формат представления технологических данных и знаний	82
3.1.2.	Реализацияprotoагента	92
3.1.3.	Онтологический словарь многоагентной системы	100
3.1.4.	Протокол взаимодействия агентов	104
3.2.	Аппаратное обеспечение MAC	109
3.3.	Выводы и результаты по третьей главе	113
3.4.	Обзор стандартов FIPA	114
3.4.1.	Взаимодействие агентов	117
3.4.2.	Коммуникационные подуровни FIPA	120
3.4.3.	Управление агентами	122
3.4.4.	Абстрактной архитектура FIPA	126
3.5.	Прикладные библиотеки разработки многоагентных систем	127
3.5.1.	JADE	128
3.5.2.	Repast	134

3.5.3. SPADE	142
3.5.4. Сравнительный анализ рассмотренных средств проек- тирования МП	144
3.6. Выводы к третьей главе	146
Глава 4. Реализация ТПП изделий из ПКМ на основе много- агентной холонической системы	147
4.1. Описание интегрируемых систем автоматизации ТПП	151
4.2. Описание агентов, моделей поведения и целей	155
4.2.1. Агент-преобразователь	156
4.2.2. Агент-интерфейс	159
4.3. Описание взаимодействия агентов ИУП ТПП	161
4.4. Выводы и результаты по четвёртой главе	170
Заключение	173
Список литературы	175
Приложение А. Представление языка ИУП ТПП в расширен- ной форме Бэкуса–Наура	191
A.1. Тип данных	191
A.2. Триплет	191
A.3. Операция	192
A.4. Функция	192
A.5. Фрейм	192
A.6. Синтагма	192
A.7. Продукция	193
Приложение Б. Онтологический словарь ИУП ТПП	194

Приложение В. Стандарт FIPA	197
Приложение Г. Описание интегрируемых систем автоматизации ТПП	199
Г.1. SALOME 6.4.0	199
Г.2. SmarTeam V5R20	200
Г.3. OpenERP 6.0.3	202
Г.4. PyCAM 0.5.1	204
Г.5. Вертикаль 2011	204
Г.6. PostgreSQL 9.1.1	206
Г.7. Samcef	207
Г.8. Digimat	207
Г.9. Moldex3D	208
Приложение Д. Программная реализация базового агента	209
Приложение Е. Акты внедрения результатов диссертационной работы	211

Список используемых сокращений

API — Application Programming Interface.

JSON — JavaScript Object Notation.

ORM — Object-relational mapping.

XML — eXtensible Markup Language.

XMPP — Extensible Messaging and Presence Protocol.

АСТПП — Автоматизированная система технологической подготовки производства.

ВМ — Виртуальная машина.

ВРМ — Виртуальное рабочее место.

ИА — Интеллектуальный агент.

ИУП ТПП — Информационно-управляющая платформа технологической подготовки производства.

МАС — Многоагентная система.

MC — Многоагентная среда.

ПКМ — Полимерный композиционный материал.

ПО — Программное обеспечение.

РВИС — Распределённый виртуальный испытательный стенд.

СМД — Словарь метаданных.

СТО — Средства технологического оснащения.

ТП — Технологический процесс.

ТПП — Технологическая подготовка производства.

ХПС — Холоническая производственная система.

Введение

Актуальность темы диссертации. Всё возрастающая конкуренция на рынке подталкивает современные приборостроительные предприятия к постоянному улучшению и развитию производства. В настоящее время одним из наиболее перспективных способов достижения высокой конкурентоспособности является повышение эффективности технологической подготовки производства (ТПП) за счёт применения современных средств автоматизации. Особенно это актуально для предприятий, использующих передовые технические решения и технологии, требующие дополнительных инженерных изысканий.

Большой вклад в разработку базовых принципов построения и взаимодействия автоматизированных систем технологической подготовки производства (АСТПП) внесли С. П. Митрофанов, В. И. Аверченков, Г. К. Горанский, В. Д. Цветков, Н. М. Капустин, В. В. Павлов, В. М. Вальков, А. Н. Филиппов, Д. Д. Куликов, Б. С. Падун, Е. И. Яблочников и многие другие. Тем не менее, в условиях современного научно-ёмкого производства многие из этих принципов нарушаются. В первую очередь это связано с отсутствием универсальной интеграционной среды, способной собрать воедино различные инструментальные средства автоматизации ТПП.

Решение данной проблемы наиболее целесообразно с применением методов распределённого искусственного интеллекта, базовой дисциплиной которого является теория многоагентных систем (МАС). Применение МАС для решения задач технологической подготовки производства позволит создать открытую среду интеграции технологических данных и знаний, построенную на простой модели расширения функциональности и горизонтального масштабирования информационного пространства технологической подготовки производства.

На сегодняшний день существует достаточное количество работ, посвящённых применению многоагентных систем в промышленности и производстве, но ни в одной из них не представлено детальное исследование рассматриваемой предметной области — технологической подготовки приборостроительного производства. На основе проводимого исследования необходимо подготовить методику, применение которой даст возможность повысить уровень автоматизации при решении задач технологической подготовки и увеличит структуризацию информационной среды современного предприятия.

Всё вышесказанное подтверждает актуальность проектирования и разработки многоагентной системы, позволяющей повысить степень интеграции информационного пространства и упростить решения задач технологической подготовки производства.

Цель диссертационной работы состоит в совершенствовании методов автоматизации технологической подготовки производства путём использования информационно-управляющей платформы, созданной на базе многоагентной системы.

Для достижения поставленной цели в диссертационной работе потребовалось решить следующие **основные задачи**:

- Исследовать существующие методы построения распределённых одноранговых многоагентных систем, их архитектуры и области применения, а также инструментальные средства разработки.
- Разработать математические модели многоагентной среды и многоагентной системы
- Описать язык представления технологических данных и знаний, используемый агентами.

- Разработать систему моделирования информационного пространства технологической подготовки производства, базирующуюся на концепциях «облачных» вычислений и виртуальных рабочих мест.
- Реализовать многоагентную систему и опробовать её при решении конкретных технологических задач.

Научная новизна работы заключается в следующем:

- Предложена методика структурной интеграции АСТПП в рамках единой информационно-управляющей платформы технологической подготовки производства, основанная на базовых принципах теории многоагентных систем и виртуального строкового пространства.
- Предложена методика моделирования информационной среды приборостроительного предприятия, базирующаяся на концепциях «облачных» вычислений и виртуальных рабочих мест.

Практическая ценность работы заключается в следующем:

- Разработан и программно реализован комплекс алгоритмов многоагентной интеграции АСТПП.
- Сконфигурирован серверный кластер, и на его основе реализован распределённый виртуальный испытательный стенд для моделирования информационной среды приборостроительного предприятия.
- Реализована многоагентная система, осуществляющая интеграцию средств автоматизации технологической подготовки производства изделий из полимерных композиционных материалов.

Реализация результатов работы. Результаты исследований и разработанный комплекс методов и инструментальных средств нашли применение в:

- НИР по государственному контракту № П571 от 05.09.08 на 3 года, заказчик Федеральное агентство по образованию/Министерство по образованию, тема «Разработка и реализация модели непрерывного повышения квалификации педагогических кадров российских технических вузов в системе „вуз–инжиниринговый центр–организация“».
- НИОКР № 21083 от 15.12.10, заказчик ООО «Завод по переработке пластмасс имени „Комсомольской правды“», тема «Создание интегрированной распределённой системы проектирования, прототипирования и подготовки производства изделий».
- НИР по государственному контракту № 310220 «Разработка базовых технологий проектирования и производства приборов нового поколения на основе полимерных композиционных материалов для реальных условий эксплуатации в авиационной, космической, морской и другой технике» по теме 2011-1.4-514-126-027;
- Учебном процессе НИУ ИТМО на кафедре технологии приборостроения.
- Программном и организационно-техническом обеспечении научно-образовательного центра НИУ ИТМО кафедры технологии приборостроения.

На защиту выносятся следующие основные результаты и положения:

- Модель многоагентной системы для решения задач интеграции автоматизированных систем технологической подготовки производства в рамках единого информационного пространства технологической подготовки производства.
- Архитектура информационно-управляющей платформы технологической подготовки производства, включающая язык представления техно-

логических данных и знаний, структуру агентов и протокол взаимодействия.

- Метод моделирования информационной среды приборостроительного предприятия, основанный на концепции «облачных вычислений».

Апробация работы. Основные результаты диссертации докладывались на следующих конференциях: VI Всероссийская межвузовская конференция молодых учёных (14–17 апреля 2009), Девятая сессия международной научной школы «Фундаментальные и прикладные проблемы надёжности и диагностики машин и механизмов» (26–30 октября 2009), XXXIX научная и учебно-методическая конференция Санкт-Петербургского государственного университета информационных технологий, механики и оптики (2–5 февраля 2010), VII Всероссийская межвузовская конференция молодых учёных (20–23 апреля 2010), XL научная и учебно-методическая конференция национального исследовательского университета информационных технологий, механики и оптики (1–4 февраля 2011), VIII Всероссийская межвузовская конференция молодых учёных (12–15 апреля 2011), Десятая сессия международной научной школы «Фундаментальные и прикладные проблемы надёжности и диагностики машин и механизмов» (24–27 октября 2011).

Публикации. Материалы диссертации опубликованы в 8 печатных работах, из них 4 статьи в изданиях, рекомендованных ВАК. Полный перечень работ приведён в конце авторефера.

Личный вклад автора. Содержание диссертации и основные положения, выносимые на защиту, отражают персональный вклад автора в опубликованные работы. Подготовка к публикации полученных результатов проводилась совместно с соавторами, причём вклад диссертанта был определяющим. Все представленные в диссертации результаты получены лично автором.

Структура и объем диссертации. Диссертация состоит из введения, 4 глав, заключения, библиографии и 6 приложений. Общий объём диссертации 131 страница, включая 24 иллюстрации и 1 таблицу. Библиография содержит 96 наименований на 12 страницах.

В первой главе приводится анализ объекта исследования, включающий рассмотрение основных задач технологической подготовки производства, принципов проектирования автоматизированных средств, обеспечивающих информационную поддержку решения этих задач, и основных проблем, возникающих в процессе интеграции данных средств автоматизации в единую информационную среду приборостроительного предприятия.

Во второй главе описываются теоретические основы построения многоагентных систем, представляющие архитектуру программного агента, математические модели многоагентной среды и многоагентной системы, а также функциональные особенности построения многоагентных систем для решения задач технологической подготовки производства.

В третьей главе описываются программные и аппаратные средства многоагентной системы, обеспечивающей функционирование информационно-управляющей платформы технологической подготовки производства.

В четвёртой главе приводятся практические результаты, полученные в процессе разработки многоагентной интеграционной сети технологической подготовки производства изделий из полимерных композиционных материалов (ПКМ), а также моделирования полученной среды на распределённом виртуальном испытательном стенде

Глава 1

Анализ состояния вопроса и постановка задачи исследования

1.1. Основные задачи технологической подготовки современного научоёмкого производства

Технологическая подготовка производства (ТПП) представляет собой комплексный многоитерационный процесс, направленный в первую очередь на обеспечение полной технологической готовности приборостроительного предприятия к производству новых изделий с заданными технико-экономическими показателями (высоким техническим уровнем, качеством изготовления, а также с минимальными трудовыми и материальными издержками — себестоимостью при конкретном техническом уровне предприятия и планируемых объемах производства) [1, 2]. Следовательно, ТПП позволяет более эффективно использовать ресурсы предприятия и раньше конкурентов выводить на рынок новые продукты.

К основным задачам технологической подготовки производства можно отнести:

- *Обеспечение технологичности изделия*, т. е. оптимизация и упрощение его конструкции без потери эксплуатационных характеристик и ухудшения внешнего вида изделия.
- *Проектирование и разработка высокоеффективного технологического процесса (ТП)*. Сводится к выбору наиболее подходящего процесса изготовления, обеспечивающего изготовление изделий по заданным характеристикам за минимальное время и с минимальными затратами. Является наиболее ответственным этапом ТПП, т. к. от выбора пра-

вильной технологии в конечном итоге зависят сроки выхода на рынок готового продукта.

- *Проектирование и изготовление средств технологического оснащения* (СТО). Данная задача является наиболее трудоёмкой частью классической технологической подготовки производства, т. к. предполагает весь комплекс работ по созданию СТО, начиная с эскизного проектирования и заканчивая разработкой технологии и непосредственно изготовлением СТО. Увеличение времени на данном этапе может существенно отодвинуть сроки начала производства, что неминуемо повлечёт за собой увеличение себестоимости конечного продукта.
- *Управление процессами ТПП*. Эффективное управление должно повысить производительность всех этапов ТПП, позволяя упростить решение многих задач за счёт более продуктивного взаимодействия между специалистами, а также более тесной интеграции средств проектирования технологической подготовки производства.

В последнее время к этим задачам можно добавить ещё одну: *проектирование материала*. Данная задача возникает из-за того, что для реализации современных конструкций уже недостаточно существующих материалов. Необходимо создавать материал под конкретное изделие, при этом его свойства становятся свойствами изделия. Сложность данной задачи сопоставима со сложностью всех предыдущих, ведь проектирование материала не ограничивается одним лишь поиском в базе данных или физическим экспериментом. Зачастую требуется моделирование структуры (состава) будущего материала с помощью специализированных методов и алгоритмов.

Решить все вышеуказанные задачи в комплексе достаточно трудно. Ресурсы (в первую очередь временные), потраченные на проектирование ТП в конечном итоге снижают суммарный экономический эффект от использо-

вания даже самых передовых технологий. Очевидно, что для повышение эффективности ТПП необходимо использовать современные информационные технологии, позволяющие автоматизировать большую часть задач проектирование технологических процессов.

1.2. Автоматизированные системы технологической подготовки производства и принципы их построения

Автоматизированные системы технологической подготовки производства (АСТПП) применяются в нашей стране уже более 40 лет. Вопросам разработки теоретических основ создания интегрированных автоматизированных систем ТПП, а также их внедрения и использования уделено большое внимание в работах С. П. Митрофанова, В. И. Аверченкова, Г. К. Горанского, В. Д. Цветкова, Д. Д. Куликова, А. Н. Филиппова, Б. С. Падуна, Е. И. Яблочникова и многих других. Первые автоматизированные системы были направлены на оптимизацию того или иного этапа ТПП: расчёт припусков, выбор инструмента и режимов резания и т. д. Со временем эти системы достигли определённого уровня автоматизация, появилась тенденция к укрупнению [3–10].

Но и само производство не стояло на месте, появлялись всё новые и новые технологии: порошковая металлургия, быстрое прототипирование, высокоскоростное многокоординатное резание, технологии работы с композиционными материалами. Как следствие, ТПП производств, использующих подобные технологии, тоже сильно усложнялась. Вместе с тем, изменились и информационные технологии, резкое снижение цен на персональные компьютеры и рабочие станции позволило оснастить ими всех специалистов,

принимающих участие в проектировании и разработке технологического процесса. Изменились сами методы проектирования, произошёл отказ от использования бумажных носителей, и на первое место вышли *технологические данные и знания*. Вместо абстрактного образа, являющегося совокупностью некоторой технологической документации, появилась информационная модель ТП, разработка которой на сегодняшний день и является основной задачей ТПП.

Всё чаще стал использоваться термин *наукоёмкое производство*, т. е. производство, находящееся на стыке промышленности и науки. Технологическая подготовка подобных производств потребовала пересмотра многих положений классической теории построения автоматизированных систем, тем не менее, базовые принципы остались прежними. К ним относятся [11, 12]:

1. ***Принцип системного единства.*** Предполагает функционирование всех компонентов АСТПП предприятия как единого целого, т. е. ключевая роль отводится вопросам информационной интеграции отдельных элементов, а не их функционированию.
2. ***Принцип декомпозиции.*** Предполагает разделение компонентов системы, на составные части, что должно существенно снизить её общую сложность за счёт упрощения отдельных её элементов, суммарная эффективность которых будет выше благодаря уменьшению общего объёма передаваемой информации и, как следствие, упрощению протокола взаимодействия.
3. ***Принцип модульности.*** Компоненты системы должны обладать достаточной автономностью и самостоятельностью, что позволит использовать их как отдельно, так в комплексе. При этом нарушение работы одного из модулей не должно приводить к нарушению работы системы в целом.

4. *Принцип совместимости.* Должен быть реализован универсальный механизм взаимодействия всех модулей и компонентов системы. Необходимо обеспечить не только «механическую» совместимость, т. е. совместимость на уровне программных интерфейсов и схем представления данных, но и интероперабельность на уровне технологических знаний и понятий, с которыми должна работать автоматизированная система.
5. *Принцип открытости.* Необходимо стремиться к максимальной децентрализации архитектуры системы, что позволит вводить в неё новые модули и компоненты в полуавтоматическом режиме, т. е. без модификации вышестоящих модулей и без внесения корректива в существующие механизмы управления, что позволит создать гибкую систему, поддерживающую горизонтальное масштабирование.
6. *Принцип стандартизации.* Подразумевает использование максимального числа унифицированных и типовых решений, позволяющих снизить затраты на проектирование и разработку единой автоматизированной системы за счёт отсутствия дублирования функций отдельных подсистем, а также повысить общую надёжность всего программного комплекса.
7. *Принцип эргономичности.* Система в первую очередь должна быть ориентирована на взаимодействие с человеком. Сложные интерфейсы необходимо разделить на небольшие блоки, ориентированные на работу с конкретным специалистом. Вкупе с принципом открытости это даёт возможность создавать новые, упрощённые интерфейсы даже для тех подсистем и компонентов, интерфейс которых уже реализован, но по каким-то причинам не подходит для решения конкретных задач. Большее внимание должно быть уделено диалогу с пользователем, а не созданию максимально подробных интерфейсов.

8. *Принцип ориентации на новые достижения.* При создании автоматизированных систем технологической подготовки производства должны быть использованы последние научно-технические достижения в области методов построения АСТПП, в области методов и средств технологической подготовки производства, а также области организации производства и интеграции связанных с ним данных и знаний.

1.3. Проблемы создания комплексной автоматизированной системы технологической подготовки производства

К сожалению, создание комплексной автоматизированной системы, решающей все основные задачи технологической подготовки современного научоёмкого производства и удовлетворяющей всем вышеуказанным принципам практически невозможно. Это обусловлено в первую очередь высокой сложностью самого процесса технологической подготовки подобных производств, что делает создание единой системы автоматизации ТПП «под ключ» экономически невыгодным. Подобная система может быть создана только в наиболее значимых отраслях промышленности — военной, космической, атомной и т. п.

Очевидно, что альтернативой создания единой автоматизированной системы ТПП «под ключ» может стать интеграция различных систем, каждая из которых выполняет отдельную функцию.

В последние годы появляется всё больше и больше различных средств автоматизации ТПП. Основные из них можно разделить на следующие классы: [13–15]:

- системы подготовки управляющих программ для станков с числовым программным управлением (*Computer-Aided Manufacturing – CAM*);
- системы инженерного моделирования (*Computer-Aided Engineering – CAE*);
- системы управления данными об изделии (*Product Data Management – PDM*);
- системы непрерывной информационной поддержки поставок и жизненного цикла (*Continuous Acquisition and Life cycle Support – CALS*);
- системы управления жизненным циклом изделия (*Product Lifecycle Management – PLM*);
- системы автоматизированного обеспечения качества (*Computer-Aided Quality assurance – CAQ*);
- системы управления и планирования производством (*Production Planning and Control – PPC*);
- системы автоматизированного проектирования технологических процессов (*Computer-Aided Process Planning – CAPP*)
- системы управления ресурсами предприятия (*Enterprise Resource Planning – ERP*);
- производственные исполнительные системы (*Manufacturing Execution System – MES*).

Совместное использование всех этих систем для технологической подготовки современного приборостроительного производства может привести к тому, что информационное обеспечение ТПП будет сложнее самого производства. Можно сказать, что появилось новое направление — *технология применения средств АСТПП*, связанное не только с пониманием самих процессов технологической подготовки, но и управлением связанными с ней данными и знаниями. Возникло противоречие: с одной стороны необходимо создать гибкую производственную среду (т. е. максимально задействовать современные средства вычислительной техники для ТПП), а с другой —

не допустить резкого увеличения затрат на поддержание работоспособности подобной системы за счёт её слабой структуризации.

Решение данной задачи обозначено выше — это интеграция. Под интеграцией понимается объединение нескольких существующих и используемых систем в единую информационную среду предприятия, что полностью соответствует принципам системного единства и открытости. К сожалению, при этом возникает следующая большая проблема. Отказ от разработки модулей системы с нуля неминуемо подталкивает предприятия к использованию готовых коммерческих решений и здесь на качество технологической подготовки производства влияют уже другие — рыночные факторы.

Каждый производитель программных решений, используемых для автоматизации работ, связанных с технологической подготовкой производства:

- Во-первых, старается наделить свою систему максимальным количеством функций, что сильно усложняет её, нарушая принципы эргономичности и эффективности. Сложность интерфейса заставляет специалистов использовать лишь ту часть функций системы, которая им наиболее понятна.
- Во-вторых, заботится о взаимодействии только своих систем, всячески усложняя методы их взаимодействия с системами конкурентов и отвергая принцип открытости.

Существует стандарт (**ГОСТ Р ИСО 10303–2002** [16]), регламентирующий единый механизм представления данных об изделии на протяжении всего жизненного цикла независимо от конкретной системы автоматизации, а также интеграцию этих данных. Но, как показывает практика, в процессе создания единого информационного пространства ТПП данный стандарт *не используется, либо используется лишь частично* [17].

Дополнительным фактором, препятствующим массовому внедрению автоматизированных систем ТПП на предприятиях, является малое количество подобных программных продуктов отечественного производства. Использование же импортных решений связано с дополнительными трудностями: предприятие либо должно отказаться от отечественных стандартов в пользу международных, либо каким-то образом модифицировать систему, зачастую отказываясь от части функций. Правда, по мнению автора, данная проблема гораздо менее актуальна, чем проблема интеграции автоматизированных систем ТПП, решение которой может коренным образом повлиять на качество технологической подготовки производства в целом.

Из всего вышесказанного можно сделать вывод о том, что на сегодняшний день при создании комплексных автоматизированных систем технологической подготовки производства возникают три основные проблемы:

1. ***Проблема сложности.*** В современных условиях практически невозможно создать целостную системы автоматизации ТПП с нуля, что приводит к необходимости использования разнородных автоматизированных систем и интеграции их в комплекс.
2. ***Проблема несовместимости.*** Программные продукты различных производителей плохо совместимы между собой, что обусловлено нежеланием разработчиков ПО лишаться конкурентных преимуществ. Данная проблема усугубляется частичной несовместимостью отечественных и международных стандартов.
3. ***Проблема избыточности.*** Избыточная функциональность многих коммерческих систем автоматизации технологической подготовки производства сильно усложняет их внедрение и эксплуатацию, а также не позволяет пользователям продуктивно работать с ними.

1.4. Используемые методы интеграции

автоматизированных систем технологической подготовки производства

Проведённые исследования показали, что для создания единой среды автоматизации технологической подготовки производства на отечественных предприятиях используются три основных метода:

1. Использование программных продуктов одной фирмы. Редко применяемый метод, подходящий, во-первых, только для предприятий, ранее не использовавших никаких средств автоматизации и, во-вторых, занимающихся выпуском несложной продукции мелкими сериями. Автоматизация в данном случае достигается за счёт отказа от использования «бумажного» документооборота в пользу «электронного». При этом вопросы автоматизации непосредственно проектирования ТП практически не рассматриваются.
2. Привлечение сторонних специалистов по интеграции. Применение данного метода обычно не даёт желаемого эффекта. Это обусловлено тем, что организации, оказывающие подобные услуги не обладают всей полнотой знаний о производственном процессе каждого конкретного предприятия, следовательно вынуждены предлагать некоторые универсальные решения, часто не улучшающие процесс автоматизации ТПП, а наоборот, ухудшающие его.
3. Реализация всего комплекса работ связанных с формированием единой интегрированной среды технологической подготовки производства силами сотрудников самого предприятия, под постоянным контролем специалистов, непосредственно являющихся будущими пользователями разрабатываемого комплекса. Основная часть работ при этом произво-

дится силами сотрудников информационного отдела, а окончательная настройка и тестирование системы проходит без остановки основного производства.

По мнению автора, из всех описанных методов наиболее предпочтительным является последний. Это обусловлено тем, что автоматизация ТПП не может быть навязана «сверху», инициатива должна исходить от специалистов (технологов, нормировщиков, конструкторов СТО и т. д.), ведь они, как никто другой, знают потребности автоматизации именно той задачи ТПП, с которой сталкиваются ежедневно. Рассмотрим основные способы реализации данного метода, а также их достоинства и недостатки.

Первый способ интеграции основывается на анализе информационных потоков, возникающих между различными АСТПП и их автоматизации. Как известно, практически все автоматизированные системы имеют *Интерфейс программирования приложений* (Application Programming Interface — API), позволяющий обращаться к внутренним функциям системы, используя либо интерпретируемые языки программирования (например, диалекты Basic, C#, Java, Python, Lua и т. д.), либо транслируемые — C, C++¹. Обычно API приложения использует *Объектную модель компонентов* (Component Object Model — COM), либо аналогичные средства межпроцессного взаимодействия. Гораздо реже используется метод прямого доступа к базам данных интегрируемых приложений с помощью *Языка структурированных запросов* (Structured Query Language — SQL).

Результатом подобной интеграции обычно является автономный модуль («чёрный ящик»), реализующий статический (односторонний или двухсторонний) обмен данными между интегрируемыми системами. При этом синхронизация происходит по расписанию, т. е. пользователь не может быть уверен,

¹ Необходимо отметить, что в современных АСТПП интерфейсы, использующие подобные языки используются крайне редко.

что в любой момент времени он получит актуальные данные. Основным достоинством данного способа является *простота реализации, при постоянстве информационных потоков и небольшом количестве интегрируемых систем*, что позволяет использовать его для автоматизации ненаукоёмких производств.

К недостаткам данного способа можно отнести:

- Экспоненциальный рост сложности при увеличении количества интегрируемых систем и связей между ними, вследствие необходимости создания полносвязной системы.
- Отсутствие единого механизма добавления новых модулей: любой модуль связывает две конкретные системы, при этом способ взаимодействия уникален.
- Низкая гибкость программной среды в целом, действия пользователей должны быть чётко регламентированы.
- Низкая отказоустойчивость, сбой работы одной из систем, участвующих в интеграционной цепочке, либо недостоверность данных может привести к неработоспособности системы в целом.
- Сложность работы в сетевой среде предприятия, что обусловлено изначальной неприспособленностью большинства API автоматизированных систем к взаимодействию по сети.

Второй способ предполагает создание системы, ключевым компонентом которой является единая система управления данными (PDM/PLM/CALS) [18–20]. С технической точки зрения используются те же методы, что и в предыдущем способе, с теми же проблемами.

С функциональной точки зрения достоинством данного метода является *наличие единой сетевой среды, позволяющей хранить все технологические*

данные в одном месте и отслеживать все их изменения. К недостаткам можно отнести:

- Направленность единой системы в первую очередь на данные и файлы, а не на знания и проектирование ТП.
- Необходимость обучения всего персонала навыкам работы с единой системой.
- Низкая отказоустойчивость системы, PDM/PLM/CALS является «узким местом», т. е. при её отказе весь процесс информационного обмена в АСТПП будет нарушен.

1.5. Пути совершенствования методов интеграции автоматизированных систем технологической подготовки производства

Очевидно, что описанные методы системной интеграции средств автоматизации ТПП нуждаются в существенной доработке. Первое усовершенствование, которое можно внести в вышеописанную схему — это *отказаться от использования интеграционных модулей*, выполняющих роль «мостов» между различными компонентами системы, т. е. отказаться от биполярной интеграции средств автоматизации ТПП в пользу униполярной. Данное положение подразумевает наличие некоторой единой информационно-управляющей среды интеграции АСТПП в рамках рассматриваемой предметной области. Основным достоинством подобной среды является возможность её лёгкой адаптации к условиям каждого конкретного предприятия вне зависимости от того, какие средства автоматизации и в каком количестве использованы.

Предлагаемый подход к усовершенствованию предполагает создание некоторой децентрализованной одноранговой метасистемы, отвечающей

в первую очередь за взаимодействие всех информационных средств технологической среды предприятия, т. е. позволяющей добиться возможности *горизонтального масштабирования и бесшовной интеграции* всех её компонентов.² Дополнительной особенностью системы должна стать возможность свободной работы не только с технологическим данными, но и знаниями.

Все дополнительные модули, создаваемые специалистами предприятия также желательно перенести в состав данной метасистемы, что на этапе промышленной эксплуатации позволит отказаться от необходимости вносить корректировки в работу используемых средств АСТПП. То есть все дополнительное информационное обеспечение будет «вынесено за скобки» конкретных систем и станет частью единой информационной среды ТПП.

Интеграция может быть осуществлена за счёт связи каждого элемента исследуемой системы с некоторым «представителем», более подробной проработки механизмов интеграции «представителя» с внутренней логикой каждого элемента технологической системы, а также разработки универсального протокола взаимодействия.

Так как описанная выше метасистема в первую очередь ориентирована на взаимодействие автоматизированных систем технологической подготовки производства за счёт интеграции не только технологических данных, но и знаний, для её построение целесообразно использовать элементы теории искусственного интеллекта (ИИ). Описание способов применения ИИ в технологии приборостроения можно найти в работах Б. С. Падуна, Д. Д. Куликова, А. Н. Филиппова и многих других. Анализ работ показал, что в интеллектуальных системах ТПП наибольшее распространение получили следующие языки и модели представления знаний [23]:

² Возможность интеллектуальной интеграции гетерогенных баз данных с помощью метасистемы управления показана, например, в работах [21, 22], автором же основное внимание будет уделено не столько технологических данным, сколько знаниям.

1. язык исчисления предикатов первого порядка (логическая модель);
2. семантические или когнитивные сети;
3. фреймы;
4. продукционные правила.

Необходимо отметить, что все представленные методы используют либо символный, либо логический подход к построению интеллектуальных технологических систем, т. е. акцентируют своё внимание на проблемах выработки правил, используемых для построения баз знаний и их обработке; в то время как для решения вышеуказанной задачи создания единой интеграционной среды ТПП большее внимание необходимо уделить вопросам адаптации и взаимодействия.

Последнее утверждение подводит нас к необходимости применения методов распределённого искусственного интеллекта, базовой дисциплиной которого является теория *многоагентных систем* (МАС). Многоагентные системы находят самое широкое применение в тех областях промышленности и науки, где в силу слабой структуризации информационного пространства неприменимы обозначенные выше классические методы искусственного интеллекта [24–26]. В рассматриваемой предметной области МАС уже успешно применяются для организации кооперации в рамках виртуального предприятия [19, 20], что подтверждает перспективность развития данного направления исследований.

Разработка многоагентной системы для интеллектуальной интеграции средств автоматизации технологической подготовки производства в рамках единого информационного пространства позволит решить сформулированные в [разд. 1.3](#) проблемы сложности и несовместимости программных средств автоматизации ТПП. Последняя проблема (проблема избыточности) может быть решена посредством создания специализированных web-серви-

сов, направленных на упрощение пользовательского интерфейса и снижение сложности эксплуатации автоматизированных систем технологической подготовки производства специалистами предприятия. Данное положение требует проработки механизма создания *виртуальных рабочих мест*, базирующегося на концепции «облачных вычислений» и подразумевающего перенос вычислительной нагрузки с персональных компьютеров и рабочих станций, используемых на предприятиях, в единый центр обработки данных, построенный по принципу консолидации вычислительных ресурсов и децентрализации вычислительных сервисов.

Таким образом, само понятие автоматизированной системы технологической подготовки производства может быть преобразовано в понятие сервиса технологического назначения, что полностью соответствует принципу декомпозиции, декларирующему снижение сложности за счёт удаления наиболее слабых организационных и информационных связей.

Затраты на использование подобной усложнённой системы на этапе первичного развёртывания несколько превышают таковые при использовании классических методов интеграции ([рис. 1.1](#)). Однако, дальнейшее использование её должно привести к упрощению внедрения новых средств автоматизации и, как следствие, снижение общее стоимости владения комплексом средств автоматизации технологической подготовки производства [27].

1.6. Выводы к первой главе и постановка задачи исследования

1. Для реализации эффективной технологической подготовки приборостроительного производства в современных условиях необходимо задействовать комплекс средств автоматизации.

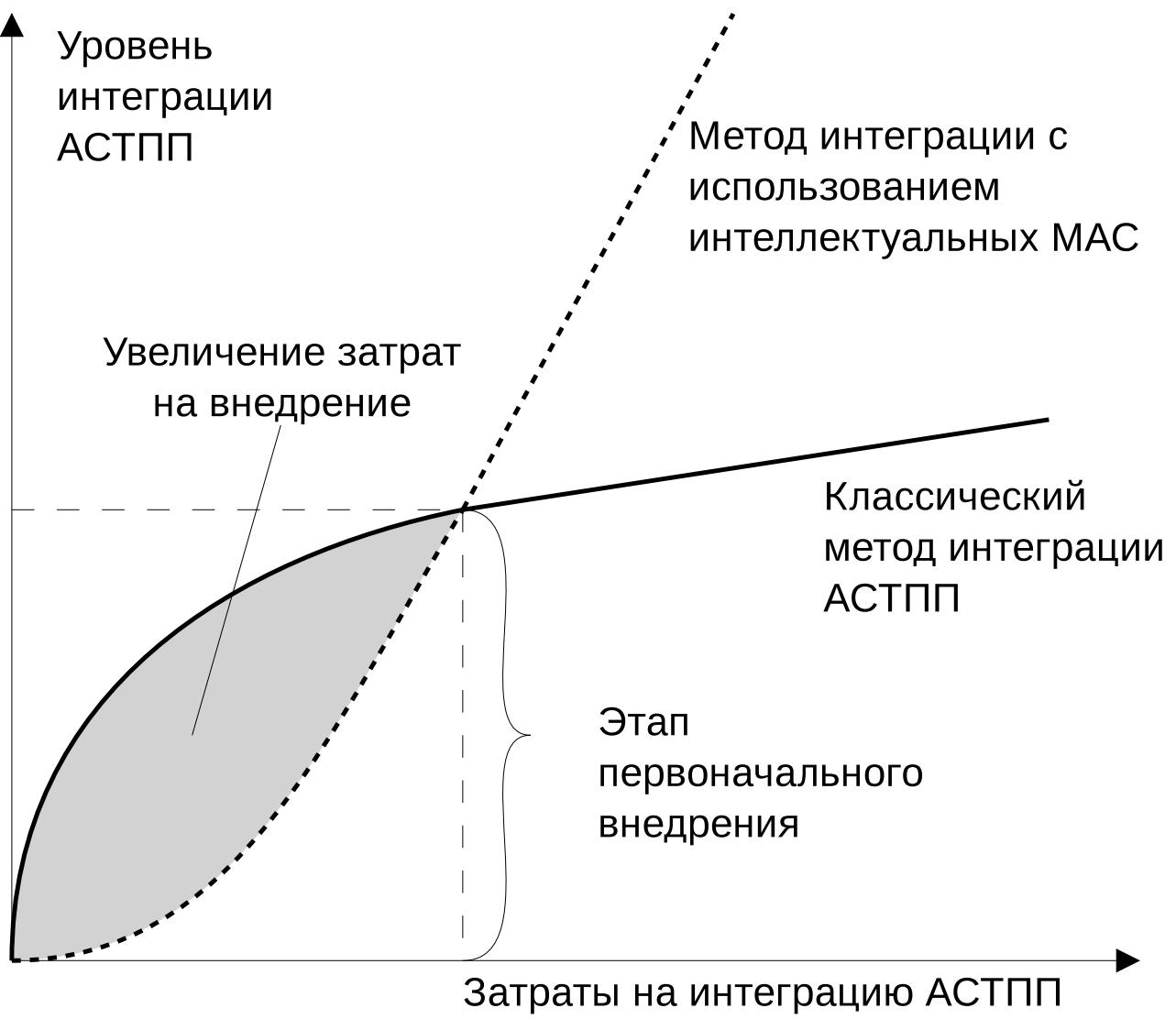


Рис. 1.1. Предполагаемый эффект от внедрения

2. Проведённые исследования показали, что основной проблемой внедрения автоматизированных систем технологической подготовки производства в научёмких отраслях является высокая сложность создания единой информационной среды ТПП, что приводит к снижению интеграции технологических данных и знаний, а в конечном итоге и к увеличению затрат на ТПП.
3. Анализ современных методов интеграции средств информационного обеспечения ТПП показал, что на текущем этапе развития АСТПП не существует готового решения для построения подобной интегрированной

системы, удовлетворяющей всем требованиям современного научно-исследовательского производства, однако базовые принципы создания адаптивных интеллектуальных АСТПП, разработанные отечественными учёными во второй половине двадцатого века остаются справедливыми по сей день.

Основными задачами представленного исследования являются:

1. Разработка методики создания единой интеграционной среды технологической подготовки производства, построенной на базе многоагентной системы и позволяющей уменьшить сложность использования автоматизированных систем технологической подготовки производства, а также улучшить интеграцию между различными их компонентами.
2. Внедрение новых методов эксплуатации автоматизированных систем технологической подготовки производства в соответствие с концепциями виртуальных рабочих мест и «облачных» вычислений.

Глава 2

Теоретические основы построения многоагентных систем

Разработку многоагентной системы, способной преодолеть основные проблемы интеграции в рамках единой информационной среды автоматизированной технологической подготовки приборостроительного производства целесообразно начать с определения ряда фундаментальных положений. Данный этап обусловлен тем, что многоагентные системы не являются программным обеспечение в классическом понимании этого слова. Многоагентная система, в первую очередь, есть метамодель, описывающая основные прикладные задачи, которые будет решать проектируемая система, базовые элементы которой будут описаны в данной главе.

2.1. Архитектура агентов

2.1.1. Основные определения

Агент (интеллектуальный агент, ИА) — это независимая компьютерная система, находящаяся в некоторой среде ([рис. 2.1](#)), способная автономно действовать в ней для достижения своих целей. С одной стороны, среда является объектом наблюдения для агента, среда поставляет агенту данные, которые он может интерпретировать в зависимости от модели своего поведения. С другой стороны, среда является субъектом, на который агент может воздействовать для достижения своих целей. Термин агент впервые был использован Джоном Маккарти (John McCarthy) и Оливером Селфриджом (Oliver G. Selfridge) в пятидесятых годах двадцатого века [[28](#), [29](#)].



Рис. 2.1. Агент и его среда

Интеллектуальный агент, являясь гибкой программно или аппаратно реализованной системой, должен обладать следующими свойствами [30]:

- *социальное поведение* (social ability) — способность функционировать в сообществе с другими агентами (программными или нет), обмениваясь с ними сообщениями с помощью некоторого общепонятного языка коммуникаций;
- *реактивность* (reactivity) — способность реагировать на изменение среды, а также воздействовать на неё;
- *инициативность* (pro-activity) — способность ИА проявлять инициативу для достижения своих целей, тем самым демонстрируя своё целенаправленное поведение.

Помимо базовых свойств агент может обладать и рядом дополнительных свойств, необходимых ему для решения конкретных проблем [31]:

- *автономность* — способность ИА действовать, принимать решения и контролировать своё поведение самостоятельно, без участия человека, что крайне важно при автоматизации операций, происходящих по расписанию, например, актуализация и синхронизация данных между подсистемами АСТПП.
- *рациональность* (rationality) — способность оценивать результаты своих действий с целью нахождения оптимального решения.

- *адаптивность* (adaptability) — способность к обучению, результатом которого должно стать изменение внутреннего поведения ИА.
- *мобильность* (mobility) — способность агента к перемещению по сети.
- *правдивость* (veracity) — агент не способен сознательно предоставлять ложную информацию.
- *благожелательность* (benevolence) — агент готов помогать другим агентам, до тех пор пока это не противоречит его собственным целям, что подразумевает отсутствие у агента конфликтующих целей.
- *знания* (knowledge) — это постоянная часть знаний агента о себе, среде и других агентах, т. е. та часть, которая не изменяется в процессе его функционирования;
- *убеждения* (beliefs) — знания агента о среде, в частности, о других агентах. Это те знания, которые могут изменяться во времени и становиться неверными, однако агент может не иметь об этом информации и продолжать основываться на убеждении, что на их основе можно делать свои выводы;
- *желания* (desires) — это состояния, ситуации, достижение которых по разным причинам является для агента желательным, однако они могут быть противоречивыми и потому агент не ожидает, что все они будут достигнуты;
- *намерения* (intentions) — это то, что агент или обязан сделать в силу своих обязательств по отношению к другим агентам (ему «это» поручено и он взял эту задачу на себя), или то, что вытекает из его желаний (т. е. непротиворечивое подмножество желаний, выбранное по тем или иным причинам, и которое совместимо с принятыми на себя обязательствами);

- *цели* (goals) — конкретное множество конечных и промежуточных состояний, достижение которые агент принял в качестве текущей стратегии поведения;
- *обязательства по отношению к другим агентам* (commitments) — задачи, которые агент берёт на себя по просьбе (поручению) других агентов в рамках достижения кооперативных целей или целей отдельных агентов в рамках сотрудничества.

Первые два из перечисленных понятий называют «позицией агента», его «точкой зрения» (attitudes), остальные характеризуют в англоязычной литературе общим термином «*pro-attitude*», суть которого в том, что они «направляют» поведение агента таким образом, чтобы сделать отвечающие данному термину содержательные и формальные утверждения истинными.

Некоторые авторы считают, что агент должен обладать также рядом других свойств. К ним относятся [30]:

- *мобильность* (mobility) — способность агента мигрировать по сети в поисках необходимой информации для решения своих задач, при кооперативном решении задач совместно или с помощью других агентов и т. д.;
- *благожелательность* (benevolence) — готовность агентов помочь друг другу и готовность агента решать именно те задачи, которые ему поручает пользователь, что предполагает отсутствие у агента конфликтующих целей;
- *правдивость* (veracity) — свойство агента не манипулировать информацией, про которую ему заведомо известно, что она ложна;
- *рациональность* (rationality) — свойство агента действовать так, чтобы достигнуть своих целей, а не избегать их достижения, по крайней мере, в рамках своих знаний и убеждений.

Реализация вышеуказанных свойств определяет один из видов агентной архитектуры: *реактивную* (основанную на реакции агентов на воздействие окружающей среды), либо *делиберативную* (основанную на целях агента).

2.1.2. Реактивная архитектура

Общая структура простого реактивного агента представлена на [рис. 2.2](#). Реактивные агенты не имеют единого символьного представления об окружающей среде. Любые действия подобных агентов основываются на правилах вида «условие–действие» (продукционная модель¹). Базовая идея заключа-

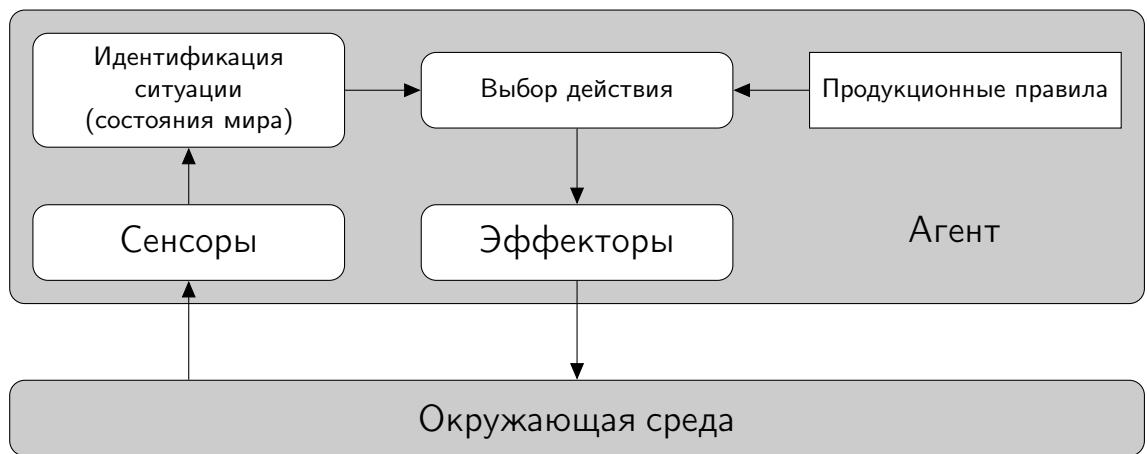


Рис. 2.2. Архитектура простого реактивного агента

ется в том, что интеллект агента является результатом его взаимодействия со средой, в которой он существует, т. е. возникает как композиция простых поведенческих схем, структурированных некоторым образом [31].

¹ Модель представления знаний, основанная на правилах вида «Если (*условие*), то (*действие*)». В общем случае продукционная модель может быть задана следующим образом:

$$i = \langle S, L, A \rightarrow B, Q \rangle, \text{ где} \quad (2.1)$$

- S описание класса ситуаций,
- L условие, при котором продукция активизируется,
- $A \rightarrow B$ ядро продукции,
- Q постусловие продукционного правила.

Архитектура реактивного агента строится на базе множества вариантов поведения («модулей поведения»), решающих некоторое множество задач без явного абстрактного логического вывода и символьного представления знаний [32].

К основным недостаткам реактивной архитектуры можно отнести:

- Необходимость полного ситуационного определения всех возможных действий агентов.
- Сложность создания обучающихся агентов.
- Сложность в проектировании агентов с многочисленными вариантами поведения (динамическим взаимодействием).

По мнению автора, несмотря на простоту и вычислительную разрешимость, в чистом виде данная архитектура не может быть использована при создании многоагентной системы технологического назначения. Учитывая хорошую проработку экспертных модулей уже много лет используемых для создания интеллектуальных АСТПП, отказ от символьного представления знаний и логического вывода нежелателен, а отсутствие единой модели среды при проектировании интеграционного окружения просто недопустимо.

2.1.3. Архитектура, управляемая целями

Архитектура, управляемая целями или архитектура *Убеждение–Желание–Намерение* (Belief–Desire–Intention – BDI) является наиболее перспективным видом делиберативной² архитектуры построения МАС. Данная архитектура, в первую очередь обеспечивает агенту механизм перераспределения времени между выбором плана действий и непосредственно выполнением данных действий, при этом само планирование остаётся за рамками рассмат-

² От английского *deliberate* — тщательно продумывать. Впервые термин был применён Майклом Генесеретом (Michael R. Genesereth) в работе [33].

риваемой модели. Таким образом архитектура BDI должна моделировать «практические рассуждения» человека в тех ситуациях, в которых решения приходится принимать ИА [34]. Следовательно, данная архитектура во многом строится по аналогии с процессами рассуждений и принятия решений специалистом. Структура BDI-агента представлена на рис. 2.3.

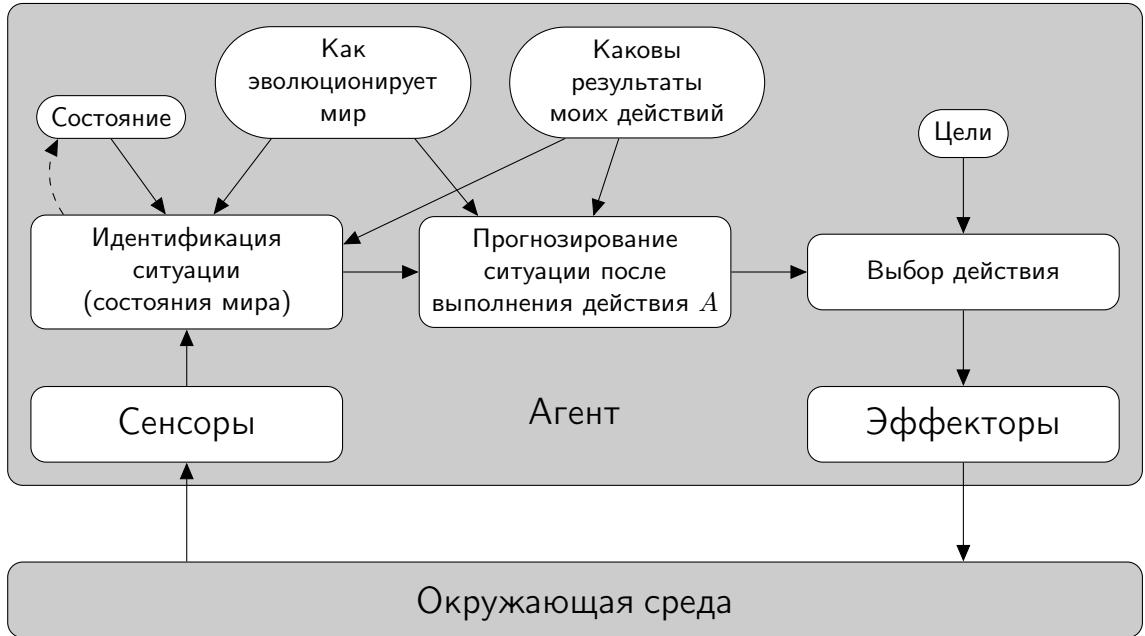


Рис. 2.3. Архитектура агента, управляемого целями

Подобное интеллектуальное поведение требует, чтобы BDI-агент обладал необходимыми знаниями, информацией о среде (убеждениями), а также мощными механизмами рассуждений о различных компонентах модели интеграционной технологической среды, которые позволяют ему динамически формировать своё поведение в соответствии с его базой знаний, намерениями и множеством целей. Формальные модели знаний, убеждений, а также соответствующие механизмы рассуждений строятся на основе логических языков [35].

Агентные убеждения соответствуют информации агента о мире и могут быть неполными и некорректными. Цели (намерения) агентов обычно соответствуют задачам, назначенным данным агентам в процессе проектирования.

Агенты не могут достичь всех своих целей, даже если они непротиворечивы, поэтому должны зафиксировать некоторое множество реальных целей и передать ресурсы для их достижения.

События, воспринимаемые агентом, помещаются в очередь событий. Внутренний интерпретатор агента непрерывно выполняет следующий цикл [36]:

1. просматривает многоагентную среду и своё внутреннее состояние, на основании полученных данных изменяет очерёдность событий;
2. генерирует новые возможные желания (задачи), находя те планы, чьи триггеры событий включены;
3. выбирает из этого множества планов один для выполнения;
4. помещает желаемое значение в существующий или новый стек, в соответствии с тем, имеется или нет подцель;
5. выбирает стек намерений, читает план, находящийся в вершине стека и выполняет следующий шаг из этого плана; если шаг есть действие — выполняет его, если это подцель — посылает эту подцель в очередь событий;
6. возвращается к шагу 1.

Внутреннее поведение агентов, построенных в соответствии с данной архитектурой практически полностью соответствует поведению специалистов в процессе интеллектуального поиска при проектировании ТП в условиях автоматизации, что несомненно позволяет использовать её при проектировании агентов многоагентной технологической системы. Тем не менее, архитектура BDI имеет и ряд недостатков:

1. Сложность создания полной и точной модели окружающей среды (может быть уменьшена за счёт следованию принципу стандартизации при построении автоматизированных систем технологической подготовки производства ([разд. 1.2](#))).
2. Возможность зацикливания в процессе логического вывода.

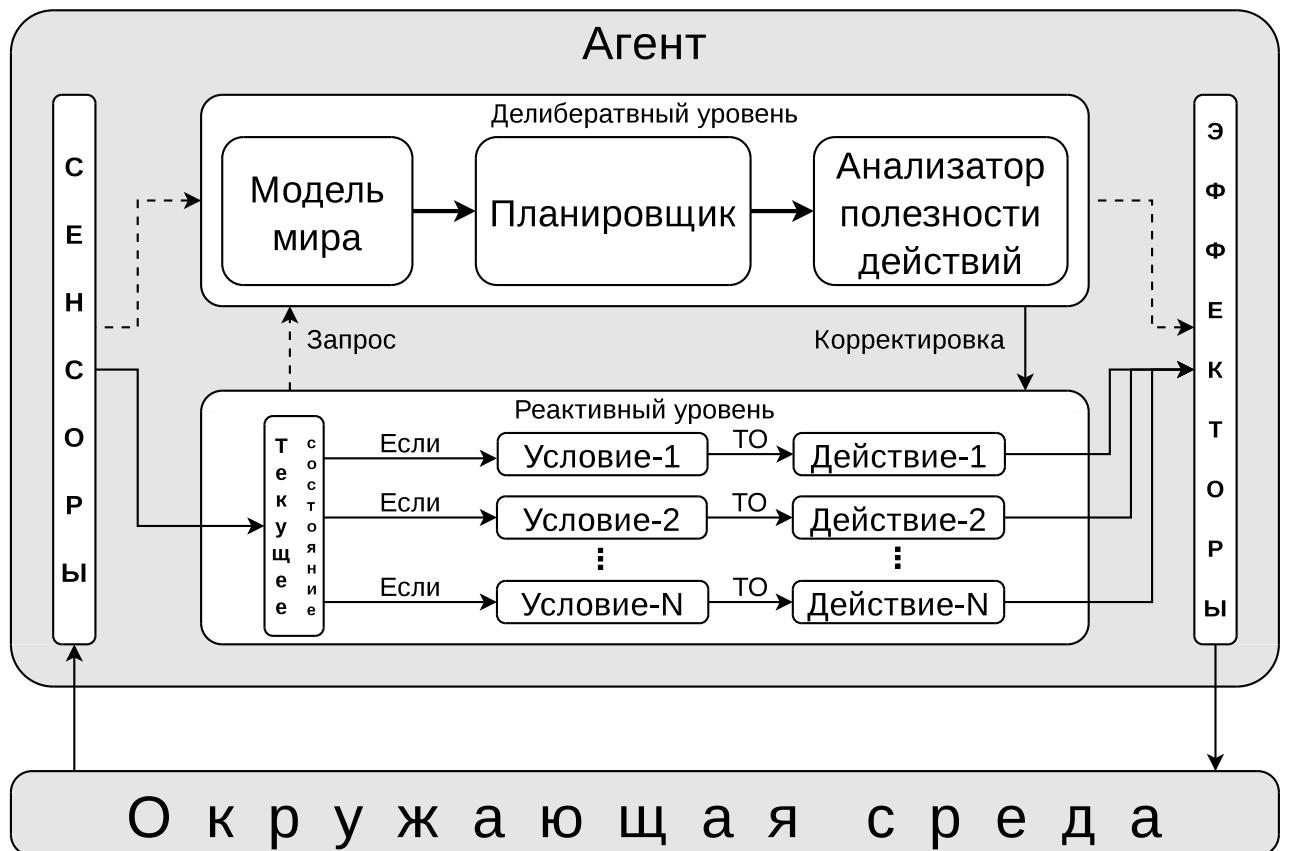


Рис. 2.4. Агент с гибридной архитектурой

3. Отсутствие способности агентов к анализу собственных действий и самообучению.

2.1.4. Гибридная архитектура

На основании всего вышесказанного можно сделать вывод о том, что наиболее предпочтительной агентной архитектурой при проектировании многоагентной системы технологического назначения должна стать архитектура, объединяющая в себе достоинства обоих подходов, т. е. *гибридная архитектура* (рис. 2.4). При проектировании производственных и технологических многоагентных систем именно эта архитектура является наиболее востребованной.

Гибридная архитектура позволяет строить агенты из двух модулей:

- BDI-модуля, содержащего символьную модель мира для принятия глобальных решений.
- Реактивного модуля для реагирования на происходящие в системе события.

Полученная в результате архитектура (рис. 2.4) является многоуровневой, т. е. подсистема контроля агента в ней состоит из двух уровней, при этом каждый вышележащий уровень работает с менее формализованной информацией.

К сожалению, и у гибридной архитектуры есть один ключевой недостаток — сложность организации взаимодействия между разными уровнями каждого агента. Следовательно, реализация многоагентной системы из агентов с подобной архитектурой — достаточно нетривиальная задача, требующая использования специализированных инструментальных средств разработки, подробнее описанных в гл. 3.

2.2. Описание математической модели

функционирования многоагентных систем

В разд. 3.1.2 было показано, что функциональностиprotoагентов явно недостаточно для создания полноценной многоагентной технологической системы. Понятиеprotoагента должно быть модифицировано, в первую очередь присвоением агенту некоторого количества поведенческих свойств. Также следует отказаться от единой логической машины вывода, передав её функции интеллектуальному модулю агента. Достигнутая таким образом децентрализация ведёт к необходимости более подробной проработке математической модели среды, в которой существуют агенты. По мнению автора понятие *многоагентной технологической среды* следует отделить от понятия *многоагентной технологической системы*. Среда представляет собой абстрактную сущность, которая может воздействовать на находящихся в ней агентов (protoагентов), и в которую агенты попадают извне. В многоагентной среде агенты не взаимодействуют друг с другом. В свою очередь, многоагентная система является многоагентной средой с дополнительным

механизмом межагентного взаимодействия, определённая в конкретной предметной области. Ниже все эти понятия будут раскрыты более подробно.

2.2.1. Многоагентные среды

Общие положения

Автор считает, что следует различать понятия многоагентной среды (в рассматриваемой предметной области — интеграционной технологической среды) и многоагентной системы, представляя первую в качестве некоторого абстрактного фундамента второй. Многоагентная среда (МС) определяется унифицированной математической моделью, позволяющей в дальнейшем перейти к рассмотрению холонических структур ([разд. 2.3.1](#)). Унификация достигается за счёт единобразности определений, позволяющей на основе информации о состоянии агента делать выводы о состоянии среды и наоборот.

Впервые подобная модель была описана в работе [37]. В этом разделе автором будет показано, что любая среда, содержащая множество агентов может быть изоморфно спроектирована на среду, в которой явно представлен всего один агент, в то время, как остальные агенты находятся за её пределами. Также будет показано как на основе заданной МС построить изоморфную ей среду, где группа агентов будет объединена для создания единого *холонического агента* и как провести *холоническую декомпозицию* (см. принцип декомпозиции, [разд. 1.2](#)), т. е. разделить единого агента на группу холонических агентов [38].

Введём понятие многоагентная среда. *Многоагентная среда* есть кортеж $(\mathcal{A}, \mathcal{E}, \Pi, \Delta)$, где $\mathcal{A} = \{\alpha_1, \dots, \alpha_n\}$ — множество всех агентов. Каждый агент α_i представляет собой кортеж (S_i, P_i, A_i, ϕ_i) множества возможных состояний S_i , множества объектов восприятия (перцепции) P_i , множества действий A_i и агентной функции $\phi_i: S_i \times P_i \rightarrow S_i \times A_i$. \mathcal{E} — множество

состояний среды. $\Pi: \mathcal{E} \rightarrow (P_1 \times \dots \times P_n)$ — функция восприятия, $\Delta: \mathcal{E} \times (A_1 \times \dots \times A_n) \rightarrow \mathcal{E}$ — функция среды.

Предполагается, что существует некоторая дискретная временная шкала, где временной шаг задаётся переходом от одной точки шкалы к другой. Любой агент α_i для всех состояний среды $e \in \mathcal{E}$ и всех состояний агентов $(s_1, \dots, s_n) \in S_1 \times \dots \times S_n$ на каждом шаге вычисления через функцию восприятия получает свой локальный объект восприятия $\Pi^i(e)$. Агент рассчитывает своё действие $a_i = \phi_i^2(s_i, \Pi^i(e))$ и своё новое состояние $s'_i = \phi_i^1(s_i, \Pi^i(e))$ на основании текущего состояния s_i и своего восприятия этого состояния,³ т. е. состояние среды меняется под действием агентов.

$$e' = \Delta(e, a_1, \dots, a_n) \quad (2.2)$$

определяет преемственное состояние среды, а

$$s'_i = \phi_i^1(s_i, \Pi^i(e)) \quad (2.3)$$

задаёт преемственное состояние агентов для всех i . Переходная функция состояния $\bar{\Delta}: \mathcal{E} \times S_1 \times \dots \times S_n \rightarrow \mathcal{E} \times S_1 \times \dots \times S_n$, определённая как $\bar{\Delta}(e, s_1, \dots, s_n) = (e', s'_1, \dots, s'_n)$, объединяет состояния агентов и состояния среды. Следовательно, функция восприятия, агентная функция и функция среды являются частями переходной функции $\bar{\Delta}$.

Две МС $(\mathcal{A}, \mathcal{E}, \Pi, \Delta)$ и $(\mathcal{A}', \mathcal{E}', \Pi', \Delta')$ изоморфны, если существует биективная функция $\Psi: \mathcal{E} \times S_1 \times \dots \times S_n \rightarrow \mathcal{E}' \times S'_1 \times \dots \times S'_n$ такая, что для всех $(e, s_1, \dots, s_n) \in \mathcal{E} \times S_1 \times \dots \times S_n$

$$\bar{\Delta}'(\Psi(e, s_1, \dots, s_n)) = \Psi(\bar{\Delta}(e, s_1, \dots, s_n)) \quad (2.4)$$

Следует обратить внимание на то, что у любой МС есть две специальные конфигурации: первая — среда без агентов, в которой все изменения состояния

³ Верхний индекс обозначает проекционную функцию Клини, например, $\Pi^i(e)$ — это P_i в области значений (P_1, \dots, P_n) функции Π .

кодируются функцией среды Δ ; вторая — МС с единственным неизменным состоянием и одним агентом, все изменения состояния которого кодируются агентной функцией ϕ . Подобная формализация позволяет редуцировать МС, содержащую несколько агентов до одноагентного случая, представляя всех агентов, кроме одного, в качестве субъектов среды.

На [рис. 2.5](#) представлено интуитивное представление подобной конструкции, которое будет явно декларировано ниже.

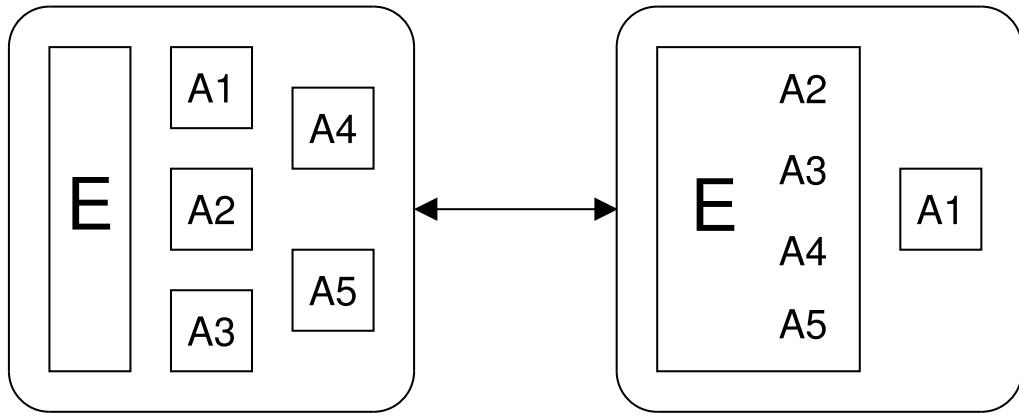


Рис. 2.5. Преобразование многоагентной среды в одноагентную

Если $E = (\{\alpha_1, \dots, \alpha_n\}, \mathcal{E}, \Pi, \Delta)$ — многоагентная среда, то для каждого $i \leq n$ существует изоморфная многоагентная среда $(\{\alpha_i\}, \mathcal{E}', \Pi', \Delta')$. Не нарушая общности рассуждения, возьмём $i = 1$. Создадим среду $E' = (\{\alpha_i\}, \mathcal{E}', \Pi', \Delta')$ такую, что $\mathcal{E}' = \mathcal{E} \times S_2 \times \dots \times S_n$, $\Pi' = \Pi^1$, для всех $(e, s_2, \dots, s_n) \in \mathcal{E}'$ и $a_1 \in A_1$ функция среды $\Delta': \mathcal{E}' \times A_1 \rightarrow \mathcal{E}'$ задаётся как:

$$\Delta'((e, s_2, \dots, s_n), a_1) = (e', s'_2, \dots, s'_n), \quad (2.5)$$

где $e' = \Delta(e, a_1, \phi_2^2(s_2, \Pi^2(e)), \dots, \phi_n^2(s_n, \Pi^n(e)))$ — преемственное состояние e в E для a_1 и остальных действий агентов $\phi_i^2(s_i, \Pi^i(e))$. $s'_i = \phi_i^1(s_i, \Pi^i(e))$ — преемственное состояние s_i в E для $2 \leq i \leq n$. Из чего непосредственно и следует свойство изоморфизма.

Также необходимо ввести понятия холонического объединения и декомпозиции (подробнее о холоническом подходе см. разд. 2.3), а также показать, как объединить множество агентов в холон (рис. 2.6).

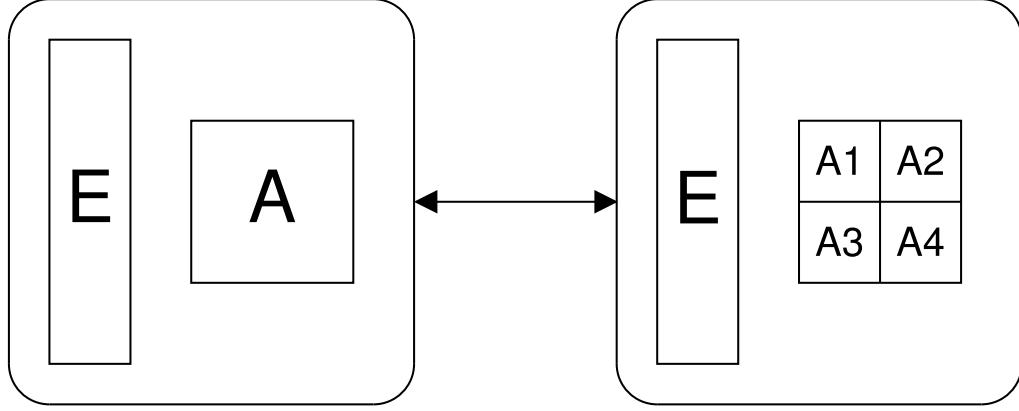


Рис. 2.6. Холоническое объединение

Рассмотрим две изоморфные МС: $(\{\alpha_1, \dots, \alpha_n\}, \mathcal{E}, \Pi, \Delta)$ и $(\{\alpha_1, \dots, \alpha_{i-1}, \alpha_{i,1}, \dots, \alpha_{i,m}, \alpha_{i+1}, \dots, \alpha_n\}, \mathcal{E}, \Pi', \Delta')$, где α_i есть холоническое объединение $(\alpha_{i,1}, \dots, \alpha_{i,m})$. Для любой МС $(\{\alpha_1, \dots, \alpha_n\}, \mathcal{E}, \Pi, \Delta)$ и $k \leq n$ можно создать изоморфное многоагентную среду $(\{\alpha', \alpha_{k+1}, \dots, \alpha_n\}, \mathcal{E}, \Pi', \Delta')$, где α' есть холоническое объединение агентов $\alpha_1, \dots, \alpha_k$.

Создадим среду $(\{\alpha', \alpha_{k+1}, \dots, \alpha_n\}, \mathcal{E}, \Pi', \Delta')$ с агентом $\alpha' = (S', P', A', \phi')$, который эмулирует агентов $\alpha_1, \dots, \alpha_k$ и адаптируем функцию восприятия и функцию среды следующим образом: $S' = (S_1 \times \dots \times S_k)$, $P' = (P_1 \times \dots \times P_k)$, $A' = (A_1 \times \dots \times A_k)$, в то время как для всех $s' = (s_1, \dots, s_k) \in S'$ и $p' = (p_1, \dots, p_k) \in P'$ агентная функция будет определяться как:

$$\phi'(s', p') = ((\phi_1^1(s_1, p_1), \dots, \phi_k^1(s_k, p_k)), (\phi_1^2(p_1, s_1), \dots, \phi_k^2(p_k, s_k))), \quad (2.6)$$

где первая (вторая) валентность ϕ' есть кортеж первых (вторых) валентностей составных агентных функций. Кроме того, можно определить функцию восприятия $\Pi': \mathcal{E} \rightarrow (P', P_{k+1}, \dots, P_n)$ для всех $e \in \mathcal{E}$ следующим образом:

$$\Pi'(e) = ((\Pi^1(e), \dots, \Pi^k(e)), \Pi^{k+1}(e), \dots, \Pi^n(e)) \quad (2.7)$$

В данном случае восприятие агента α' состоит из восприятия агентов $\alpha_1 \dots \alpha_k$. Наконец, функция окружения $\Delta': \mathcal{E} \times A' \times A_{k+1} \times \dots \times A_n$ определяется как:

$$\Delta'(e, a', a_{k+1}, \dots, a_n) = \Delta(e, a_1, \dots, a_n) \quad (2.8)$$

для всех $e \in \mathcal{E}$, $a' = (a_1, \dots, a_k) \in A'$, $a_{k+1}, \dots, a_n \in A_n$. Из чего непосредственно и следует возможность холонического объединения и декомпозиции, что позволяет представлять множество агентов в виде одного суперхолона и редуцировать суперхолон до одноагентного состояния.

Алгебраические свойства операции холонического объединения

Основная идея холонического объединения заключается в возможности *слияния* нескольких субхолонов в один суперхолон, который можно представить с помощью операции последовательного выполнения (композиции):

$$(H_1 \otimes H_2), \quad (2.9)$$

где каждый из холонов H_1 и H_2 связан с некоторой агентной функцией $\phi_i: S_i \times P_i \rightarrow S_i \times A_i$, $i = 1, 2$.

Пусть I — единичный холон, дающий тождественное отображение, т. е.:

$$\forall H: I \otimes H = H \otimes I = H \quad (2.10)$$

Операция холонического объединения ассоциативна, т. е. $\forall H_1, H_2, H_3$:

$$(H_1 \otimes H_2) \otimes H_3 = H_1 \otimes (H_2 \otimes H_3) \quad (2.11)$$

Данное утверждение следует непосредственно из определений состояния среды и того факта, что прямое произведение, семантика которого используется, ассоциативно по определению. Можно сделать вывод, что (H_i, I, \otimes) —

моноид. Также следует отметить, что операция холонического объединения коммутативна, т. е.:

$$H_1 \otimes H_2 = H_2 \otimes H_1, \quad (2.12)$$

но не обладает свойством идемпотентности, т. е.:

$$H \otimes H \neq H \quad (2.13)$$

Последнее положение легко доказывается на основании представления состояния среды в виде кортежа (s, s) , отличного от единичного состояния s , иными словами, два технологических объекта (в данном случае, холона технологических объектов), будучи объединены в один, приобретают новые свойства. Из этого следует, что используемая для определения многоагентной технологической среды алгебраическая структура (H_i, I, \otimes) есть моноид Абеля, неидемпотентная по определению.

2.2.2. Многоагентные системы

Общие положения

Многоагентная система (МАС) — вычислительная система, в которой два или более агента (холона) взаимодействуют (сотрудничая, соперничая или комбинируя первое и второе), чтобы достичь определённых индивидуальных или коллективных целей, находящихся за пределами индивидуальных способностей и знаний каждого агента.

Теория многоагентных систем занимается исследованием координации взаимодействия группы автономных интеллектуальных агентов (существующей, либо вновь созданной). Анализируется индивидуальное поведение каждого агента и его вклад в поведение системы в целом. Основными направлениями исследований являются модели поведения агентов внутри МАС, стратегии сотрудничества и кооперации, интеллектуальное посредничество,

задачи оптимизации производительности, самообучение, формирование коалиций и т. д.

Таким образом, МАС является программной технологией общего назначения, базирующейся на фундаментальных исследованиях автономного поведения, интеллектуальной кооперации, теории формирования групп и др. Эта технология находит самое широкое применение в тех областях, в которых требуется взаимодействие некоторых сущностей в сложной гетерогенной среде, следовательно может быть применена для совершенствования механизмов взаимодействия программных средств в рамках комплексной автоматизации технологической подготовки производства.

Существует несколько математических моделей, описывающих многоагентные системы. Наиболее полная из них описана в работах Майкла Вулдриджа⁴ [30, 38, 40, 41]. В данной модели многоагентная среда никак не отделяется от многоагентной системы, являясь её неотъемлемой частью. Автором же будет рассмотрена более простая модель, впервые предложенная в работе [42], и лучше вписывающаяся в предлагаемую концепцию, где МС является абстрактным фундаментом, а МАС — коммуникационной надстройкой.

Фундаментальное определение многоагентных систем

Для любой программной системы принято отделять её статическое определение (технические требования) от динамического экземпляра времени исполнения (runtime instance). В то время как концепции и теории описания технических требований к программному обеспечению достаточно хорошо описаны и документированы, методы описания и анализа

⁴ Очень интересная математическая модель многоагентных систем представлена в работе [39], в ней МАС описывается с помощью теории формальных языков и грамматик. В силу своей специфики данная работа рассматриваться не будет.

динамического поведения программных систем всё ещё плохо изучены. Особенno остро эта проблема проявляется в отношении многоагентных систем, в которых самоорганизация в процессе исполнения является ключевым аспектом. Это позволяет выделить многоагентные системы в отдельный класс, для которого неприменимы традиционные парадигмы разработки программного обеспечения.

Предположим, что есть некоторая сущность, позволяющая агентам самостоятельно изменять свою структуру (самоорганизовываться). Например, общеизвестные стандарты FIPA ([разд. 3.4](#)) описывают именно такую агентную модель. В спецификациях FIPA определена *Агентная служба каталога* (Agent Directory Service — ADS), позволяющая агентам взаимодействовать друг с другом. Данная служба обязана предоставлять агентам как минимум сервис «*белых страниц*», где агенты могут узнавать адреса других агентов. Дополнительно у ADS есть сервис «*жёлтых страниц*», хранящий информацию о всех службах, предоставляемых участниками многоагентной системы. В проектируемой системе службами (агентными сервисами) являются средства информационного обеспечения АСТПП (подробнее см. [разд. 2.2.3](#)).

Для описания многоагентной системы в рассматриваемой предметной области определяется множество агентов-прототипов, созданных в соответствии с принципами организации многоагентных сред. Следовательно, можно дать следующее статическое определение многоагентной системы:

$$\mathcal{MAS}_{prot} = (\mathcal{A}_{prot}, ADS), \text{ где} \quad (2.14)$$

- | | |
|----------------------|---|
| \mathcal{A}_{prot} | множество агентов-прототипов $\{A^1, \dots, A^n\}$, $n \in \mathbb{N}$, экземпляры которых могут быть динамически включены в систему. |
| ADS | специализированный агент-прототип, реализующий агентную службу каталога. |

Объекты конечного множества (2.14) формируют МАС в определённой предметной области. Процесс решения любой задачи в рамках МАС начинается с инициализации одного из агентов системы:

$$\begin{aligned} \mathcal{MAS}_{init} &= (\mathcal{A}_{init}, ADS_{init}), \text{ где} \\ \mathcal{A}_{init} &= \{A_1^1, \dots, A_{k_1}^1, \dots, A_1^n, \dots, A_{k_n}^n\}, k_1, \dots, k_n \in \mathbb{N} \text{ и} \quad (2.15) \\ \forall A_j^i \in \mathcal{A}_{init} : & A^i \triangleright A_j^i \wedge A^i \in \mathcal{A}_{prot}. \end{aligned}$$

Выражение $A^i \triangleright A_j^i$ (читается « A_j^i — экземпляр A^i ») показывает, что A_i^j является экземпляром агента-прототипа A_i . Агент A_i^j наследует поведение и все изначальные знания агента-прототипа, а также может обладать некоторыми дополнительными свойствами или знаниями (например, уникальным идентификатором, позволяющим другим агентам многоагентной системы взаимодействовать с ним).

Окружение МАС открыто, поэтому можно предположить, что ADS_{init} представляет некоторый существующий каталог ADS , который находился в состоянии ADS_{init} в то время, когда был запущен первый агент. Продолжая эту линию рассуждения, можно также предположить, что не все агенты из множества A_{init} были разработаны программистами, некоторые из них уже существовали в обозначенной выше открытом окружении. На следующем этапе рассматриваемая модель может быть дополнена множеством подобных открытых агентов $\mathcal{A}_{open} = \{A_1^1, \dots, A_{k_1}^1, \dots, A_1^m, \dots, A_{k_n}^m\}, 1 \leq m < n$. Поэтому описанная спецификация агентов-прототипов, вероятно, является неполной в том смысле, что программисту, разрабатывающему \mathcal{A}_{prot} будет нужна информация только о $\{A^1, \dots, A^m\}$, остальные же агенты множества \mathcal{A}_{open} просто будут пользоваться сервисами этих агентов.

Из \mathcal{MAS}_{init} можно вывести определение динамической среды \mathcal{MAS}_t :

$$\begin{aligned} \mathcal{MAS}_t &= (\mathcal{A}_t, ADS_t), \text{ где} \\ \mathcal{A}_t &= \{A_1^{1,t}, \dots, A_{l_1}^{1,t}, \dots, A_1^{n,t}, \dots, A_{l_n}^{n,t}\}, l_1, \dots, l_n \in \mathbb{N} \text{ и} \\ \forall A_j^{i,t} \in \mathcal{A}_t : & A^i \blacktriangleright A_j^{i,t} \wedge A^i \in A_{prot}. \end{aligned} \quad (2.16)$$

Операция \blacktriangleright (читается «*преобразуется в*»), являющаяся сокращённой записью выражения $\langle \triangleright \rangle \circ \langle \rightsquigarrow^* \rangle$ и означающая, что мы имеем $A^i \triangleright A_j^i$, где $A_j^i \in \mathcal{A}_{init}$ и $A_j^i \rightsquigarrow^* A_j^{i,t}$, где \rightsquigarrow^* обозначает, что преобразование A_j^i происходит за один шаг вычисления.

Работа МАС базируется на взаимодействии агентов. В систему в любое время могут быть введены новые агенты, а старые убраны из неё. Каждый агент имеет уникальный адрес, который может быть зарегистрирован агентом ADS и стать доступным для всех агентов МАС. Адрес агента ADS известен всем агентам изначально.

2.2.3. Абстрактные ресурсы и их роль в МАС

Общие положения

Концепция *абстрактных ресурсов* является часто используемым обобщением в теории информации, где под ресурсами обычно подразумеваются процессорное время и объём используемой памяти. Для МАС термин абстрактный ресурс используется для обозначения *любых инструментов и средств многоагентного окружения, которые помогают агенту достичь наивысшей полезности* [37, 38]. Ресурсы (например, информация, восприятие, умение) могут усовершенствовать возможности агента, а их отсутствие — существенно ограничить их. Это понятие включает в себя не только такие «классические» ресурсы, как пространство и время, но и, например, информацию, восприятие или другие средства и умения агента. Управление ресурсами

осуществляется с помощью традиционного механизма, именуемого *семафором*. Архитектура семафора предполагает, что только один агент может обладать ресурсом, позволяющим ему действовать (участвовать в процессе вычисления).

Для суперхолона умения и возможности его субхолонов являются абстрактными ресурсами. Отсюда следует, что в холоническом сообществе абстрактными ресурсами могут быть также умения *субагентов*, из которых состоит холон, *количество агентов* (т. е. субсубхолонов), выполняющих конкретные задачи, *различные коммуникационные протоколы* и т. д.

Ресурсы, выделяемые суперхолону, могут быть реструктурированы (разделены на небольшие части) и перераспределены между его субхолонами. Данное распределение можно рассматривать как некоторую «директиву», определяющую дальнейшее поведение субхолонов.

Формальное определение ресурсов

Каждому состоянию МАС необходимо присвоить фиксированные значения, определяющие, что для агентов внутри МАС некоторые состояния более предпочтительны, чем другие. От того, как будет рассчитано данное вспомогательное значение (именуемое *функцией полезности*) для каждого конкретного агента, зависят его локальные перспективы развития, а следовательно, это значение субъективно и, возможно, даже ошибочно.

Как было показано выше, мы можем редуцировать многоагентную среду до одноагентного состояния. Следовательно, для определения функции полезности достаточно рассмотреть одноагентное окружение $E = (E_1, \dots, E_n)$ с агентом $\alpha = (S, P, A, \phi)$.

Функция полезности есть отображение $u: E \times S \rightarrow \mathbb{R}$, где $u(e)$ выражает значение полезности действия агента α , находящегося в ситуации e . Ранее,

в разделе 2.2.1, состояние среды было задано в виде кортежа независимых подсостояний. Представляя ресурсы как часть МАС, можно идентифицировать определённые подмножества подсостояний как ресурсы.

Ресурсы, являющиеся частью МАС, позволяют агенту достичь состояния, когда его полезность станет выше. В частности, нас интересует последовательность действий способствующая достижению наивысшей полезности, и мы ищем ресурсы, которые необходимы для успешного выполнения этой последовательности (рис. 2.7). Далее будет рассмотрен положительный случай (важные ресурсы доступны); негативный случай (ресурсы не доступны) задан в скобках.

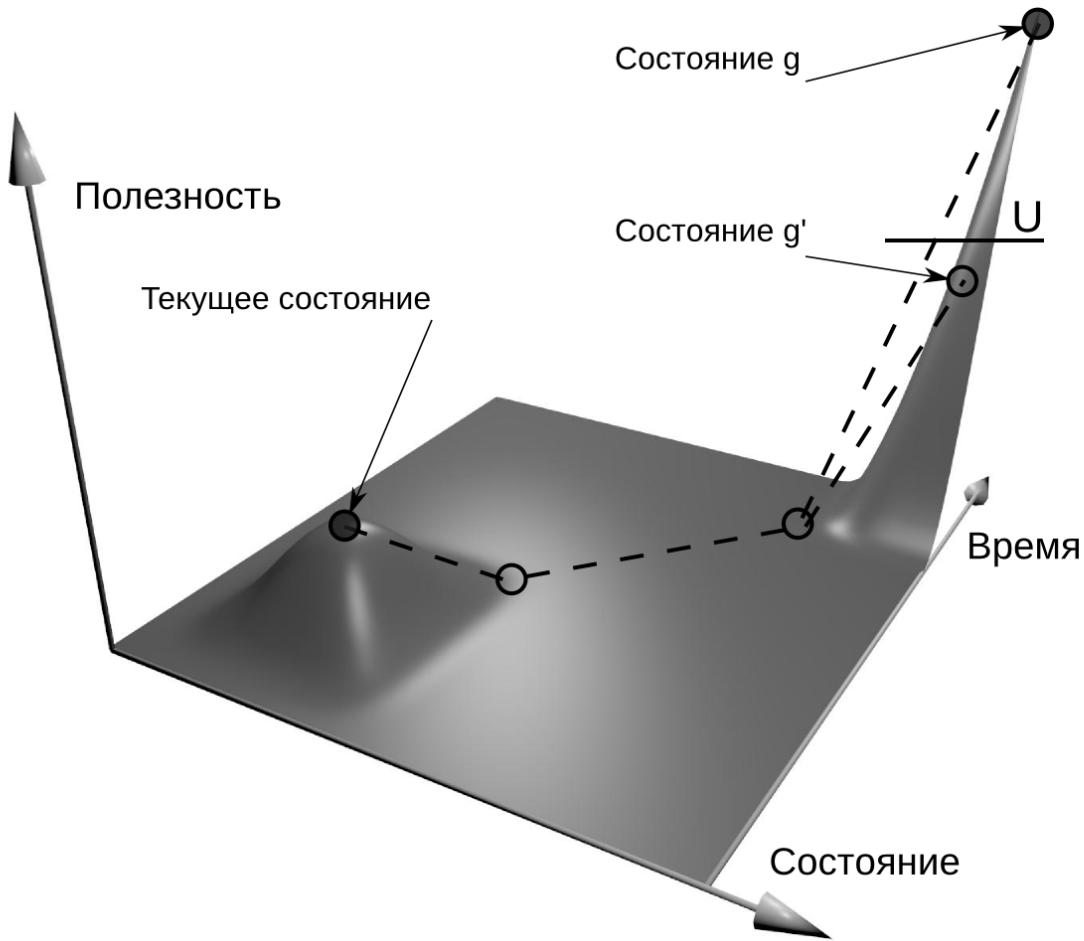


Рис. 2.7. Значение полезности для различных состояний среды

Пусть $e = (e_1, \dots, e_i, \dots, e_n, s) \in E \times S$ — состояние среды, а $(u: E \times S \rightarrow \mathbb{R})$ — функция полезности. Ресурс E_i является частью состояния среды $E_1 \times \dots \times E_n \times S$ по отношению к U , если:

- существует $U \in \mathbb{R}$ и $U > u(e)$;
- существует последовательность действий $a_1, \dots, a_n \in A^*$, которая переводит (не переводит) агента a из состояния e в состояние g с $u(g) \geq U$;
- существует другое состояние $e'_i \in E_i: e' = (e_1, \dots, e'_i, \dots, e_n, s)$, при котором агент a не сможет (сможет) перейти в состояние g' с $u(g') \geq U$ при той же последовательности действий.

Холоны обладают теми же основными характеристиками, что и агенты: автономность, стремление к сотрудничеству и открытость (возможность реорганизации). Но в дополнение к этим «поведенческим» характеристикам, у холонов есть и «структурные свойства». Одним из них является свойство *Рекурсивности*, которое подразумевает возможность создавать холоны с иерархической структурой, т. е. холоны могут состоять из холонов, которые в свою очередь созданы из других холонов и так вплоть до того момента, пока дальнейшее деление характеристик уже не возможно, либо бессмысленно. Другим важным свойством холонов является объединение модуля обработки данных с необязательным модулем управления физическими объектами ([разд. 2.3.3](#)).

В следующих разделах все основные характеристики холонов будут рассмотрены более подробно.

2.3. Холонический подход к решению задачи

технологической подготовки сложных производств

Очевидно, что для построения полноценной многоагентной системы технологической подготовки приборостроительного производства недостаточно выбрать архитектуру агента и описать математическую модель его поведения в среде. Разрабатываемая МАС должна решать задачи в конкретной предметной области, где недостаточно формального определения её базовых компонентов — требуется детализация.

Всем известно, что любая технологическая система (особенно опирающаяся на достижения современных информационных технологий) является очень крупным и сложным объектом, плохо поддающимся управлению и прогнозированию. Кроме того, чтобы решать проблемы современного научоёмкого производства эта система должна удовлетворять таким фундаментальным требованиям как:⁵

- Внутрипроизводственная интеграция.
- Распределённое управление.
- Неоднородность технологической среды.
- Технологическая совместимость.
- Открытая и динамическая структура производства.
- Возможность работы в кооперативной среде.
- Эффективное взаимодействие персонала с современными программными и аппаратными средствами производства.
- Гибкость.

⁵ Не следует путать данные требования с основными принципами разработки автоматизированных систем технологической подготовки производства, описанные в [разд. 1.2](#), т. к. рассматриваемые требования относятся ко всем этапам проектирования технологии безотносительно степени их автоматизации.

- Масштабируемость.
- Отказоустойчивость.

Существуют несколько парадигм организации технологической подготовки производств, удовлетворяющих всем этим требованиям. Две из них: *Распределенные интеллектуальные производственные системы* (Distributed Intelligent Manufacturing Systems – DIMS) и *Холонические производственные системы* (Holonic Manufacturing Systems – HMS) небезосновательно заслуживают особого внимания со стороны промышленности и науки [43].

В последнее время исследователи стали применять агентные технологии в самых разных областях промышленного производства начиная от производственной кооперации в рамках виртуальных предприятий [44, 45], управления внутри- и внешнепроизводственными цепочками поставок [46, 47], планирования производства и заканчивая управлением автоматическими складами, мобильными промышленными роботами [48, 49] и учётом запасов. Но основным приложением агентных технологий в промышленном производстве все же является использование гетерархической архитектуры в качестве парадигмы управления.

Гетерархическая система управления представляет собой плоскую структуру, состоящую из независимых сущностей (агентов). Эти агенты обычно предоставляют некоторые ресурсы и/или решают определённые задачи. Выбор конкретных ресурсов, необходимых для решения конкретных задач осуществляется на основе динамических рыночных механизмов. Такой подход позволяет создавать простые и отказоустойчивые системы, т. к. ни одному из агентов не требуется никакой априорной информации о других агентах. Как следствие, небольшие отклонения в работе любого из агентов могут быть легко исправлены.

Тем не менее, некоторые допущения, положенные в основу данной парадигмы не позволяют использовать её в технологической подготовке производства. Автономность агентов запрещает им пользоваться глобальными данными, без которых невозможно проектирование ТП. Таким образом, общая производительность системы сильно зависит от существующих правил взаимодействия агентов. Для решения этой проблемы в начале 90-х годов консорциум *Интеллектуальных Производственных Систем* (Intelligent Manufacturing Systems, IMS) [50, 51] инициировал ряд проектов, направленных на создание парадигмы «предприятий будущего». Одним из этих проектов были *Холонические производственные системы*, положения которой должны расширить возможности проектируемой интеграционной среды автоматизации технологической подготовки производства.

2.3.1. Холон

Термин *холон*, уже упоминавшийся в разд. 2.2.1 при описании математической модели многоагентной технологической среды, впервые был использован британским писателем и журналистом Артуром Кёстлером (Arthur Koestler) для объяснения эволюции биологических и социальных систем. С одной стороны, эти системы создают устойчивые промежуточные формы, остающиеся самостоятельными в процессе своего развития [52]. С другой — в организации подобных систем очень трудно провести черту между «целым» (wholes) и «частями» (parts): почти всё целое может в тоже время быть и частью чего-то другого.

Эти наблюдения подтолкнули Кёстлера к созданию нового термина «холон», который представляет собой сочетание греческого слова «*holos*», означающего целое и суффикса «*on*», обычно использующегося в названиях частиц (например, *протон*, *нейтрон* и т. д.). Кёстлер отмечал, что в пол-

ностью независимых живых организмах и общественных организациях нет невзаимодействующих сущностей. Каждая уникальная единица, например клетка в организме или семья в обществе, состоит из более мелких единиц (плазмы и ядра, родителей и детей), в то же время являясь составной частью более крупной единицы (мышечной ткани или человеческого социума).

Основным достоинством холонических организаций, или *холархий*, является возможность создания очень сложных систем, которые, тем не менее, могут эффективно использовать ресурсы, имеют очень высокую устойчивость к помехам (как внутренним, так и внешним) и обладают способностью быстро адаптироваться к изменениям окружающей среды. Холоны в холархии могут динамически создавать и изменять иерархические связи. Более того, холоны могут участвовать в нескольких иерархиях одновременно. Холархии обладают свойством рекурсивности в том смысле, что холон может сам по себе быть холархией, которая действует как автономная и кооперативная единица в холархии более высокого порядка [53].

Высокая стабильность холонов и холархий основывается на их полной самостоятельности, а также умении решать свои проблемы без обращения за помощью к холонам высших уровней. Холоны могут получать их указания и, в определённой степени, контролироваться ими. Однако, полная самостоятельность и автономность холонов позволяет им оставаться нечувствительными к возмущениям среды, а подчинённость вышестоящим холонам обеспечивает эффективную работу системы в целом [54–57].

2.3.2. Холонические производственные системы

Концепция *Холонических производственных систем* (Holonic Manufacturing Systems, HMS) была предложена японским исследователем Хироюки Суда (Hiroyuki Suda) в 1989 году [58]. В 1992 году несколько независимых

команд отраслевых экспертов, учёных и инженеров приступили к разработке фреймворка⁶ для международного сотрудничества в области *Интеллектуальных Производственных Систем*. К февралю 1992 года было создано технико-экономическое обоснование проектируемого подхода, которое показало, что его использование позволит добиться значительных результатов за относительно короткие сроки.

Холонические производственные системы основываются на концепции «холонических систем» (разд. 2.3.1). Холоны в ХПС содействуют оператору, работающему с некоторым объектом технологического процесса, подбирая наиболее подходящие параметры, строя свои собственные стратегии и взаимодействуя с другими холонами. Программная реализация ХПС представляет производственную систему как единую среду, состоящую из автономных модулей (холонов) с децентрализованным управлением [59–62]. Основная цель подобной модели — получить преимущества, которые даёт холоническая структура живым организмам, а именно: гибкость, устойчивость к негативному воздействию среды и эффективное использование имеющихся ресурсов. Концепция ХПС сочетает в себе лучшие черты иерархических и гетерархической моделей, сохраняя стабильность иерархии, обеспечивая динамическую гибкость гетерархии (рис. 2.8).

Консорциумом ХПС был разработан следующий перечень определений, позволяющий применять данный подход в производстве и технологии:

- *Холон* — автономный, способный к взаимодействию строительный блок производственной системы, используемый для преобразования, передачи, хранения и проверки информации от потоков управления или физических объектов. Холон может быть частью другого холона.

⁶ англ. *framework*, син. каркас — специализированное программное обеспечение, позволяющее облегчить разработку и объединение разных слабосвязанных компонентов крупного программного проекта.

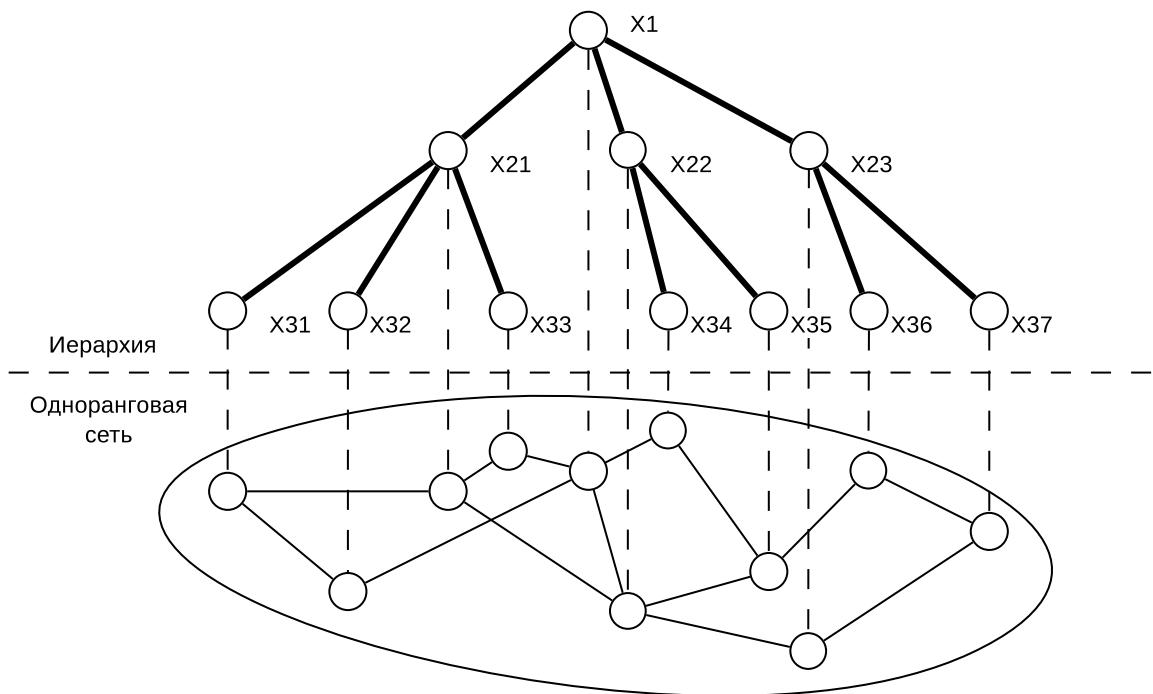


Рис. 2.8. Общая структура многоуровневой холонической системы

- *Автономия* — способность некоторой сущности планировать свои действия и контролировать процесс достижения своих целей.
- *Кооперация* — процесс, в котором несколько независимых объектов создают взаимоприемлемые планы и действуют в соответствии с ними.
- *Холархия* — группа холонов, сотрудничающих для достижения некоторой совместной цели. Холархия определяет основные правила взаимодействия холонов, тем самым ограничивая их автономию.
- *Холоническая производственная система* — холархия, объединяющая все аспекты производства и технологии начиная с проектно-конструкторских работ и заканчивая промышленным освоением новых изделий.

Последнее утверждение даёт возможность при необходимости на время отказаться от предложенной одноранговой модели взаимодействия агентов внутри проектируемой ИУП ТПП, вернувшись к классической иерархической модели, что позволит более гибко конфигурировать разрабатываемую систему, подстраивая его под нужды конкретной технологической задачи.

2.3.3. Современное состояние ХПС

За последние 10 лет были проведены исследования возможности применения ХПС в реальных промышленных условиях. В этом разделе будут рассмотрены основные направления этих исследований, такие как:

- Архитектура холонов.
- Взаимодействие холонов.
- Функционирование холонов.
- Алгоритмы холонического управления.
- Методология разработки ХПС.

Архитектура холонов

Автоматизированные системы технологической подготовки производства (АСТПП) состоят из программных модулей, взаимодействующих с различными физическими объектами производственной среды: ресурсами, заказами, персоналом и т. д. Программные модули и физические объекты связаны в единую информационную сеть, являющуюся холоном производственной системы. Каждый холон может рассуждать, принимать решения и общаться с другими холонами в режиме реального времени. Количество и типы программных модулей и способ их взаимодействия с физическими объектами определяют архитектуру холонов [40, 63–66].

Общая структура холона впервые была предложена Джеймсом Кристенсеном (James H. Christensen) в 1994 году [67]. На рис. 2.9 представлены два основных компонента этой архитектуры: модуль взаимодействия с информационным потоком и модуль взаимодействия с материальным потоком (необязателен). Примерами холонов не взаимодействующих с материальным потоком являются, например, холоны поиска заказов, холоны планирования и т. д.

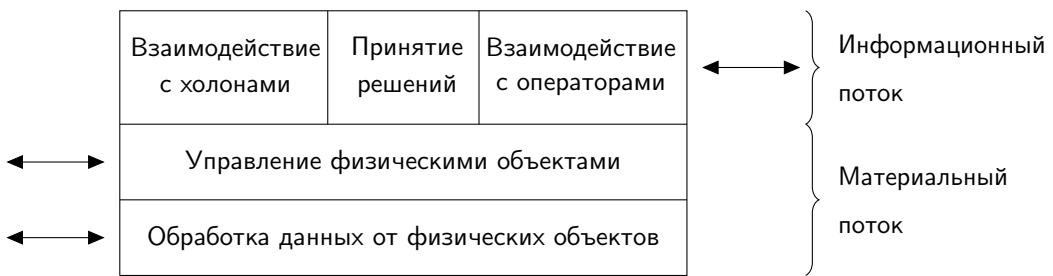


Рис. 2.9. Общая структура холона

Модуль взаимодействия с материальным потоком состоит из двух частей: непосредственно обработки данных от физических объектов и управления аппаратными средствами (например оборудованием с ЧПУ). Модуль взаимодействия с информационным потоком состоит из трех частей: ядра холона, отвечающего за принятие решений, интерфейса взаимодействия с другими холонами и интерфейса с операторами ХПС, позволяющего холону получать команды и выводить результат своей работы.

Агентная архитектура модуля взаимодействия с информационным потоком у Кристенсена базируется на холоническом представлении автономных и взаимодействующих сущностей. Это представление включает в себя три основных аспекта. Во-первых, холоны — есть автономные сущности, поведение которых полностью определяется объектами, с ними связанными. Холоны могут заниматься планированием и выбором стратегии своей работы. Автономное поведение подразумевает некий модуль принятия решений, позволяющий холону управлять физическими объектами. Во-вторых, в случае необходимости два или более холонов могут взаимодействовать между собой. При этом холоны оценивают возможности сотрудничества, создают правила взаимодействия и согласования и, наконец, принимают решение о сотрудничестве. В-третьих, холоны могут участвовать в нескольких квазиустойчивых организациях, именуемых *холархиями*. Создание холархии подразумевает объединение нескольких разрозненных частей технологическо-

го или производственного процесса с целью повышения производительности. Это означает разделение выполняемых задач и ответственности за их выполнение, а также определение модели взаимодействия, позволяющей холонам реорганизовываться, меняя общий вектор выполняемых работ.

Включение этих компонентов в общую структуру Кристенсена даёт нам новую агент-ориентированную архитектуру, представленную на [рис. 2.10](#). Для

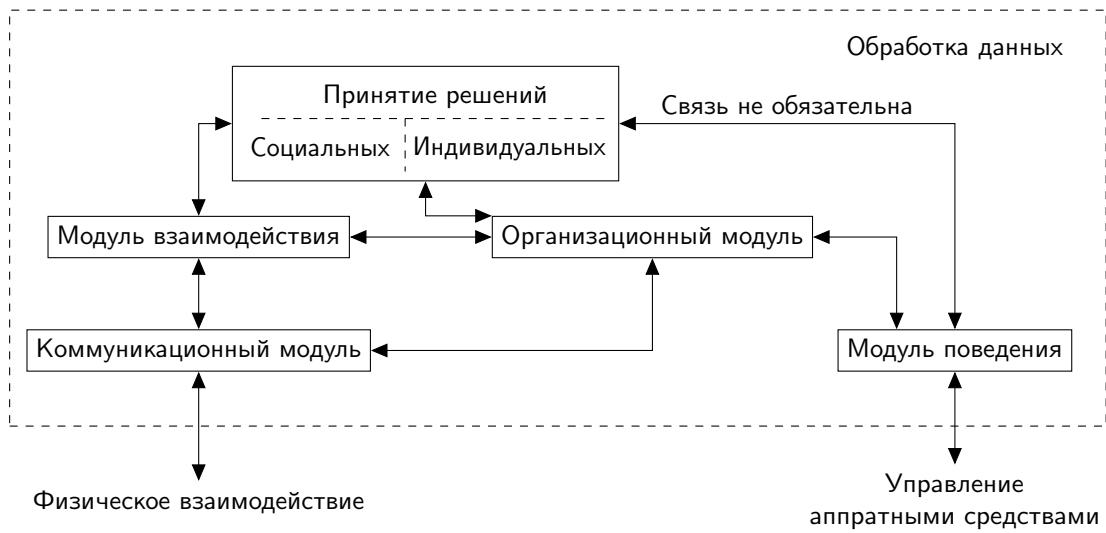


Рис. 2.10. Агент-ориентированная структура холона

получения информации о поведении среды и оценки текущей ситуации, холон вырабатывает определённый план, позволяющий ему в будущем достичь своих целей. Из *модуля принятия решений* этот план передаётся в *модуль контроля поведения*, где переводится в набор управляющих команд, которые передаются непосредственно *аппаратным средствам*. Одновременно с этим, этот же план попадает в *модуль взаимодействия*, где с помощью специальных методов преобразуется в набор коммуникационных команд, которые могут быть переданы в *коммуникационный модуль*. Эти команды будут получены другими холонами, участвующими в холархии и на их основе, после взаимного анализа и проведения переговоров (за этот процесс отвечает *организационный*

модуль), может быть принято решение о реорганизации с целью улучшения производственного процесса [68–70].

Группа исследователей Университета Кила (Keele University, UK) предложила архитектуру холона, использующую агентную модель с функциональным блоком. Функциональные блоки (ФБ) определены в международном стандарте IEC 61499 [71]. Основными особенностями ФБ являются [72, 73]:

- машина состояний потоков событий (называемая диаграммой ЕСС — Execution Control Chart);
- входные и выходные событийные переменные с собственным управлением;
- входные и выходные переменные для представления потоков данных, а также ориентация функциональных блоков на реализацию (ФБ — выполнимая спецификация).

Установка (sampling) значений входных и выходных переменных производится с помощью событийных входов и выходов, соответственно. Для этого используются специальные WITN-связи. С состояниями ЕСС могут быть связаны выходные события и алгоритмы, определённые на одном из языков программируемых контроллеров стандарта IEC 61131-3 [74]. В соответствии со стандартом IEC 61499 управляющее приложение представляется в виде сети связанных между собой ФБ, которые могут выполняться на различных ресурсах и устройствах системы.

В стандарте IEC 61499 приводятся неформальные или полуформальные определения ФБ, затрагивающее только концептуальные аспекты. Этого вполне достаточно для пользователей и потребителей систем на основе ФБ, включая инженеров по управлению, системных интеграторов, менеджеров и т. д. Однако для построения систем автоматизированного проектирования распределённых компонентно-базированных систем управления промышлен-

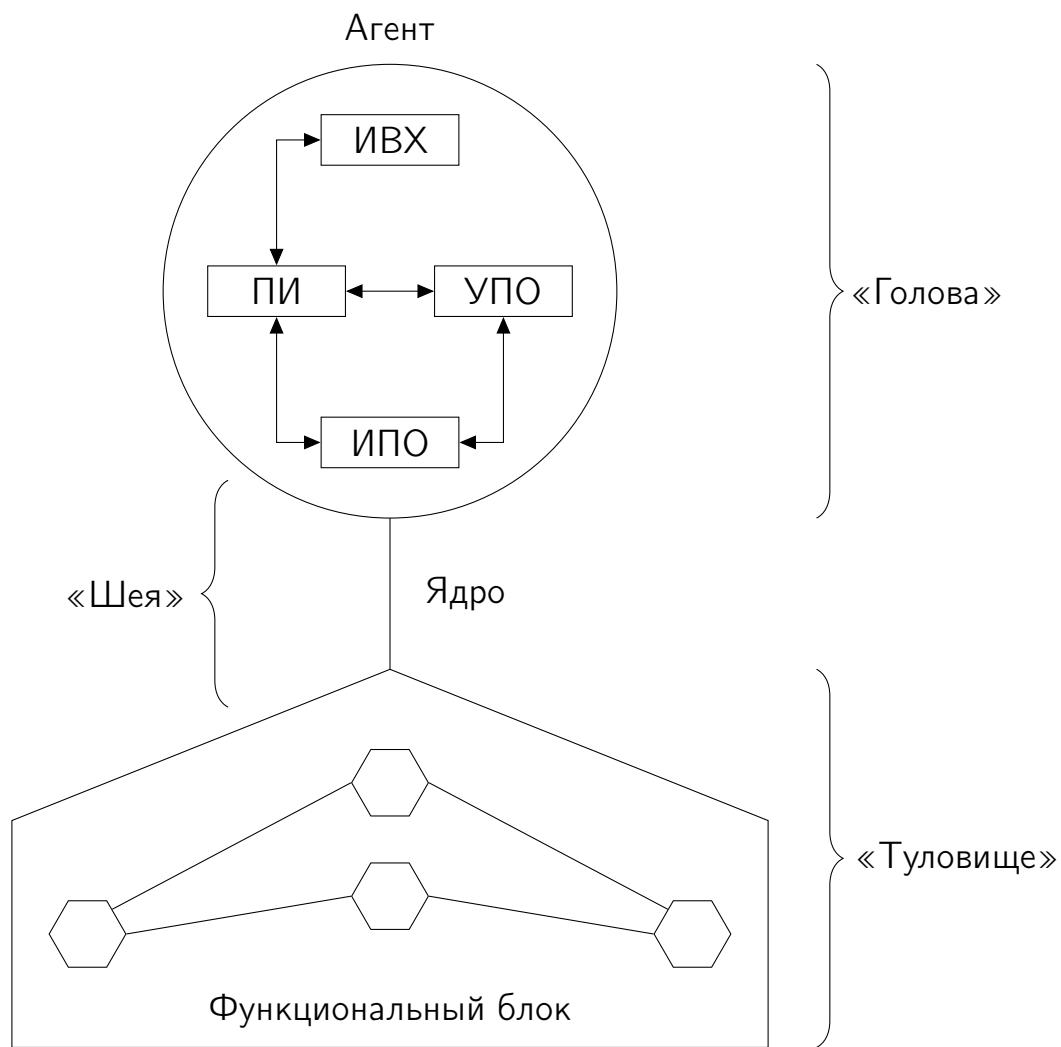


Рис. 2.11. Архитектура холона с функциональным блоком

ными процессами (РКБСУПП) этого явно недостаточно и требуются более формальные определения и модели.

Производственные и технологические холоны обычно состоят из модулей, отвечающих за обработку данных и знаний, а также необязательных модулей взаимодействия с аппаратными средствами. С точки зрения своего функционирования холон может состоять из интеллектуальной системы управления («Головы») и системы обработки данных («Туловища»), рис. 2.11. «Голова» холона строится в соответствии с агентной архитектурой, описан-

ной в работе [67]. Интеллектуальная система управления (ИСУ) состоит из:

- УПО (Управление процессами/объектами) — осуществляет управление уже запущенными процессами.
- ИПО (Интерфейс процессов/объектов) — осуществляет логическое и физическое взаимодействие с системой обработки данных через коммуникационную среду.
- ПИ (Пользовательский интерфейс) — осуществляет взаимодействие с оператором.
- ИВХ (Интерфейс взаимодействия с холонами) — отвечает за коммуникацию с другими холонами.

Система обработки данных включает в себя все модули, необходимые для осуществления производственной деятельности. Таким образом, ИСУ позволяет холону абстрагировать объекты производства в виде автономных подсистем, взаимодействующих со средой и другими холонами. ИСУ составляет определённые операционные правила и стратегии, и на их основе делегирует часть технологических функций системе обработки данных.

Холархии и другие виды взаимодействия холонов на основе агентной модели организуются с помощью структуры, именуемой *Доменом кооперации* (рис. 2.12). Домен кооперации (составной холон) является логическим пространством, в котором холоны общаются и сотрудничают, также существует некоторая среда, где холоны могут искать друг друга с целью взаимодействия.

Домен кооперации не может существовать сам по себе, он должен быть членом хотя бы одного холона. Как только холон начинает функционировать, динамически создаётся домен кооперации. Любая холоническая система состоит как минимум из одного домена кооперации. Холон может одновременно быть членом нескольких доменов кооперации. Каждый домен возглавляется

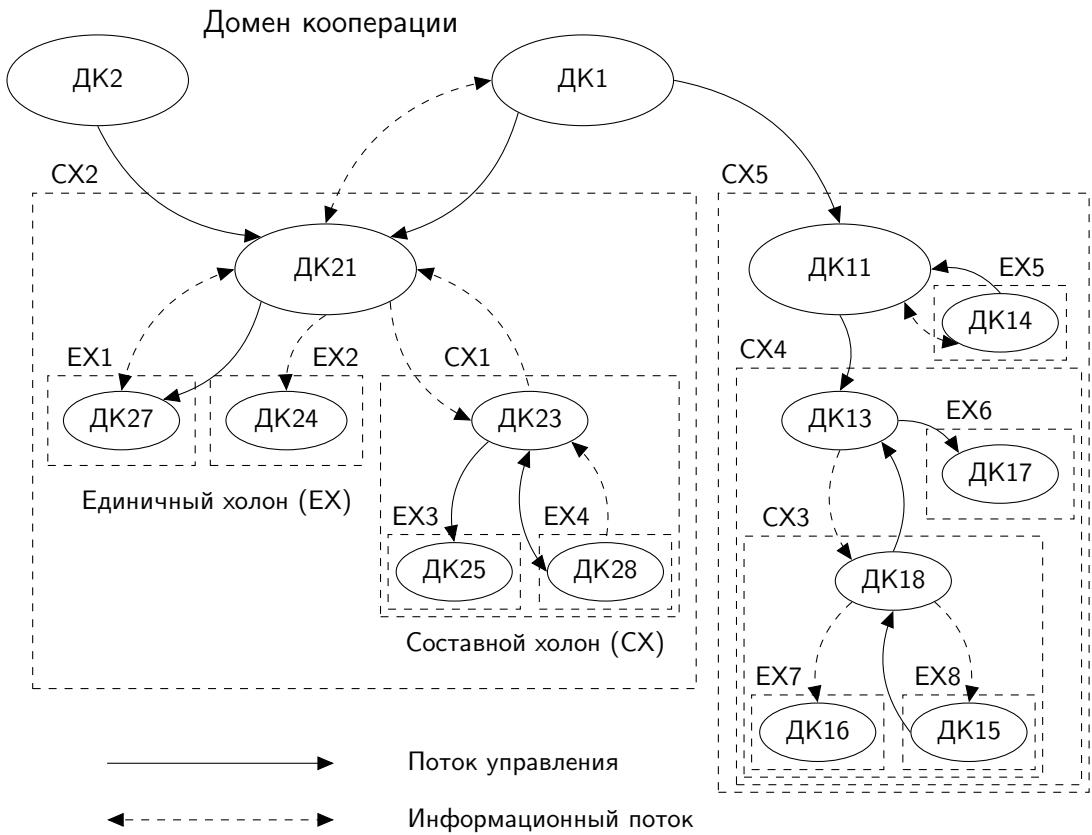


Рис. 2.12. Холархия на основе домена кооперации

Координатором, являющимся интерфейсом домена с внешней средой (другими доменами). Холон может присоединиться к домену кооперации, запросить его атрибуты, обменяться информацией с другими холонами и выйти из него по завершению своей работы.

Сайд М. Дин (Sayyed M. Deen) и Мартин Флетчер (Martyn Fletcher) предлагают модель реорганизации холархии. Модель основана на концепции *температурного равновесия* [75]. В случае возникновения задержек на одном из этапов технологического процесса, холоны испытывают перегрузку, которая делает их «горячими». Таким образом, если холон регистрирует, что его температура превышает заданное пороговое значение, он должен сообщить другим холонам о сложившейся ситуации. Если существует «холодный» холон, который может решить задачи «проблемного» этапа, он начинает

переговоры с «горячим» холоном о передаче полномочий. Самоорганизация системы в целом достигается, когда холархии, входящие в систему стараются поддерживать температурное равновесие.

В работе [76] Роберт В. Бреннан (Robert W. Brennan) и Дуглас Х. Норри (Douglas H. Norrie) предлагают агент-холоническую архитектуру, использующую агентов в совещательном слое и функциональные блоки для управления физическими объектами. В работе [77] рассматривается холоническая архитектура для управления оборудованием (ХАУБ).

В данной архитектуре совещательный слой отвечает за выполнение: общих функций системы и специфических пользовательских функций. Общие функции — это *планирование, моделирование, управление и диагностика* технологического процесса. *Совещательный слой* взаимодействует с другими слоями с помощью *таблицы данных* устройств через *Общий интерфейс доступа к таблицам данных* (ОИДТД). Совещательный слой и слой управления могут записывать и считывать данные в/из ОИДТД. *Слой управления* представляет собой пользовательское приложение, управляющее аппаратными средствами (AC), включенными в систему. На этом уровне используются функциональные блоки. *Физический уровень* представляет аппаратное обеспечение (сенсоры и исполнительные устройства), управляемые ХАУБ. *Уровень симуляции* занимается симулированием аппаратных средств.

В Немецком исследовательском центре искусственного интеллекта (Deutsches Forschungszentrum für Künstliche Intelligenz — DFKI) была разработана агент-ориентированная архитектура холонических систем [42, 78]. Исследователи отмечают, что несмотря на схожесть понятий *холонических производственных систем* и *холонических многоагентных систем*, они существенно отличаются:

- Моделирование рекурсии агентных групп является неотъемлемой частью холонических многоагентных систем. Дублирование (зеркалирование) сложных задач позволяет холоническим агентам являться частью комплексных иерархических структур с произвольной глубиной вложенности. Поэтому такие системы не только возможны, но и являются основой любой холонической многоагентной системы, чего нет в классических ХПС.
- ХПС не делают никаких предположений о внутренней структуре первичного холона ([разд. 2.3.3](#)), т. к. он выступает лишь в роли блока управления. В то время как в холонических многоагентных системах первичных холон обязан обладать всеми свойствами агента.
- Первичный холон холонической многоагентной системы не обязан координировать работу физических ресурсов (аппаратных средств), вместо этого он координирует работу нескольких информационных агентов, существующих лишь виртуально (информационные агенты сотрудничают для увеличения эффективности, объединения ресурсов и компетенций и устранения «узких мест»).
- При проектировании структуры взаимодействия холонов в ХПС используются только рыночные механизмы. А проектирование холонических многоагентных систем, в первую очередь, связано с выбором *долгосрочных* партнёров (т. е. образованием постоянной коалиции), и исследованиями многообразия различных организационных структур.
- В холонических многоагентных системах холон — это не программная процедура или экземпляр некоторого класса. Это всего лишь концепция, возникающая в результате взаимодействия агентов, определяющая взаимные обязательства агентов (как раз существующих в виде исходного

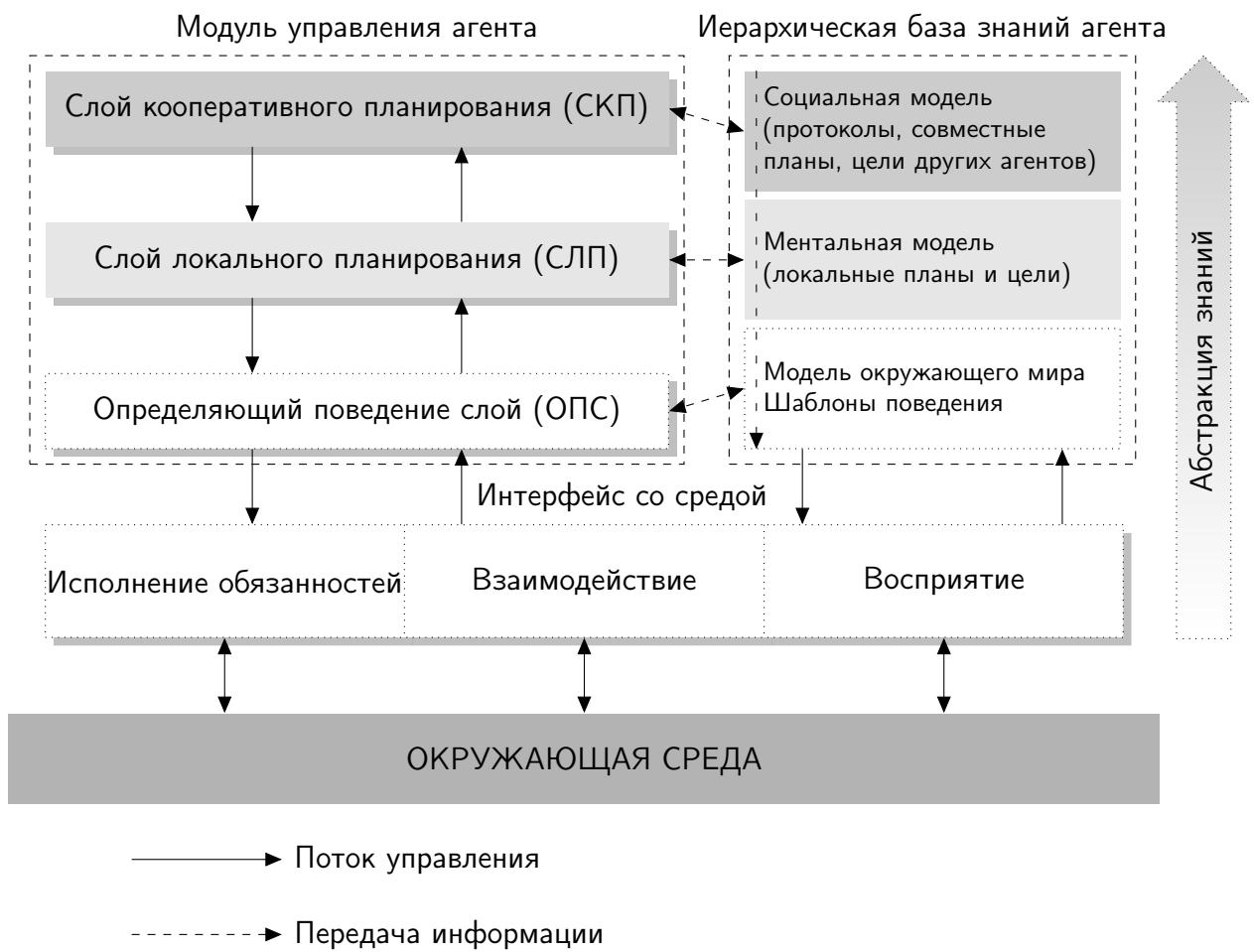


Рис. 2.13. Агентная архитектура INTERRAP

кода) и позволяющая сохранить общее направление работы агентов для достижения их общих целей.

Йоргом П. Мюллером (Jörg P. Müller) описана холоническая модель INTERRAP, основанная на взаимодействии трех конкурирующих агентных слоёв [79]. На рис. 2.13 изображена архитектура INTERRAP, в которой состав и конфигурация холонических систем реализованы в *слое кооперативного планирования* (СКП). СКП обеспечивает взаимодействие, возможность ведения переговоров и администрирование холонической структуры. Эта архитектура может быть применена в таких областях как: сети поставок, ХПС, виртуальные предприятия и т. д.

Взаимодействие холонов

Способ взаимодействия холонов определяется моделью организации, используемой для конструирования и реализации кооперативных холархий. Сотрудниками исследовательской группы технологии приборостроения и механики Университета Калгари (University of Calgary) были разработаны несколько моделей управления интеллектуальными производственными системами. Среди них: MetaMorph I, ABCDE⁷, DIDE⁸, FBIICDE⁹ и MetaMorph II. Все эти проекты основаны на промышленной архитектуре распределённого интеллектуального управления [76, 77].

Основной особенностью систем, основанных на MetaMorph [80] является их возможность изменять свою структуру. Эти системы могут адаптироваться для решения новых задач и подстраиваться под новые условия. Архитектура MetaMorph основана на концепции *домена кооперации*, разработанной Дином и Флетчером (разд. 2.3.3), но в данном случае её называют *динамическим виртуальным кластером*. В MetaMorph, как и в системе PROSA (разд. 2.3.3) определено понятие *первичного холона* и есть деление холонов на группы. Существуют холоны продуктов (изделий), холоны моделей продуктов (изделий) и холоны ресурсов. Структура холонов продуктов двояка: с одной стороны они представляют реальные (физические) изделия, которые сами по себе уже являются продуктами, а с другой — хранят информацию о других компонентах продукта и статусе технологического процесса. Холоны моделей продуктов накапливают информацию об этапах жизненного цикла изделия, данных конфигурации, конструкторских спецификациях, используемых ма-

⁷ англ. *agent based concurrent design environment* — основанная на агентной модели конкурентная среда проектирования.

⁸ англ. *distributed design environment* — распределенная система проектирования.

⁹ англ. *feature based integrated and intelligent concurrent design system* — интегрированная и интеллектуальная признак-ориентированная конкурентная система проектирования.

териалах и т. д. Холоны ресурсов используются для моделирования средств технологического оснащения и операций.

Координация и авто-организация реализуется с помощью динамических виртуальных кластеров (рис. 2.14).

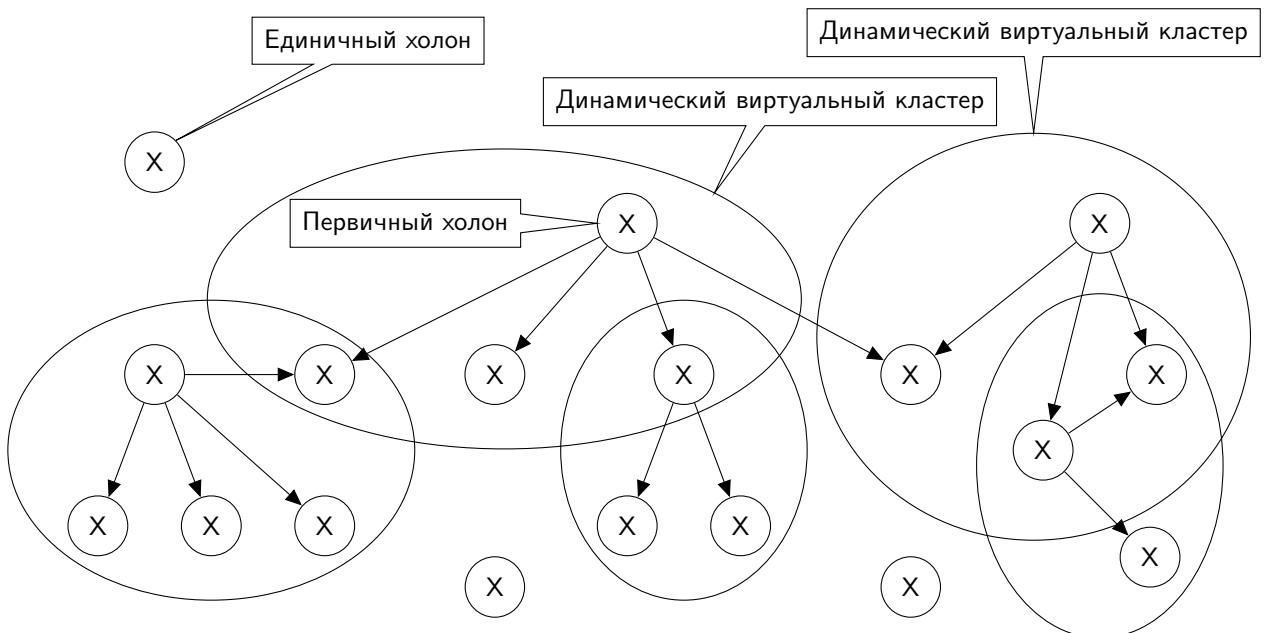


Рис. 2.14. Объединение холонов и холархия

В динамическом виртуальном кластере холоны могут одновременно находятся в нескольких виртуальных холархиях и сотрудничать через домен кооперации. Первичные холоны в данном случае выполняют ту же функцию, что и координатор холонов Дина и Флетчера, а также являются менеджерами кластера, координирующими взаимодействие холонов. Кластер существует до тех пор, пока общая задача, выполняемая холонами остаётся актуальной, как только задача решена, кластер распадается. Жизненный цикл виртуального кластера может быть следующим:

1. Первичный холон просматривает базу заказов на выполнение каких-либо работ (контрактов), выбирает несколько (или все) и формирует задачу. После перепланирования и анализа технологического процесса,

первичный холон составляет список требований для решения задач в кооперации.

2. Создаётся промежуточный холон, необходимый для поиска холонов, которые потенциально могут войти в кооперацию.
3. Найденным холонам предлагается вступить в виртуальный кластер. Они оценивают возможность участия в кооперации и высылают предложения по всем задачам, которые их заинтересовали.
4. Первичный и промежуточный холоны собирают и оценивают все полученные предложения. Когда найден оптимальный исполнитель для конкретной задачи, первичный холон передаёт контракт непосредственно холону-субподрядчику.
5. Как только все задачи по контрактам выполнены, виртуальный кластер, промежуточный холон и все кооперативные связи исчезают.

Функционирование холонов

В любой производственной системе существуют ключевые операции или функций, реализация которых непосредственно сказывается на качестве выполнения всего технологического процесса. Выявление и моделирование подобных функций является ещё одним направлением исследований и разработки в области ХПС.

PROSA (Product-Resource-Order-Staff Architecture¹⁰) является одной из наиболее часто используемых архитектур для проектирования ХПС [81, 82]. По большому счёту, PROSA — это архитектура взаимодействия холонов, декларирующая типы холонов необходимые любой производственной системе, их обязанности и среду, в которой они могут взаимодействовать. Архитектура состоит из трёх основных холонов ([рис. 2.15](#)):

¹⁰ англ. архитектура Продукт-Ресурс-Заказ-Персонал.

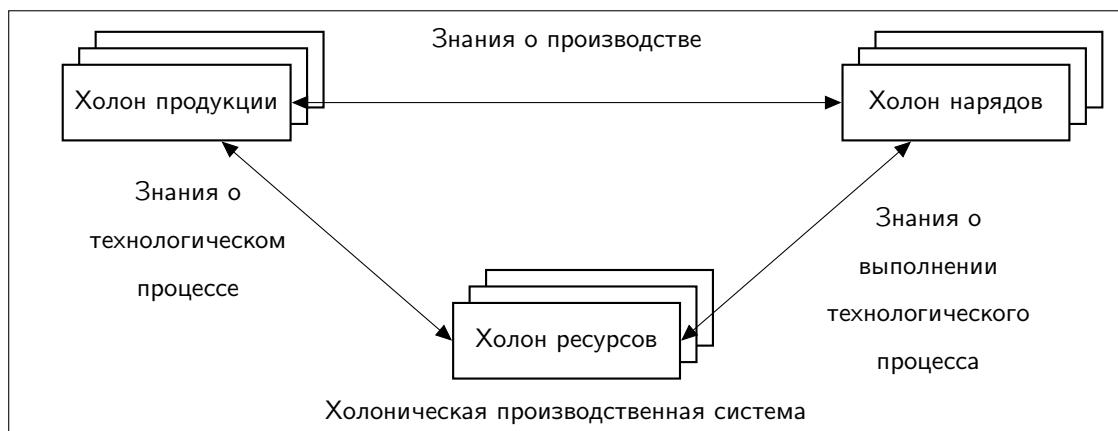


Рис. 2.15. Архитектура PROSA

- Холоны нарядов.
- Холоны продукции.
- Холоны ресурсов.

Каждый базовый холон отвечает за один из аспектов технологического процесса, например: внутрипроизводственная логистика, планирование производства, управление ресурсами и т. д. Для того, чтобы расширить функциональность базовых холонов дополнительными экспертными знаниями, создаются т. н. холоны «персонала».

Структура производственной системы в целом определяется двойной холархией, состоящей из субхолархии выделения ресурсов (*холоны нарядов*, *холоны продукции* и *холоны персонала*) и субхолархии управления технологическим процессом (*холоны продуктов* и часть *холонов ресурсов*, отвечающая за управление технологическим процессом). *Холоны ресурсов* имеют физическую часть (реальные ресурсы предприятия) и часть, отвечающую за обработку данных и управление вышеуказанными ресурсами. *Холоны продукции* хранят знания о технологическом процессе и производимом продукте, необходимые для его правильного изготовления. Они выступают в роли информационного сервера для других холонов ХПС. *Холоны нарядов* пред-

ставляют конкретные задачи, решаемые в технологическом процессе. Они отвечают за своевременное и правильное выполнение нарядов на различные работы, а также управляют материальным и информационным потоками, связанными с конкретным продуктом (изделием). Управление и координация холархий, построенных по архитектуре PROSA осуществляется с помощью модели социального поведения роя (стигмергии), описанного в работах [83, 84].

В архитектуре ADACOR (adaptive holonic control architecture¹¹) [85] исследователями предлагается холонический подход к динамической адаптации в условиях повреждения структуры *Гибких производственных систем* (ГПС). В основе этой архитектуры лежит группа автономных, взаимодействующих и интеллектуальных сущностей (холонов), представляющих все компоненты производства. Этими компонентами могут быть как физические ресурсы (стойки ЧПУ, промышленные роботы, программируемые контроллеры и др.), так и логические объекты (продукция, заказы и др.). Холоны, участвующие в группах могут быть следующих типов [86]:

- Холоны продукции.
- Холоны задач.
- Эксплуатационные холоны.
- Холоны-наблюдатели.

Каждый продукт представляется одним *холоном продукции*, хранящим всю связанную с продуктом информацию и отвечающим за ход технологического процесса. Также он может принимать заказы на изготовление этого продукта. Каждый производственный заказ связан с соответствующим *холоном задач*, который следит за исполнением плана производства и хранит данные о декомпозиции заказа, плане распределения ресурсов и выполнении

¹¹ англ. *адаптивная архитектура холонического управления*.

этого плана. *Эксплуатационные холоны* представляют физические ресурсы предприятия, например: рабочих, станки, роботов и т. д. Они управляют поведением этих ресурсов с учетом их целей, ограничений и возможностей и пытаются оптимизировать их деятельность. Холоны продукции, задач и эксплуатационные холоны очень похожи на холоны продукции, заказов и нарядов в архитектуре PROSA [81]. Холоны-наблюдатели в системе ADACOR – это холоны персонала архитектуры PROSA. Они отвечают за глобальную оптимизацию задач и координацию работы эксплуатационных холонов.

Архитектура НСВА (holonic-component-based-architecture¹²) [87] основывается на CBD (component-based development¹³) и ХПС. В НСВА определены два основных типа холонов: продукции и ресурсов. Холоны ресурсов являются встроенными компонентами системы, которые могут выполнять такие операции как: изготовление, сборка, транспортировка и технический контроль. Холон продукции может состоять из физической части и части управления. Физическая часть обычно связана с сырьем, заготовками, деталями или СТО. Часть управления отвечает за ход технологического процесса, занимается принятием решений и хранит информацию об изготавливаемом продукте.

Алгоритмы холонического управления

В этом разделе будут представлены некоторые работы, связанные с определением и реализацией алгоритмов холонического управления. Все эти работы могут быть условно разделены на четыре категории: проектирование маршрутной технологии, планирование, управление производственным процессом и управление оборудованием.

Проектирование маршрутной технологии Основные методы холонического проектирования этапов маршрутной технологии изложены в

¹² англ. *холоническая, компонентно-ориентированная архитектура*.

¹³ англ. *компонентно-ориентированное программирование*.

работах... В основном, все эти подходы связаны с разработкой интерактивного сценария, в котором холон продукции отвечает за выбор необходимых деталей, подсборок и производственных операций. Ресурсы, связанные с каждой операцией, а также их последовательность определяются посредством взаимодействия с холонами ресурсов. В условиях крупносерийного производства преимущество холонического подхода по сравнению с традиционным заключается в более высокой гибкости технологического процесса и возможности быстрой перенастройки уже запущенного процесса.

Планирование На данном направлении значительные усилия исследователей были приложены к разработке холонических алгоритмов управления. Можно выделить следующие предметные области:

- Гибкие производственные системы [26].
- Управление мелкосерийным производством.
- Проектирование участков механобработки и сборки.
- Сборочные линии.
- Техническое обслуживание оборудования.

В работах [88–91] предложены универсальные методы планирования холонических производств. Основной причиной большого количества исследований именно в области производственного планирования связано в первую очередь с высокой проработанностью методов интеллектуального планирования и распределённого управления, а также схожестью используемого в этой области подхода с предлагаемой холонической концепцией. Использование любого из них позволяет более динамично распределять время и ресурсы, что может быть достигнуто только за счёт использования методов автономного планирования.

Главной особенностью холонического подхода к планированию является тот факт, что каждый холон представляет собой автономную интеллекту-

альную сущность, способную решать поставленные перед ней задачи. Для обмена информацией и принятия взаимоприемлемых решений холоны могут использовать стратегии сотрудничества. Также существуют механизмы позволяющие не выходить за рамки глобальных ограничений. И, наконец, есть единый механизм координации.

Преимущества холонического подхода к управлению по сравнению с классическими обусловлены наличием распределённой среды принятия решений и вычислений, а также интерактивным характером холонов.

Управление производственным процессом Эта деятельность включает в себя постановку, управление, контроль и завершение задач, а также затрагивает действующие в настоящий момент планы и параметры производства. В отличии от обычных алгоритмов управления, холонический подход включает в себя следующие новые элементы: ход производственного процесса определяется последовательностью переговоров между холонами и их результатами; холоны ресурсов (оборудования), выполняющие все технологические операции, принимают решения и отвечают за сроки и способ достижения поставленных целей.

Управление оборудованием Под управлением подразумевается запуск, считывание параметров датчиков и обратная связь с исполнительными устройствами оборудования. Подобные задачи в контексте ХПС решаются также как и любые другие задачи управления. Большинство исследований в этом направлении связаны с разработкой эффективного интерфейса с устройствами.

Методология разработки ХПС

Любое современное производство — сложная и комплексная система, следовательно, и управляющая ей холоническая структура тоже не может

быть простой. Из этого следует, что в процессе разработки ХПС необходимо руководствоваться общепринятыми методами проектирования сложных программных систем. Однако, при разработке каждой новой системы инженеры обычно придерживаются принятых именно на их предприятии методов, и нет двух систем спроектированных по единому сценарию. Возможно, это стало результатом отсутствия каких-либо значимых работ по методологии разработки ХПС.

В работах [85, 86, 92] представлена формальная спецификация процесса управления ХПС, однако она всё ещё находится в разработке. В ней не до конца определены этапы разработки и нет детального описания таких направлений моделирования ХПС как: сотрудничество внутри холархии, автономия холона, гибкость системы в целом. В работе [42] предлагается отказаться от какой-либо конкретной архитектуры холона/холархии, что позволит существенно сократить время разработки ХПС. Тем не менее, этот подход ориентирован в первую очередь на описание холархий, а не на их разработку.

2.4. Выводы ко второй главе

1. Для создания полнофункциональной многоагентной системы интеграции средств автоматизации в рамках единой технологической среды необходимо спроектировать метамодель, включающую в себя описание архитектуры агентов и математической модели их взаимодействия внутри многоагентной системы.
2. Проведённые исследования показали, что на сегодняшний день существует всего две основные агентные архитектуры, используемые при построении адаптивных одноранговых агентных сетей: реактивная (основанная на продукционной модели поведения агентов) и дели-

беративная (основанная на целях агента и его восприятии модели окружающей среды).

3. Проведённый анализ достоинств и недостатков рассмотренных архитектур показал, что ни одна из них в чистом виде не может быть использована для построения многоагентной технологической системы — поэтому при проектировании должна быть использована гибридная двухуровневая схема построения агента, использующая разные подходы для решения разных задач интеграции АСТПП.
4. Функционирование и взаимодействие агентов в МАС, должно опираться на унифицированную математическую модель, представляющуюся совокупностью понятий многоагентной среды (описывающей поведение агентов в процессе решения поставленных им прикладных задач) и многоагентной системы, являющейся коммуникационной надстройкой многоагентной среды и обеспечивающей жизненный цикл агентов и их взаимодействие.
5. Для построения гибкой интеграционной системы автоматизации ТПП понятие агента должно быть расширено введением в него дополнительных функций, позволяющих ему участвовать не только в одноранговом взаимодействии в рамках интеграционной сети, но и в классических иерархических системах. Применение подобных агентов (также называемых холонами) позволит создать адаптивную систему, позволяющую при необходимости использовать не только методы многоагентного взаимодействия, но и классические централизованные системы управления технологическими данными и знаниями.

Глава 3

Программное и аппаратное обеспечение многоагентной системы технологического назначения

Процесс разработки интеллектуальной многоагентной системы с нуля — сложная и очень ответственная задача. Реализация полноценной МАС силами даже большого научного коллектива практически невозможна. Сложность обусловлена тем, что разрабатываемая система строится в рамках конкретной предметной области, следовательно, должна создаваться в первую очередь соответствующими специалистами, выполнение работ силами только профессиональных программистов сопряжена с трудностями непонимания ими основных целей разрабатываемой системы.

Решением данной проблемы является чёткое следование существующим стандартам и применение готовых инструментальных средств. Использование этих принципов позволит экспертам в заданной предметной области (в нашем случае технологической подготовке приборостроительного производства) сконцентрироваться на интеграции системы в существующую технологическую среду, оставив за скобками низкоуровневую архитектуру многоагентной системы.

На сегодняшний день существует достаточно большое количество специализированных языков программирования и инструментальных средств разработки многоагентных систем [93].

Тем не менее, единственным стандартом разработки в этой области остаётся стандарт FIPA.

В данном разделе будет сделан обзор этого стандарта, а также проведен анализ существующих прикладных библиотек. При этом предпочтение будет отдано свободно распространяемым инструментам вследствие их более высокой гибкости по сравнению с закрытыми решениями.

Задача данной главы — показать как из существующих на сегодняшний день многочисленных инструментальных средств создать целостную программно-аппаратную среду, способную стать единым фундаментом для построения единой информационно-управляющей платформы технологической подготовки производства.

3.1. Программное обеспечение МАС

Как было показано в гл. 2, для создания многоагентной системы необходимо как минимум:

1. Реализоватьprotoагента, т. е. агента-прототип, для которого не определена модель поведения и коммуникационные возможности.
2. Создать открытую среду, в которой могут существовать агенты.
3. Разработать протокол взаимодействия агентов, позволяющий в дальнейшем перестраивать МАС для решения конкретных задач.

По мнению автора, для того, чтобы связать воедино все эти три компонента в рамках рассматриваемой предметной области, также необходимо описать специализированный *символьный формат (язык)*¹ представления технологических данных и знаний, позволяющий создать формализованную основу для дальнейших разработок.

¹ Впервые идея передачи информации в символьном виде были предложена Григорием Самуиловичем Цейтниным [94]. По существу, эта идея лежит в основе создания рассмотренной ниже теории ВСПТД и представляет собой одну из первых предпосылок для эффективной организации МАС.

Традиционно в языках программирования сильны возражения против использования связей между модулями в виде символьных имён. За этим взглядом стоит не только стремление к эффективности, но и убеждённость в том, что имя всегда заменяет некоторый конкретный объект и имеет смысл лишь в связи с этим объектом. Этому подходу удобно следовать, пока мы остаёмся в пределах математических задач, на базе которых и выросли традиционные языки программирования, но реальные ситуации, возникающие за пределами таких задач, демонстрируют необходимость более широкого использования имён, в частности, установления связи между независимо построенными программными модулями путём использования одинаковых имён. Кроме того, представление информации в символьном виде даёт возможность проводить логический анализ связей между различными объектами, что позволяет организовать соответствующий аппарат логического вывода, ориентированный именно на *символьное представление* [95].

3.1.1. Универсальный формат представления технологических данных и знаний

На сегодняшний день существует достаточное количество универсальных форматов сериализации данных и знаний. К наиболее распространённым можно отнести:

- *XML* – *eXtensible Markup Language* (расширяемый язык разметки).
- *JSON* – *JavaScript Object Notation* (объектная нотация языка JavaScript).
- *YAML* – *YAML Ain't Markup Language* (рекурсивный акроним, означающий «YAML — не язык разметки»).

Основным достоинством всех вышеперечисленных языков является их широкое распространение, что подразумевает большое количество готовых инструментальных средств, позволяющих упростить и стандартизовать работу

проектируемой МАС. Тем не менее, у всех их есть и один существенный недостаток — все рассматриваемые форматы являются универсальными, т. е. дают возможность работать с данными в любых предметных областях. Поэтому эти форматы могут быть использованы только лишь в качестве контейнеров, способных упростить обработку, анализ, хранение и передачу структурированной текстовой информации, но не её семантическое представление. Последнее положение обосновывает необходимость поиска специализированного языка, учитывающего особенности рассматриваемой предметной области.

Проведённый анализ показал, что среди немногочисленных специализированных языков представления технологических данных и знаний наиболее целесообразно использовать язык, являющийся методологической основой теории *виртуального строкового пространства технологических данных (ВСПТД)*. Выбор обусловлен имеющимся положительным опытом применения ВСПТД для решения задач технологической подготовки производства,² простотой синтаксиса данного языка, а также возможностью единообразно представлять данные и знания в символьной форме, что является несомненным преимуществом при создании многоагентной системы технологического назначения.

Методология ВСПТД предполагает создание единой информационной среды описания технологических данных и знаний,³ основным структурным элементом которой является триплексная строка. Триплексная строка — это интеллектуальный объект с универсальным языком. С помощью триплекс-

² Методология виртуального строкового пространства технологических данных была положена в основу системы автоматизированного проектирования технологических процессов «ТЕХКОМ», используемой ранее на многих отечественных предприятиях [96, 97].

³ Отсюда следует, что более корректно было бы обозначить рассматриваемую методологию как *ВСПТДиЗ*, т. е. Виртуальное Строковое Пространство Технологических Данных и Знаний, но термин уже устоялся, поэтому возможность работы не только с данными, но и со знаниями предполагается, но явно не декларируется.



Рис. 3.1. Общая схема представления триплета

ных строк можно описывать самые разные структуры данных и знаний.⁴ Машина логического вывода оперирует триплексными строками, опираясь на имеющиеся *факты* и пытается достичь поставленных перед ней *целей* [101].

Теория ВСПТД во многом обогнала своё время, и сейчас мы видим, что большинство положений, ставших основой ВСПТД, могут быть легко перенесены в современные условия, что позволит создать на его основе гибкую и легко масштабируемую многоагентную систему.

Рассмотрим основные положения теории ВСПТД более подробно, начав с описания трёх базовых понятий: *триплет*, *факт* и *цель*.

Триплет является специализированным символьным объектом, форма которого представлена на рис. 3.1. В данном случае, под объектом понимается определённая сущность (математическая абстракция), находящаяся в виртуальном пространстве (среде) и обладающая определённым состоянием и поведением, чётко выделяющимися на фоне среды и фоне своей собственной структуры, т. е. частей, из которых состоит рассматриваемый объект. Объект существует только внутри среды, являясь при этом внутренней границей этой среды. Объект и среда взаимодополняют друг друга.

Технологические объекты как понятия впервые рассмотрены в работе [102]. Теория ВСПТД вводит символьное представление этих понятий. Нетрудно заметить, что подобная организация триплексных строк позволяет легко оперировать с любыми технологическими объектами, как существующими, так и вновь создаваемыми. Указанный подход позволяет отказаться от необходимости создания всё новых и новых интерфейсов для обмена ин-

⁴ Например, для представления параметризованной геометрической информации из CAD-систем, с последующей обработкой полученных параметров в АСТПП [98–100].

формацией, что, по мнению автора, является незаменимым свойством при проектировании открытых одноранговых систем интеграции.

Проектирование технологических процессов средствами ИУП ТПП со-пряжено с постоянной обработкой огромного количества разнородных данных, генерируемых структурными компонентами платформы. Часть данных попадает в систему в процессе диалога с пользователем (например, требуемые параметры материала или описание заготовки), часть выбирается из различных информационных баз данных, часть получаются расчёты путём и т. д. Следовательно, необходимо определить такую структуру данных и знаний, которая позволила бы осуществлять бесшовную интеграцию вычислительных модулей и систем. Как уже было отмечено выше, в качестве подобной универсальной структуры в теории ВСПТД используется триплет. Рассмотрим варианты применения данной структуры более подробно.

Триплеты, описывающие данные, которыми информационно-управляющая платформа технологической подготовки производства оперирует в текущий момент, называются фактами.

$$\Phi = \langle Prefix, Name, Value \rangle, \quad (3.1)$$

где *Prefix* — префикс, *Name* — имя параметра, *Value* — значение параметра.

В представленном формальном описании префикс задаёт контекст параметра, т. е. указывает на конкретный объект в предметной области. Например, если **SIGMA** — имя параметра, обозначающая предел прочности описываемого объекта, то выражение **МТ.SIGMA** будет обозначать предел прочности материала, в случае если **МТ** — это префикс, указывающий на объект «материал». То есть **МТ** — это множество всех технологических характеристик объекта «материал», а **МТ.SIGMA** — один из элементов этого множества. Текстовые параметры объекта заключаются в апострофы, чтобы отличать их от числовых параметров, переносы строк внутри текстовых параметров интерпретируются

как пробелы. Подобное представление фактов ВСПТД удобно для реализации продукционных правил вывода, а также сопутствующих пояснений. Помимо этого, для реализации функций нечёткой логики [103, 104] значение каждого факта может быть снабжено некоторым коэффициентом уверенности (см. ниже). Таким образом, формируется некоторое множество (пространство) фактов \mathcal{F} , которое в процессе работы ИУП ТПП размещается в ВСПТД [96]:

$$\mathcal{F} = \bigcup \Phi_i, \text{ где } \Phi_i — \text{триплет } i\text{-го факта.} \quad (3.2)$$

В процессе взаимодействия компонентов информационно-управляющей платформы технологической подготовки производства постоянно возникает необходимость получения новых фактов. Ранее неизвестные системе триплеты именуются целями. Префикс и имя триплета цели записываются точно также, как и в триплете факта, а вот «значение» должно быть задано по следующей формуле (схеме):

$$\text{Префикс.} \textbf{Имя} = \text{Заявка}; \quad (3.3)$$

Подобный триплет в теории виртуального строкового пространства технологических данных именуется «простой целью». Задача информационно-управляющей платформы технологической подготовки производства — заменить заявку данного триплета на некоторое значение:

- либо используя информационно-поисковую систему;
- либо расчёты или логическим путём на основе уже имеющихся в системе фактов;
- либо с помощью лица, принимающего решение (ЛПР).

Получение фактов может быть как жёстко регламентировано, так и быть получено с помощью специальной управляющей структуры:

$$\Psi = \langle Prefix, Name, Mode, Number \rangle, \quad (3.4)$$

где *Prefix* — префикс, *Name* — имя параметра, *Mode* — способ достижения цели, *Number* — идентификатор исполняющей функции.

Очевидно, что представленная управляющая структура не всегда обладает исчерпывающей информацией о том, какие конкретно факты необходимы для корректного исполнения правил. Эти сведения предоставляют сами исполняющие правила в виде *триплетов цели*. В качестве заявки триплету цели может быть передан специальный управляющий символ («:», двоеточие), произвольная информация (:любой-поддерживаемый-тип-данных), либо пустое значение (что указывает на необязательность заполнения данного триплета, т. е. он относится к категории триплетов с необязательной целью). Управляющий символ «:» указывает на то, что рассматриваемый триплет является триплетом с обязательной целью и его заполнение в зависимости от контекста и уровня формализации должно быть произведено любым из трёх указанных выше способов. Если в качестве значения заявки триплету передано произвольное значение, это означает, что данный триплет будет заполнен без участия ЛПР, т. е. напрямую выбрано из виртуального структурного пространства технологических данных, а в случае его отсутствия — будет использоваться текст после управляющего символа «:». Например, триплет цели \$MT.SIGMA=:1000; будет преобразован к виду триплета факта \$MT.SIGMA=1000; и станет триплетом с наложенным значением. Таким образом, в процессе проектирования появляется множество целей \mathcal{C} :

$$\mathcal{C} = \bigcup \mathcal{C}_i, \text{ где } \bigcup \mathcal{C}_i \text{ — триплет } i\text{-й цели.} \quad (3.5)$$

Кроме того, может быть два типа сложных целей:

1. Объектная (структурированная) цель.
2. Составная цель.

Объектная (структурированная) цель ожидает получения всей информации по заданному объекту, в соответствии с его префиксом. Например,

по заявке $\$MT.=;$; из виртуального строкового пространства технологических данных будет извлечена вся информация о текущем материале, т. е. множество фактов с префиксом MT . Объектная цель обязательного типа ($\$MT.=:;$) будет преобразована в множество простых обязательных целей согласно структуре заданного объекта. Составная цель может быть получена из некоторой фразы содержащей факты, цели, и объяснения. Например следующее выражение:

Листинг 3.1. Пример триплексной строки, описывающей переход

```
1 $L . WOB=019 ; "притереть" ; "поверхность"
2 $L . KE =: ;      "выдерживая требования"
3 $D . TT =: ;
```

включает в себя факт, простые цели и объяснения. После заполнения всех целевых заявок будет получен текст перехода, готовый для передачи специализированному генератору отчётов.

В процессе проектирования триплеты группируются в *триплексные строки* и помещаются в виртуальное строковое пространство технологических данных. В дальнейшем под *триплексной строкой* будем подразумевать некоторое множество фактов и целей, объединённых одним ключом. А виртуальное строковое пространство технологических данных — это множество триплексных строк [98].

Описанный механизм позволяет в рамках рассматриваемой задачи работать со всеми основными типами и структурами данных (целями и вещественными числами, строками, массивами, списками, кортежами, множествами, хэш-таблицами и др.), формировать на их основе более сложные структуры, например, графы или многосвязные списки. Также определены три базовые структуры представления знаний: *синтагмы*, *фреймы* и *продукционные правила*.

Синтагмы⁵

Синтагмы и синтагматические цепи относятся к классу логико-лингвистических методов представления знаний. Синтагма представляет собой выражение $A\rho B$, где A и B — некоторые понятия или объекты, а ρ — отношение между ними с квантификаторами, модификаторами, либо оценками. Например, если если префикс L задаёт множество элементов обработки, а _PER_ обозначает отношение перпендикулярности, выражение:

`$L13 _PER_ $L8;$L7 _PER_ $L4;`

будет означать, что поверхность 13 перпендикулярна поверхности 8 и поверхность 7 перпендикулярна поверхности 4. Также в синтагму можно включать дополнительную информацию в виде смысловой оценки [105], значение которой определяется из контекста рассматриваемого отношения, поэтому выражение `$L5 _NEPER_ $L7 "<0.02"`; обозначает неперпендикулярность поверхности 5 к поверхности 7 не более, чем на 0.02.

Фреймы

Термин *фрейм* (от англ. *frame* — «каркас» или «рамка») впервые был предложен Марвином Минским [106, 107] в конце 70-х годов для обозначения структуры знаний, представляющий собой схему действий в реальной ситуации. Фрейм представляет собой комплексный пакет знаний, хранимый в человеческом сознании, либо в памяти компьютера. Каждый фрейм содержит отделения — *слоты*, в которых собраны атрибуты (характеристики) и соответствующие им значения. Например, абстрактный образ «*фрезерный станок*», будучи произнесённым вслух порождает у слушающих некоторый

⁵ Синтáгма (др.-греч. σύνταγμα, букв. «сопорядок», от др.-греч. σύν «с» и др.-греч. τάγμα «порядок») — многозначный термин, который может быть переведён как классификация, систематизация, компоновка, расстановка.

образ: «металлорежущий станок, главным движением которого является вращение неподвижно закреплённой в патроне фрезы, а обработка осуществляется за счёт подачи стола, на котором закреплена заготовка». Из данного описания ничего нельзя убрать, но в нём есть определённые отделения — *слоты*, в которых собраны атрибуты (характеристики) и соответствующие им значения, например «главное движение» или «инструмент».

В теории фреймов подобный образ называется фреймом фрезерного станка, фреймом также называется и формализованная модель для отображения образа. Это самые простые фреймы, Минским также предложены и гораздо более сложные взаимозависимые структуры знаний, фреймы которых в своих слотах могут содержать другие фреймы и входить в состав более крупных фреймов. Программа, имеющие в своём распоряжении достаточное количество фреймов, может искать в предложении символы, совпадающие с теми, которые хранятся в имеющихся слотах, вовлекая в работу и все родственные фреймы. Таким образом, система может выводить знания, отсутствующие в явном виде в исходных данных.

Различают *фреймы-образцы* или *фреймы-прототипы*, хранящиеся в базе знаний, и *фреймы-экземпляры*, которые создаются для отображения реальных фактических ситуаций на основе поступающих данных. Рассматривая теорию фреймов как один из основополагающих методов представления знаний, в теории ВСПТД описан метод, позволяющий интегрировать технологические знания при взаимодействии различных элементов ТПП. За основу в этом механизме взято единое представление элементарных фактов и целей. То есть в качестве основной единицы представления фрейма-слота в теории ВСПТД предложен триплет. Можно выделить следующие основные группы представления технологических знаний в виде фреймов: фреймы-переходы и фреймы-операции, фреймы-формулы, фреймы-таблицы, фреймы-продукции,

фреймы-терминалы, фреймы-документы, фреймы-объяснения и фреймы-запросы.

В данном перечне дополнительного объяснения требуют только *фреймы-терминалы*. Фрейм-терминал представляет собой триплексную строку, содержащую факты и цели. На их основании строиться диалог с пользователем (форма), при этом факты описывают статические элементы формы, а цели — поля, которые пользователю необходимо заполнить.

Продукционные правила

Продукционные правила (продукции) представляют собой структуру представления знаний вида ЕСЛИ (Условие) ТО (Действие). Условия продукции включают в себя обычные функции сравнения текстовых и числовых величин, арифметические выражения и тригонометрические функции, а также предикаты ЕСТЬ и НЕТ, проверяющие соответственно наличие (отсутствие) указанного параметра в ВСПТД. Правила вывода разбиваются на группы по выполняемым функциям. Каждой группе присваивается постоянный номер, который используется управляющей структурой при определении последовательности их исполнения [98].

Каждая продукция может быть снабжена коэффициентом уверенности, т. е. априорной оценкой истинности того или иного результата (факта). Коэффициент уверенности представляет собой целое число в диапазоне от -100 до 100 (на самом деле от -1 до 1, но для удобства отображения все коэффициенты уверенности умножаются на 100). Здесь коэффициент уверенности 100 означает «совершенно точно, что...», -100 — «совершенно точно, что не...» (отрицательное значение коэффициента уверенности указывает на то, что данное правило опровергает сформулированное заключение), а 0 — «неизвестно» [101]. В случае конфликта, т. е. наличия в базе знаний двух продукции, описывающих суждения

об одном и том же понятии, результирующий коэффициент уверенности $\mathcal{K}(X, Y)$ рассчитывается по следующей формуле:

$$\mathcal{K}(X, Y) = \begin{cases} (X + Y) - (X \cdot Y), & \text{при } X > 0 \text{ и } Y > 0 \\ (X + Y) + (X \cdot Y), & \text{при } X < 0 \text{ и } Y < 0 \\ \frac{X + Y}{1 - \min(|X|, |Y|)}, & \text{при } (X > 0 \text{ и } Y < 0) \text{ или } (X < 0 \text{ и } Y > 0) \\ X, & \text{при } X \neq 0 \text{ и } Y = 0 \\ Y, & \text{при } X = 0 \text{ и } Y \neq 0 \\ 0, & \text{при } X = 0 \text{ и } Y = 0 \end{cases}$$

Все базовые положения теории ВСПТД, представленные выше, могут быть использованы для создания универсального формата описания технологических данных и знаний. Подробное описание языка представления данных и знаний многоагентной информационно-управляющей платформы технологической подготовки производства, представленное в расширенной форме Бэкуса-Наура представлено в [прил. А](#).

3.1.2. Реализацияprotoагента

Protoагент представляет собой автономный программный модуль⁶ (в общем случае экземпляр класса), обладающий уникальным идентификатором, набором слотов, в которые могут быть записаны транспортные адреса, а также контекстами поведения. Protoагент связан с определённым технологическим объектом (или пользователем) через внутренний интерфейс. Наличие контекста даёт возможность создавать распределённую сетевую структуру, в которой могут взаимодействовать подобные абстрактные агенты. Архитектура рассматриваемого *protoагента* представлена на [рис. 3.2](#).

⁶ В соответствии с ГОСТ 19781–90 [108], *программный модуль* (program module) — программа или функционально завершённый фрагмент программы, предназначенный для хранения, трансляции, объединения с другими программными модулями и загрузки в оперативную память.

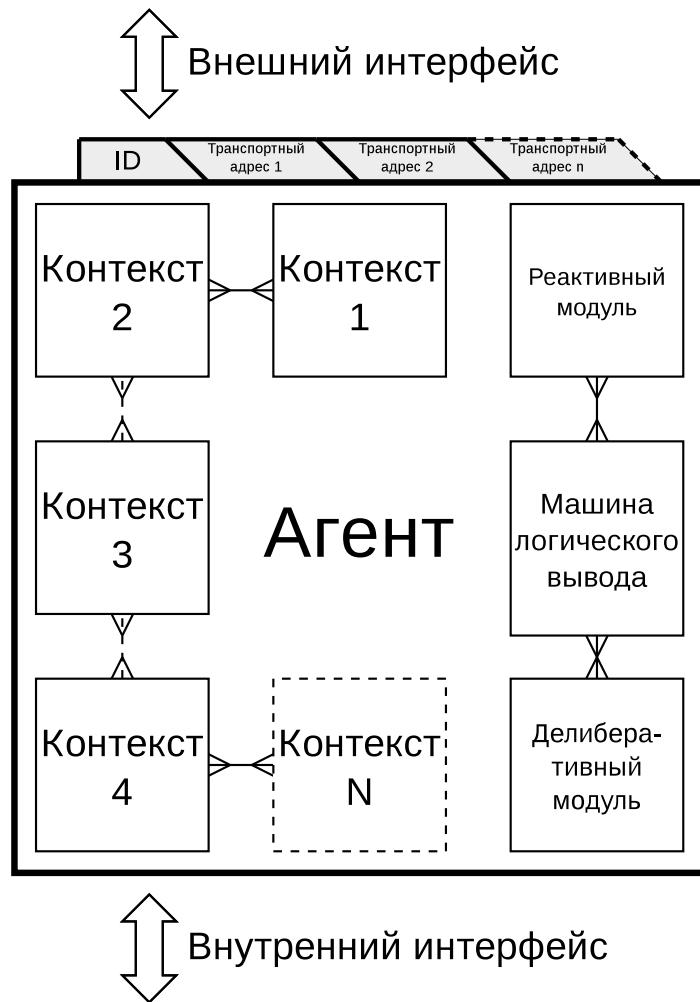


Рис. 3.2. Архитектураprotoагента

Контекст, в котором существуетprotoагент может включать набор ассоциированных с ним поведенческих характеристик. Данные характеристики могут влиять на внутреннее поведение агента. Для разных контекстов будут определены разные модели поведенияprotoагента. Контексты могут быть упорядочены в иерархическую структуру, т. е. содержать подконтексты. Таким образом, контексты при необходимости могут быть разделены на компоненты, при этом членство в контекстах наследуется.

Протоагентом данная модель называется потому, что имеет лишь потенциальную возможность к взаимодействию в агентной среде, т. к. в ней, во-первых, не определено внутреннее поведение (лишь контекст технологи-

ческого процесса, конкретная модель поведения будет присвоена агенту при инициализации), а во-вторых, не задан механизм поиска других агентов в сети.

Основной задачейprotoагента является создание стандартизованного внутреннего интерфейса, позволяющего связывать его с объектом, который он представляет: интегрируемой системой автоматизации или пользователем. Вопросы взаимодействия с пользователем будут подробно описаны в [гл. 4](#), далее же будет рассмотрена реализация абстрактных методов интеграции protoагента с автоматизированной системой (подразумевается, что независимо от типа интеграции возникает необходимость конвертирования данных и знаний, возникающих в процессе информационного обмена интегрируемых систем из/в формат ВСПТД).

Проведённое исследование показало, что существуют всего три основных способа интеграцииprotoагента с системой автоматизации технологической подготовки производства:

1. Через СОМ-интерфейс системы.
2. По протоколу XML-RPC.
3. Через специфический интерфейс программирования приложения (Application Programming Interface — API).

Ниже будет продемонстрирована реализация первых двух способов. Последний вариант встречается крайне редко, поэтому в представленном исследовании рассматриваться не будет — специфический API требует индивидуального подхода в каждом конкретном случае, следовательно практически не может быть стандартизирован.

Интеграция по протоколу XML-RPC

Протокол XML-RPC (*Extensible Markup Language Remote Procedure Call* — XML-вызов удалённых процедур) [109] является универсальным интеграционным протоколом, используемым многими автоматизированными системами для взаи-

модействия по сети. Отличительной особенностью этого протокола является его простота, обусловленная в первую очередь использованием уже готового стандарта XML. Данный механизм поддерживается многими современными системами, такими как: ProEngineer, ENOVIA SmarTeam, OpenERP, TechCard (частично) и многими другими. Как уже отмечалось выше формат XML прекрасно подходит для инкапсуляции символьных данных, к которым относятся и триплексные строки ВСПТД. Реализация базового модуля, обеспечивающего агенту XML-RPC интерфейс представлена в листинге 3.2, модуль разработан на языке Python. Приведена только основная часть алгоритма без обработчиков исключений и вспомогательных методов.

Листинг 3.2. Базовый модуль XML-RPC интерфейса

```

1 # Импорт модуля ВСПТД
2 import vsptd
3
4 # Импорт методов для работы с XML-RPC
5 from xmlrpclib import ServerProxy
6
7
8 class AgentXMLRPC(object):
9     """Реализует интеграцию агента с автоматизированной
10    системой по протоколу XML-RPC.
11    """
12
13    def __init__(self, addr):
14        """Инициализирует объект класса.
15        Аргументы:
16            addr: адрес xml-rpc сервера.
17        """
18
19        # Обращению к машине, на которой установлена
20        # интегрируемая система
21        self.server = ServerProxy(addr)
22
23    def call_func(self, func, trp_str):
24        """Вызывает функцию интегрируемой системы.
25        Аргументы:
26            func: имя вызываемой функции.
27            trp_str: триплексная строка.
28        Возвращаемое значение:
29            Объект, полученный в результате обработки.
30        """
31
32        # Получить параметры функции из
33        # триплексной строки
34        params = vsptd.get_params(trp_str)
35
36        return getattr(self.server, func)(*params)

```

Например, агенту необходимо получить перечень параметров материала Lustran QE 1088 из связанной с ним PDM системы, в интеграционном

интерфейсе которой определена функция `GetMaterialParams`. Пример использования данного модуля представлен в листинге 3.3.

Листинг 3.3. Пример использования модуля XML-RPC интерфейса

```
1 interface = AgentXMLRPC('localhost:8000')
2 print intreface.call_func(
3     'GetMaterialParams',
4     '$MT.NMAE="Lustran QE 1088"',
5 )
```

В данном случае модулем будет сгенерирован XML-документ вида:

Листинг 3.4. Запрос серверу по протоколу XML-RPC

```
1 <?xml version="1.0"?>
2 <methodCall>
3   <methodName>GetMaterialParams</methodName>
4   <params>
5     <param>
6       <value><string>Lustran QE 1088</string></value>
7     </param>
8   </params>
9 </methodCall>
```

Сервером будет возвращён следующий ответ:⁷

Листинг 3.5. Ответ сервера по протоколу XML-RPC

```
1 <?xml version="1.0"?>
2 <methodResponse><params><param><struct>
3   <member>
4     <name>Плотность</name>
5     <value><double>1.06</double></value>
6   </member>
7   <member>
8     <name>Трекинг индекс</name>
9     <value><i4>600</i4></value>
10    </member>
11 </struct></param></params></methodResponse>
```

⁷ Необходимо обратить внимание на тэг `struct`, описывающий массив элементов с ключами. Нетрудно заметить, что данная структурная единица протокола XML-RPC представляет триплексную строку, содержащую набор фактов.

Интеграция через СОМ-интерфейс

Технология СОМ (Component Object Model⁸) [110, 111], разработанная компанией Microsoft, представляет собой платформонезависимую распределённую объектно-ориентированную систему для создания приложений, взаимодействующих друг с другом.

СОМ является даже более распространённой технологией, чем XML-RPC. По мнению автора, единственным её недостатком является чрезмерная сложность, особенно при реализации работы по сети. Поэтому разработчики, зачастую, реализуют в своих системах оба этих интерфейса, что добавляет гибкости используемому в работе подходу к интеграции. Основными достоинствами данной технологии являются:

- Полная независимость от используемого языка программирования. Так же как и XML-RPC, данная технология является лишь способом обращения к внутреннему интерфейсу интегрируемого приложения, единственное отличие — ориентация на объектную, а не процедурную модель.
- Возможность расширять приложение без необходимости пересборки.
- Возможность взаимодействовать с другими объектами вне зависимости от рабочей среды.
- Стандартизация объектно-ориентированного программирования, включающая: типы объектов, стандартные методы, соглашения обозначений, инкапсуляцию.

Данными, относящимися к СОМ объекту, можно манипулировать через его интерфейсы. Автором был разработан модуль, позволяющий управлять интерфейсами автоматизированных систем, имеющими СОМ API. Реализация модуля представлена в листинге 3.6.

⁸ англ. *объектная модель компонентов*.

Листинг 3.6. Базовый модуль СОМ интерфейса

```
1 # Импорт методов COM интерфейса
2 from win32com.server.register import UseCommandLine
3 from win32com.client import Dispatch
4
5 class AgentCOM():
6     """Реализует интеграцию агента с системой
7     автоматизации по протоколу COM.
8     """
9
10    # Параметры публичного интерфейса
11    _public_methods_ = ['CallFunction']
12    _reg_progid_ = 'AgentCOM'
13    _reg_clsid_ =
14        '{79A29526-F93B-4E38-96C2-7E5DB85BC4CB}',
15
16    def CallFunction(self, fname, *args):
17        """Вызывает функцию интегрируемой системы.
18
19        Аргументы:
20            fname: имя вызываемой функции.
21            args: список аргументов.
22
23        Возвращаемое значение:
24            Объект, полученный в результате обработки.
25        """
26        # импорт функции расширения
27        module = __import__(fname, globals(), locals(),
28                            [fname], -1)
29        # Последовательная обработка аргументов
30        # диспетчером
31        args = [Dispatch(arg) for arg in args]
32
33        return getattr(module, fname)(*args)
34
35 if __name__ == '__main__':
36     # Регистрация COM сервера
37     UseCommandLine(AgentCOM)
```

Вызов рассмотренной выше функции `GetMaterialParams` может быть реализован с помощью следующего модуля:

Листинг 3.7. Пример использования модуля СОМ интерфейса

```

1 def GetMaterialParams(trp_str):
2
3     params = vsptd.get_params(trp_str)
4     return CallFunction('GetMaterialParams',
5                         params)
6
7 print GetMaterialParams('$MT.NMAE="Lustran QE 1088"')

```

В данном случае функция будет загружена в СОМ-сервер и запущена в качестве локального объекта.

3.1.3. Онтологический словарь многоагентной системы

В теории виртуального строкового пространства технологических данных определено понятие *словарь метаданных (СМД)*. С помощью СМД осуществляется контроль за данными, некоторыми элементами синтаксиса и семантики базы знаний виртуального строкового пространства технологических данных, обеспечивается лингвистическое обеспечение системы при ведении диалога и формировании отчётов. Принципиально важной функцией словаря является обеспечение интерфейса с пользователем на понятийном уровне. Все элементы предметной области имеют общий механизм интерфейса понятий. Это позволяет избавить пользователя от необходимости работы с кодируемой информацией. В каждом описываемом в словаре метаданных реквизите содержится, как минимум, следующая информация: мнемоническое обозначение, полное наименование, формат и способ получения реквизита. Очевидно, что роль словаря метаданных в многоагентных системах резко возрастает.

На основании проведённого исследования автором предлагается расширить понятие словаря метаданных, добавив в него *семантические онтологии*.

Онтологии позволяют агентам не просто оперировать данными и знаниями в виде триплексных строк, но осуществлять преобразование понятий, которые при обычном подходе могли вызвать трудности, связанные с дублированием. Помимо преобразования символьных понятий онтология является универсальным инструментом для преобразования знаний за счёт введения в её структуру присоединённых процедур.

Согласно [107], онтология — эксплицитная⁹ спецификация концептуализации предметной области (как правило таксономическая¹⁰), представляющая собой словарь метаданных и знания о предметной области во внешнем представлении, которое может транслироваться во внутреннее представление агентов. Пример онтологических разделов словаря метаданных технологических терминов приведён в прил. Б.

Классическая онтология состоит из следующих частей:

- Множество концептов — понятий.
- Отношения между концептами.
- Атрибуты и свойства концептов.
- Ограничения на свойства и атрибуты.
- Экземпляры и присоединённые процедуры, заданные в форме линейных списков.
- Знания из предметной области.

То есть, с одной стороны онтологический словарь может быть плоской структурой, описывающей отдельные концепты, а с другой — являться классификатором этих концептов за счёт наличия отношения наследования.

⁹ От англ. *explicit* — «явный», «открыто выраженный».

¹⁰ Таксономия (от др.-греч. *τάξις* — строй, порядок и *νόμος* — закон) — учение о принципах и практике классификации и систематизации.

Онтология виртуального строкового пространства технологических данных может быть задана следующим кортежем:

$$\mathcal{O} = \langle \mathcal{T}, \mathcal{R}, \mathcal{I} \rangle, \quad (3.6)$$

где \mathcal{T} — множество триплетов (концептов), \mathcal{R} — множество отношений между ними, \mathcal{I} — множество интерпретаций.

Использование онтологий позволяет отделить семантику используемых в процессе проектирования ТПП понятий от непосредственно данных, которые они представляют. Подобное разделение необходимо при проектировании языка взаимодействия технологических агентов, ведь, как уже отмечалось ранее, агенты внутри ИУП ТПП должны стать универсальными преобразователями данных и знаний, т. е. для их успешного функционирования необходимо избавиться от двусмысленности понятий.

Особенно важную роль словарь играет при взаимодействии с действующими (внешними) БД, имеющимися на предприятии. Например, в одной из баз данных режущего инструмента поле диаметр обозначается как DIAM, а в другой — DIAMETR, в то время как в ВСПТД параметр обозначен как Е.Д. Для таких случаев в словаре метаданных вводится графа, обеспечивающая соответствие имён, содержание которой в данном примере может быть таким:

Листинг 3.8. Пример преобразования понятий в СМД

1 \$I1.DIAM; \$I2.DIAMETR;

Где I_1 и I_2 указывают на базы данных, используемые системой в текущий момент.

Для унификации представления знаний в онтологиях предлагается инкапсулировать триплексные строки в контейнер, применяя вышеописанный формат JSON по аналогии с языком запросов MQL, используемый для доступа к базе знаний *Freebase* [112]. Однако, предлагаемый автором подход

является более гибким вследствие того, что ВСПТД — это универсальный формат для представления данных и знаний, JSON для него всего лишь удобная обёртка, а MQL — просто язык запросов, использующий синтаксис формата JSON.

Например, сообщение, описываемое подклассом понятий, относящихся к суперклассу *материал*, может быть задано следующим образом:

Листинг 3.9. Использование обертки JSON для представления метаданных онтологии

```

1 { "-name"      : "Параметры полимерных
2                           композиционных материалов",
3   "-prefix"    : "PCM", // Определяет иерархическую связь
4   "-row"       : [
5     { "denom"     : "Объёмное сопротивление",
6       "format"    : "9999", // Формат записи
7       "name"      : "$PCM.VOLRES", // Имя концепта
8       "synonyms"  : ["Volume Resistivity",
9                     "R_объём."],
10      "glinks"    : ["$PCM.RES"], // Общие связи
11      "procs"     : ["pcm.get_volres"], // Присоединённые
12                           // процедуры
13      "params"    : [{"стандарт": "ASTM D 257",
14                      "ед.изм.": ["Ом*см", "Omh*cm"]},
15                      {"стандарт": "IEC 60093",
16                      "ед.изм.": ["Ом*м", "Omh*m"]}],
17                  ] // Дополнительные параметры
18    }
19  ]
20 }
```

Гибкая структура ВСПТД триплетов позволяет динамически модифицировать онтологический словарь многоагентной системы вводя в его структуру различные свойства, знания и отношения в соответствие с рассматриваемой предметной областью.

3.1.4. Протокол взаимодействия агентов

Протокол взаимодействия агентов внутри МАС стандартизован *Фондом интеллектуальных физических агентов* (Foundation for Intelligent Physical Agents — FIPA) [113]. Данная некоммерческая организация является 11 комитетом *Института инженеров по электротехнике и электронике* (Institute of Electrical and Electronics Engineers — IEEE) и занимается разработкой способов проектирования многоагентных приложений.¹¹

Согласно FIPA агент «обладает способностью предоставлять в рамках унифицированной и интегрированной исполнительной модели один или более сервисов, которые могут включать доступ к внешнему программному обеспечению, пользователям и коммуникационным возможностям» [113].

Работа FIPA, основываются на следующих принципах:

1. Агентные технологии предоставляют новую парадигму решения старых и новых проблем.
2. Некоторые агентные технологии уже достаточно развиты.
3. Для использования некоторых агентных технологий требуется стандартизация.
4. Стандартизация внутренней структуры агентов не является приоритетной задачей, важнее разработать инфраструктуру и язык, дающие возможность агентам беспрепятственно взаимодействовать.

Язык взаимодействия агентов FIPA (FIPA Agent Communication Language — FIPA ACL) основывается на теории речевых актов [114, 115], декларирующей, что любое коммуникативное сообщение равноценно поступку (действию), речевой акт также известен как *перформатив*.

FIPA-ACL оперирует 22 перформативами,¹² из которых два — *Inform* и *Request* — являются базовыми, а остальные представляют собой *макроперформативы*.

¹¹ Перечень ключевых спецификаций FIPA представлен в табл. В.1 прил. В.

¹² Полный перечень речевых актов FIPA представлен в табл. В.2 прил. В.

определения, заданные в терминах этих перформативов. Результат базовых речевых актов определяют *предусловия*, т. е. условия, которые должны быть истинными, чтобы перформатив достиг цели, и *рациональный эффект* — та самая цель, достичь которую надеется отправитель перформатива. Каждый перформатив представляет собой некоторое сообщение, общая структура сообщения представлена на [рис. 3.10](#).

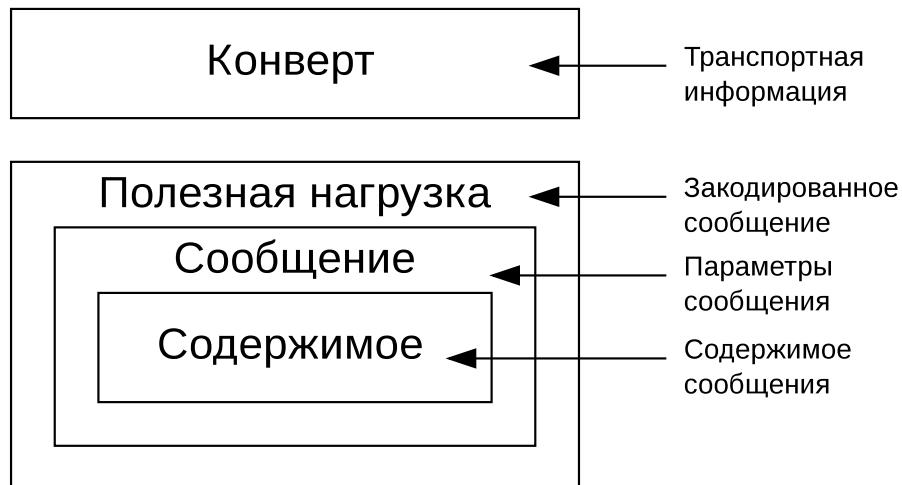


Рис. 3.3. Структура сообщения FIPA

Речевой акт Inform

Содержимое перформатива представляет собой *утверждение* (statement).

Предусловие заключается в том, что отправитель:

- убеждён, что содержимое сообщения истинно;
- намеревается сделать всё, чтобы получатель поверил, что содержимое сообщения истинно;
- пока сомневается, что получатель осведомлён о том, является ли содержимое истинным или нет.

Формальная семантика коммуникационного акта **Inform** может быть задана следующим образом:

$$\begin{aligned} & \langle s, \text{inform}(r, p) \rangle \\ FP: & \mathcal{B}_s p \wedge \neg \mathcal{B}_s (\mathcal{B}_{\text{if}}_r p \vee \mathcal{U}_{\text{if}}_r p) \\ RE: & \mathcal{B}_r p, \end{aligned} \quad (3.7)$$

где s — отправитель сообщения, r — получатель сообщения, p — содержимое сообщения (утверждение), \mathcal{B} — модальный оператор «убеждён», \mathcal{U} — модальный оператор «сомневается», $\mathcal{B}_{\text{if}}_r p$ — сокращённая запись от $\mathcal{B}_r p \vee \mathcal{B}_r \neg p$, $\mathcal{U}_{\text{if}}_r p$ — сокращённая запись от $\mathcal{U}_r p \vee \mathcal{U}_r \neg p$, FP — предусловия выполнимости, RE — предполагаемый эффект сообщения.

Речевой акт **Request**

Содержимое перформатива представляет собой *действие* (action). Предусловие заключается в том, что отправитель:

- намеревается сделать всё, чтобы было выполнено действие, указанное в содержимом сообщения;
- убеждён, что получатель способен выполнить это действие;
- сомневается, что к настоящему моменту у получателя уже есть намерение выполнить это действие.

Формальная семантика коммуникационного акта **Request** аналогична семантике **Inform**. В соответствии со стандартом любой агент должен иметь возможность принимать и обрабатывать любые сообщения. Если агент по каким-либо причинам не может обработать сообщение, он должен ответить на него перформативом *Не понимаю* (Not understood). Также FIPA определён набор коммуникационных протоколов, каждый из которых описывает какую-то последовательность речевых актов для создания различных

коммуникационных сетей и два специализированных сервиса: *Служба управления агентами (СУА)*, организующая жизненный цикл агентов и хранящая их адреса и *Служба каталога (СК)*, регистрирующая сервисы (абстрактные ресурсы), которыми располагает каждый агент.

Реализация протокола взаимодействия

Протокол взаимодействия агентов реализован с помощью языка Python, с применением специализированной агентной библиотеки SPADE (Smart Python multi-Agent Development Environment¹³). Данная библиотека представляет собой прикладной набор инструментальных средств (платформу) для проектирования МАС на базе *Расширяемого протокола обмена сообщениями и информацией о присутствии* (Extensible Messaging and Presence Protocol — XMPP¹⁴).

К основным особенностям инструментария SPADE можно отнести:

- Полное соответствие стандартам проектирования FIPA.
- Реализация трёх протоколов межагентного взаимодействия: XMPP, P2P¹⁵ и HTTP¹⁶.
- Поддержка двух языков описания содержимого: FIPA-ACL и RDF.¹⁷
- Базовые классы SPADE позволяют инициировать делиберативные или реактивные процессы, т. е. на их основе можно реализовать рассмотренную в [разд. 2.1.4](#) гибридную архитектуру.

¹³ англ. Среда разработки интеллектуальных многоагентных систем на языке Python.

¹⁴ Протокол прикладного уровня базовой эталонной модели OSI [116, 117].

¹⁵ англ. peer-to-peer — равный к равному. Протокол описания оверлейных компьютерных сетей, основанных на равноправии участников.

¹⁶ англ. HyperText Transfer Protocol — «протокол передачи гипертекста». Клиент-серверный протокол прикладного уровня передачи данных в виде гипертекстовых документов.

¹⁷ англ. Resource Description Framework — это разработанная Консорциумом всемирной паутины (World Wide Web Consortium — W3C) модель для представления данных, в особенности — метаданных. RDF представляет утверждения о ресурсах в виде, пригодном для машинной обработки [118].

- Библиотека SPADE предоставляет удобный web-интерфейс для управления агентной системой.
- Наличие подсистемы уведомлений позволяет системе определять текущее состояние всех агентов в реальном времени.
- Библиотека SPADE позволяет объединять агентов в т. н. *организации* для решения непересекающихся задач, что даёт возможность создавать на её основе сложные агент-холонические системы управления ТПП.

Архитектурная модель библиотеки SPADE представлена на рис. 3.14.

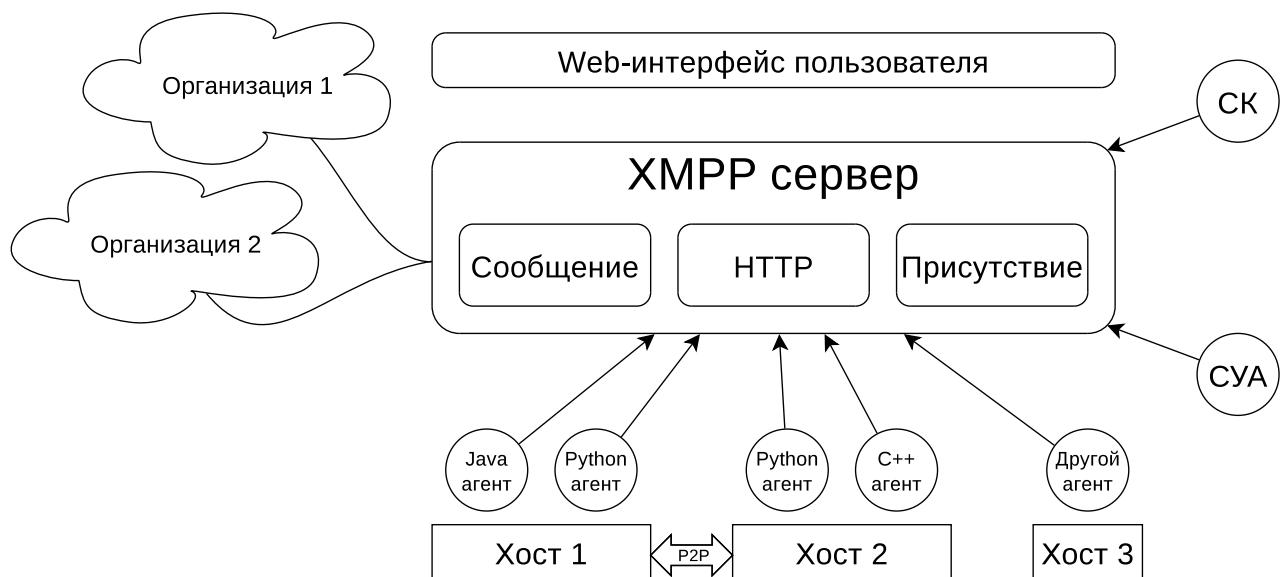


Рис. 3.4. Архитектура платформы SPADE

Нетрудно заметить, что представленная архитектура не требует, чтобы все агенты МАС создавались с помощью библиотеки SPADE. Платформа лишь представляет инструменты взаимодействия агентов внутри МАС и даёт возможность интеграции с другими платформами, соответствующими стандартам FIPA. Единственное требование: наличие специализированного адаптера, дающего возможность сторонним агентам работать с сообщениями инкапсулированными по протоколу XMPP.

Как уже было отмечено выше, всё взаимодействие агентов на уровне платформы осуществляется по протоколу XMPP. Изначально этот прото-

кол использовался для организации сервисов передачи коротких сообщений. На сегодняшний день он находит все более широкое применение в системах, где требуется оперативный обмен информацией в режиме, близкому к режиму реального времени.

3.2. Аппаратное обеспечение МАС

Рассматриваемая многоагентная ИУП ТПП строится на базе существующего на предприятии информационного пространства, следовательно, аппаратным обеспечение МАС являются персональные компьютеры и рабочие станции, используемые сотрудниками предприятия. Автором предложена методика моделирования информационного пространства приборостроительного предприятия, использующая концепции «облачных» вычислений и виртуальных рабочих мест, позволяющая настраивать и конфигурировать разработанную ИУП ТПП *до внедрения её на предприятии*.

В соответствии с предложенной методикой создан *Распределённый Виртуальный Испытательный Стенд (РВИС)*. РВИС предоставляет информационно-телекоммуникационную среду, предназначенную для тестирования средств информационного обеспечения, работающего в условиях крупного промышленного предприятия.¹⁸ РВИС построен на базе серверного кластера, на котором установлено специализированное программное обеспечение (*монитор виртуальных машин*), позволяющее создавать актуальную модель имеющегося на предприятии компьютерного оборудования.

С точки зрения реализации распределённый виртуальный испытательный стенд является «облачной» платформой виртуализации вычислительной и телекоммуникационной инфраструктуры предприятия.

¹⁸ Вопросы симуляции и эмуляции производственной управляющей многоагентной системы более подробно рассмотрены в работе [119].

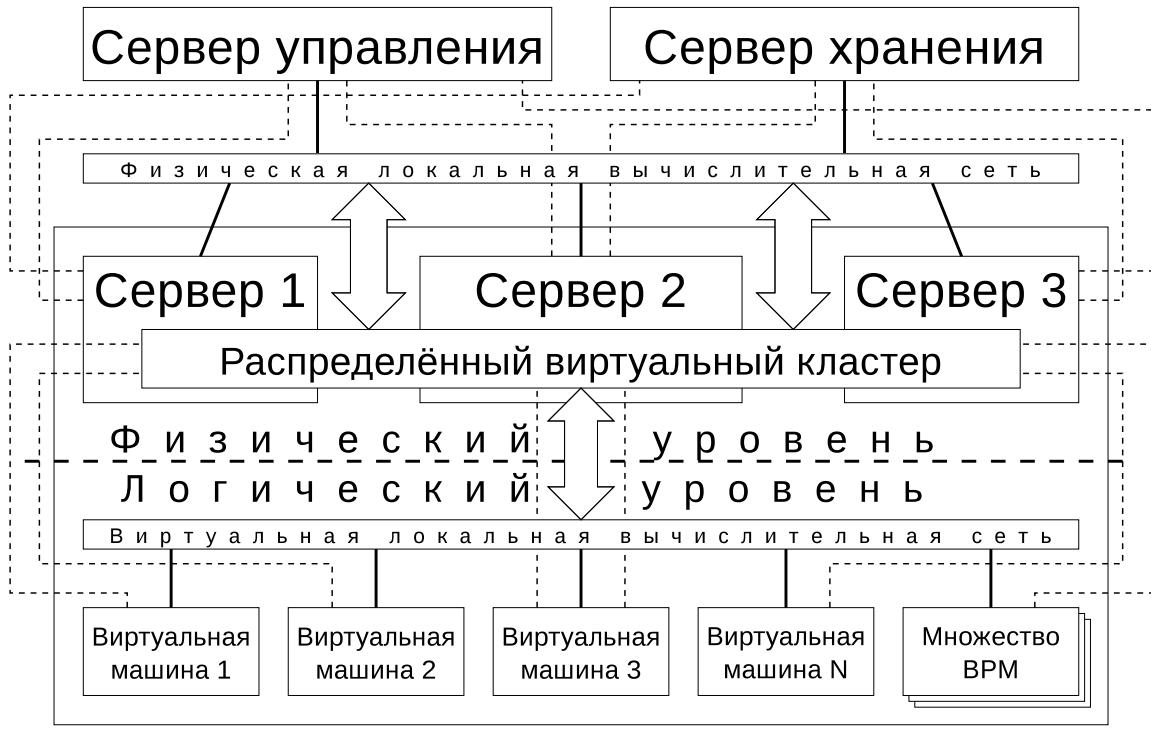
Внешний вид серверного кластера РВИС представлен на рис. 3.5, архитектура — на рис. 3.6.

Термин «облачный» [120] подразумевает абстрагирование аппаратного обеспечение за счёт консолидации ресурсов серверного кластера и создании на его основе множества виртуальных сущностей (компьютеров, рабочих станций, сетевых устройств и т. д.) с заданными характеристиками. Устройства, позволяющие пользователю напрямую работать с графическим интерфейсом, установленных на них программ через специализированный web-интерфейс [121], а также дающие возможность доступа к периферийному оборудованию, называются *виртуальными рабочими местами* (ВРМ), прочие вычислительные устройства — *виртуальными машинами*, телекоммуникационные устройства — *виртуальной локальной вычислительной сетью*. Состав аппаратного обеспечения, используемого при создание кластера РВИС представлен в табл. 3.1.

Разработанная архитектура соответствует концепции построения «облачных» платформ, называемой *Инфраструктура как услуга* (Infrastructure-as-a-Service — IaaS) [120]. Данный подход обеспечивает высокую гибкость при моделировании аппаратного обеспечения информационно-управляющей платформы технологической подготовки производства, вследствие возможности эмулировать поведения практически любых персональных компьютеров



Рис. 3.5. Внешний вид серверного кластера РВИС



Физические связи _____ Виртуальные связи

Рис. 3.6. Архитектура распределённого виртуального испытательного стенда

Таблица 3.1
Состав аппаратного комплекса РВИС

Оборудование	Характеристики	Количество
Сервер IBM x3650	2 процессора Intel Xeon E5405, 4 ГБ ОЗУ DDR3 ECC, НЖМД IBM 146 ГБ SAS, 2x1 Гб LAN, ОС Scientific Linux 6.1	2 шт
Сервер Intel 5400	2 процессора Intel Xeon E5405, 4 ГБ ОЗУ DDR3 ECC, НЖМД Seagate 320 ГБ SATA, 2x1 Гб LAN, ОС Scientific Linux 6.1	1 шт.
Сервер Intel 5500	2 процессора Intel Xeon E5606, 24 ГБ ОЗУ DDR3 ECC, НЖМД Seagate 300 ГБ SAS, 2x1 Гб LAN, ОС Scientific Linux 6.1	2 шт.
Сетевое хранилище	2 процессора Intel Xeon E5405, 4 ГБ ОЗУ DDR3 ECC, Disk Shelve IBM 4200 12 НЖМД Seagate 500 ГБ, 2x1 Гб LAN, ОС Scientific Linux 6.1	1 шт.

и рабочих станций, используемых на отечественных предприятиях: начиная с устаревших и заканчивая самыми современными.

Для управления распределённым виртуальным испытательным стеном используется удалённая графическая оболочка Virtual Machine Manager, написанная на языке Python. Данная графическая оболочка независима от основной платформы (серверного кластера распределённого виртуального испытательного стенда) и может быть установлена на любом внешнем по отношению к РВИС персональном компьютере, имеющим доступ к сети Интернет. К основным функциям оболочки относятся: создание новых виртуальных машин и включение их в кластер РВИС, доступ к консоли управления виртуальной машины, сохранение состояния виртуальной машины (создание точек восстановления), миграция виртуальных машин между хостами РВИС, балансировка нагрузки, а также доступ к виртуальным рабочим местам пользователей автоматизированных систем технологической подготовки производства. На [рис. 3.7](#) представлен интерфейс оболочки с открытыми окнами управления двумя виртуальными машинами и окном управления виртуальным рабочим местом. Доступ к виртуальным рабочим местам через консоль менеджера виртуальных машин является первым из реализованных способов удалённого управления виртуальными рабочими местами. Данный способ используется только для решения сервисных задач, связанных с установкой и настройкой программного обеспечения распределённого виртуального испытательного стенда. В [разд. 4.1](#) будет рассмотрен второй способ, основанный на концепции web-сервисов и не требующий установки дополнительного клиентского программного обеспечения.

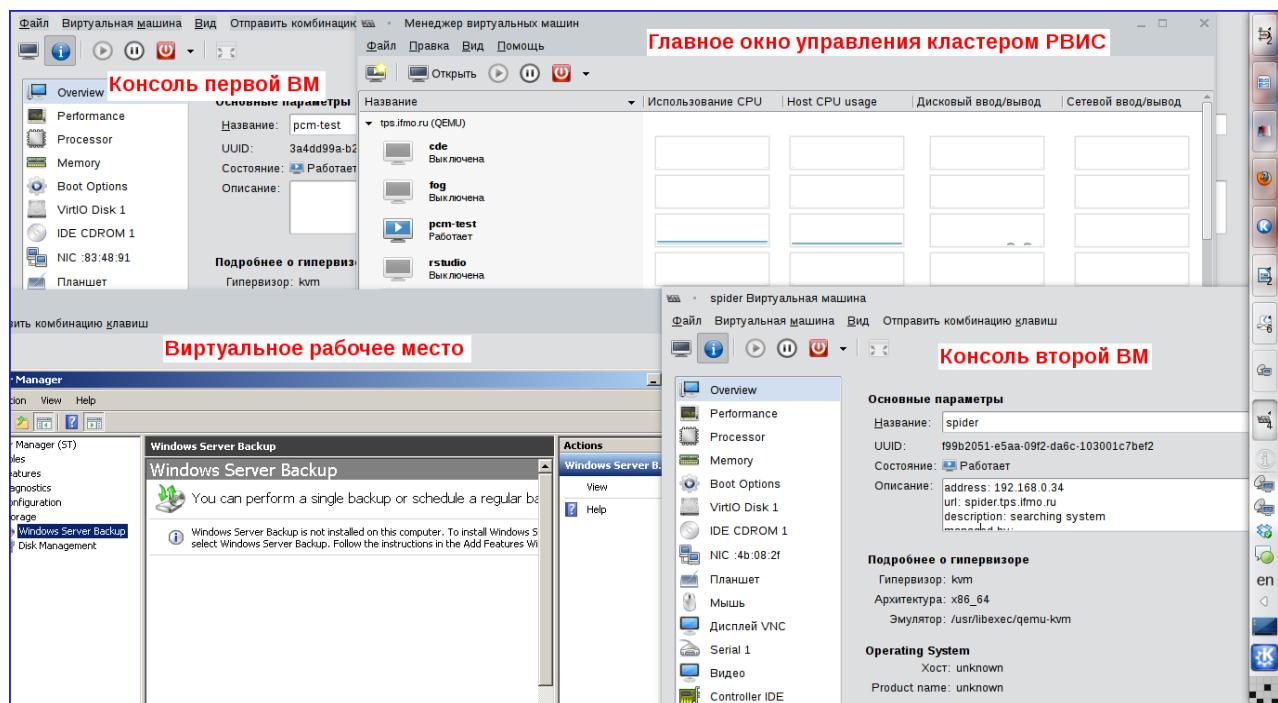


Рис. 3.7. Интерфейс управления кластером РВИС

3.3. Выводы и результаты по третьей главе

Разработан и программно реализован комплекс информационного обеспечения, позволяющий создать многоагентную систему в рассматриваемой предметной области. Комплекс включает в себя следующие модули и инструментальные средства:

1. Универсальный формат (язык) представления технологических данных и знаний, созданный в соответствии с основными принципами теории виртуального строкового пространства технологических данных и позволяющий агентам системы участвовать в интеграционной сети ИУП ТПП.
2. Набор базовых классов, реализующих модельprotoагента, т. е. агента, содержащего набор информационных контекстов, позволяющих менять поведение конкретных агентов при решении задач интеграции и взаимодействия с пользователем, а также модулей интеграции ВСПТД с интерфейсами СОМ и XML-RPC.

3. Онтологический словарь технологических концептов, позволяющий отделить семантику используемых в процессе проектирования ТПП понятий от непосредственно данных, которые они представляют. Онтологический словарь позволяет агентам беспрепятственно обменивать любыми технологическими данными и знаниями без дублирования понятий или двусмысленностей.
4. Протокол взаимодействия агентов, реализованный в соответствии со стандартами FIPA. Данный протокол позволяет агентам взаимодействовать по сети, обмениваясь специальными символными структурами, именуемыми речевыми актами или перформативами. Для передачи речевых актов используются различные транспортные протоколы прикладного уровня эталонной модели OSI.

Предложен метод моделирования информационного поля приборостроительного предприятия, основанный на концепциях «облачных» вычислений и виртуальных рабочих мест. На его основе разработан распределённый виртуальный испытательный стенд, позволяющий создать модель интеграционной сети проектируемой информационно-управляющей платформы для отработки и верификации исследуемых технологий.

3.4. Обзор стандартов FIPA

Фонд интеллектуальных физических агентов (Foundation for Intelligent Physical Agents — FIPA) — международная некоммерческая организация, созданная для разработки комплекта стандартов проектирования многоагентных систем. Первоначальным составом участников, в который вошли представители различных академических организаций и промышленных предприятий, был выработан набор положений, *де-юре* определяющих способы разработки многоагентных приложений.

К тому времени теория многоагентных систем была хорошо изучена научным сообществом, но, тем не менее, практически не использовалась для создания коммерческих приложений. Участники FIPA взяли на себя обязательства подготовить стандарты, которые должны были стать основой новой отрасли промышленности, способной создать междисциплинарный уровень взаимодействия различных приложений.

Согласно FIPA агент «обладает способностью предоставлять в рамках унифицированной и интегрированной исполнительной модели один или более сервисов, которые могут включать доступ к внешнему программному обеспечению, пользователям и коммуникационным возможностям»

Работа FIPA, основываются на следующих принципах:

1. Агентные технологии предоставляют новую парадигму решения старых и новых проблем.
2. Некоторые агентные технологии уже достаточно развиты.
3. Для использования некоторых агентных технологий требуется стандартизация.
4. Стандартизация внутренней структуры агентов не является приоритетной задачей, важнее разработать инфраструктуру и язык, дающие возможность агентам беспрепятственно взаимодействовать.

Первоначально предполагалось, что организация просуществует всего пять лет. За это время должны были быть стандартизованы лишь некоторые аспекты работы многоагентных систем; этот мандат был бессрочно продлён в 2001 году. С 2006 FIPA года является 11 комитетом *Института инженеров по электротехнике и электронике* (Institute of Electrical and Electronics Engineers — IEEE). Всего FIPA было принято 3 редакции стандартов: FIPA'97, FIPA'98 и FIPA2000. На сегодняшний день разработано 94 спецификации, наиболее значимые из них представлены в табл. B.1.

Ключевые спецификации FIPA

Индекс	Описание
00001	Спецификация абстрактной архитектуры (Abstract Architecture Specification)
00007	Спецификация языка описания содержимого (Content Languages Specification)
00008	Спецификация языка описания содержимого SL (SL Content Language Specification)
00023	Спецификация управления агентами (Agent Management Specification)
00024	Спецификация транспортировки сообщений агентов (Agent Message Transport Specification)
00025	Спецификация библиотеки протокола взаимодействия (Interaction Protocol Library Specification)
00037	Спецификация библиотеки коммуникативных актов (Communicative Act Library Specification)
00061	Спецификация структуры сообщения ACL (ACL Message Structure Specification)
00063	Спецификация агентного контроля (Control Agent Specification)
00082	Спецификация сетевого управления и резервирования (Network Management and Provisioning Specification)
00084	Спецификация протокола транспортировки сообщений агентов для HTTP (Agent Message Transport Protocol for HTTP Specification)
00086	Спецификация сервиса онтологий (Ontology Service Specification)
00094	Спецификация качества обслуживания (Quality of Service Specification)

Можно выделить следующие основные результаты работы FIPA:

- Разработан портфель стандартов определяющих параметры межагентного взаимодействия и обеспечивающих его вспомогательных сервисов.
- Стандартом FIPA2000 полностью охвачена абстрактная архитектура многоагентных платформ.
- Детально описаны язык взаимодействия агентов, язык описания содержимого и основные протоколы взаимодействия.
- Реализовано несколько свободных¹⁹ библиотек разработки многоагентных платформ.

¹⁹ Open source, с открытым исходным кодом.

- Создано агент-ориентированное расширение *Универсального языка моделирования* (Unified Modeling Language — UML), названное AUML [122].

3.4.1. Взаимодействие агентов

Язык взаимодействия агентов FIPA (FIPA Agent Communication Language — FIPA ACL) основывается на теории речевых актов [114, 115], декларирующей, что любое коммуникативное сообщение равноценно поступку (действию), речевой акт также известен как *перформатив*. FIPA-ACL оперирует 22 перформативами представленными в табл. B.2.

Inform и **Request** являются базовыми речевыми актами FIPA ACL, остальные представляют собой *макроопределения*, заданные в терминах этих перформативов. Результат базовых речевых актов определяют *предусловия*, т. е. условия, которые должны быть истинными, чтобы перформатив достиг цели, и *рациональный эффект* (rational effect) — та самая цель, достичь которую надеется отправитель перформатива.

Речевой акт Inform

Содержимое перформатива представляет собой *утверждение* (statement). Предуслугие заключается в том, что отправитель:

- убеждён, что содержимое сообщения истинно;
- намеревается сделать всё, чтобы получатель поверил, что содержимое сообщения истинно;
- пока сомневается, что получатель осведомлён о том, является ли содержимое истинным или нет.

Таблица 3.3
Речевые акты языка FIPA-ACL

Речевой акт	Описание
Accept Proposal	Принятие предложения другого агента
Agree	Согласие выполнить какое-то действие
Cancel	Уведомление о прекращении выполнения действия
Call for Proposal	Запрос предложений на выполнение действия
Confirm	Уведомление об истинности некоторого суждения в случае сомнений получателя
Disconfirm	Уведомление о ложности некоторого суждения в случае, когда получатель считает его истинным
Failure	Уведомление о неудаче
Inform	Уведомление об истинности некоторого суждения
Inform If	Сообщение о том было ли суждение истинным или нет
Inform Ref	Уведомление о соответствии полученного объекта некоторому дескриптору
Not Understood	Осознание необходимости сделать что-либо, но отсутствии понимания что именно
Propagate	Сообщение о необходимости принять сообщение и передать его другим агентам в соответствии с заданным дескриптором
Propose	Предложение выполнить некоторое действие с учётом заданных условий
Proxy	Уведомление о необходимости передать сообщения другим агентам в соответствии с заданным дескриптором
Query If	Запрос о том было ли суждение истинным или нет
Query Ref	Запрос о соответствии имеющегося объекта некоторому дескриптору
Refuse	Отказ от выполнения некоторого действия без сообщения причины
Reject Proposal	Отклонение предложения выполнить некоторое действие во время проведения переговоров
Request	Запрос на выполнение некоторого действия
Request When	Запрос на выполнение некоторого действия в случае получения предложения
Request Whenever	Запрос на выполнение некоторого действия постоянно при получении предложений
Subscribe	Желание получать сообщения в соответствии с некоторым условием до тех пор, пока оно не изменится

Формальная семантика коммуникационного акта **Inform** может быть задана следующим образом:

$$\langle s, \text{inform}(r, p) \rangle$$

$$FP: \mathcal{B}_s p \wedge \neg \mathcal{B}_s (\mathcal{B}_{\text{if}} r p \vee \mathcal{U}_{\text{if}} r p) \quad (3.8)$$

$$RE: \mathcal{B}_r p, \text{ где}$$

<i>s</i>	отправитель сообщения,
<i>r</i>	получатель сообщения,
<i>p</i>	содержимое сообщения (утверждение),
\mathcal{B}	модальный оператор «убеждён»,
\mathcal{U}	модальный оператор «сомневается»,
$\mathcal{B}_{\text{if}} r p$	сокращённая запись от $\mathcal{B}_r p \vee \mathcal{B}_r \neg p$,
$\mathcal{U}_{\text{if}} r p$	сокращённая запись от $\mathcal{U}_r p \vee \mathcal{U}_r \neg p$,
<i>FP</i>	предусловия выполнимости (feasibility precondition),
<i>RE</i>	предполагаемый эффект сообщения (rational effect).

Речевой акт **Request**

Содержимое перформатива представляет собой *действие* (action). Предусловие заключается в том, что отправитель:

- намеревается сделать всё, чтобы было выполнено действие, указанное в содержимом сообщения;
- убеждён, что получатель способен выполнить это действие;
- сомневается, что к настоящему моменту у получателя уже есть намерение выполнить это действие.

Формальная семантика коммуникационного акта **Request** аналогична семантике **Inform**.

В соответствии со стандартом любой агент должен иметь возможность принимать и обрабатывать любые сообщения. Если агент по каким-либо причинам не может обработать сообщение, он должен ответить на него перформативом *Не понимаю* (Not understood). Также FIPA определён набор коммуникационных протоколов, каждый из которых описывают какую-то последовательность речевых актов для создания различных коммуникационных сетей.

3.4.2. Коммуникационные подуровни FIPA

Стек коммуникационных протоколов FIPA базируется на *Базовой эталонной модели взаимосвязи открытых систем* (Open Systems Interconnection basic reference model — OSI) [116, 117] и также условно разделён на 7 коммуникационных подуровней. Рассмотрим эти подуровни подробнее:

- *Подуровень 1 (Транспортный)*: Определяет транспорт для передачи сообщений FIPA-ACL. В соответствии с текущей версией стандарта используются три основных транспортных протокола: *Межбрюкерный протокол для Интернет* (Internet InterORB Protocol — IIOP), *Беспроводной протокол передачи данных* (Wireless Application Protocol — WAP) и *Протокол передачи гипертекста* (HyperText Transfer Protocol — HTTP).
- *Подуровень 2 (Кодирование)*: Вместо обычных байт-кодированных сообщений стандарт FIPA позволяет оперировать высокоуровневыми структурами данных, например, XML, строками (String) или специализированным битовым форматом Bit-Efficient. Последний используется при работе через низкоскоростные соединения.
- *Подуровень 3 (Обмен сообщениями)*: В соответствии со стандартами FIPA на этом уровне задаются дополнительные параметры, специфи-

цирующие полезную нагрузку сообщения, например адреса отправителя и получателя, тип сообщения (коммуникативного акта), таймауты ответов и т. д. Кодировка передаваемого сообщения не регламентируется, оставляя право разработчикам выбирать наиболее подходящую им кодовую страницу.

- *Подуровень 4 (Онтология)*: Каждое поле передаваемых в сообщении данных может быть связано с некоторой концептуальной моделью или, иными словами, онтологией. Но, несмотря на то, что связь полей и онтологий явно регламентирована, в стандарте FIPA не уточняется в каком виде они должны быть представлены, поэтому разработчики могут использовать любые предметно-ориентированные онтологии, либо web-онтологии, если того требует конкретная производственная задача.
- *Подуровень 5 (Описание содержимого)*: Актуальное содержимое передаваемого сообщение может быть представлено в любой форме. Тем не менее, стандартами FIPA регламентированы некоторые общие положения использования формул алгебры логики, предикатов и алгебраических операций. Наиболее распространённым языком описания содержимого передаваемых сообщений на сегодняшний день является FIPA-SL. В данном языке реализованы все основные логические операции такие как: *or* (или, \vee), *and* (и, \wedge), *not* (не, \neg), *equiv* (равно, $=$), *implies* (импликация, \implies) и т. д. А также предикаты *any* (любой), *all* (все) и др.
- *Подуровень 6 (Коммуникационные акты)*: На данном уровне реализован простой классификатор коммуникативных актов (перформативов), см. табл. [B.2](#).
- *Подуровень 7 (Протокол взаимодействия)*: Обычно сообщения передаются не только от агента к агенту, но и участвуют в некоторой

коммуникативной последовательности. Стандартами FIPA определены несколько протоколов для передачи и маршрутизации сообщений в МАС.

3.4.3. Управление агентами

В соответствии со стандартами FIPA, вторым по важности аспектом работы МАС (после организации взаимодействия агентов) является управление поведением отдельных агентов. В данном контексте особое внимание уделяется разработке внутренних правил и норм, которым должны следовать агенты, т. е. созданию логической модели, в соответствии с которой будут создаваться, взаимодействовать, мигрировать и работать все агенты МАС. Референсная (эталонная) модель управления агентами изображена на [рис. 3.8](#).

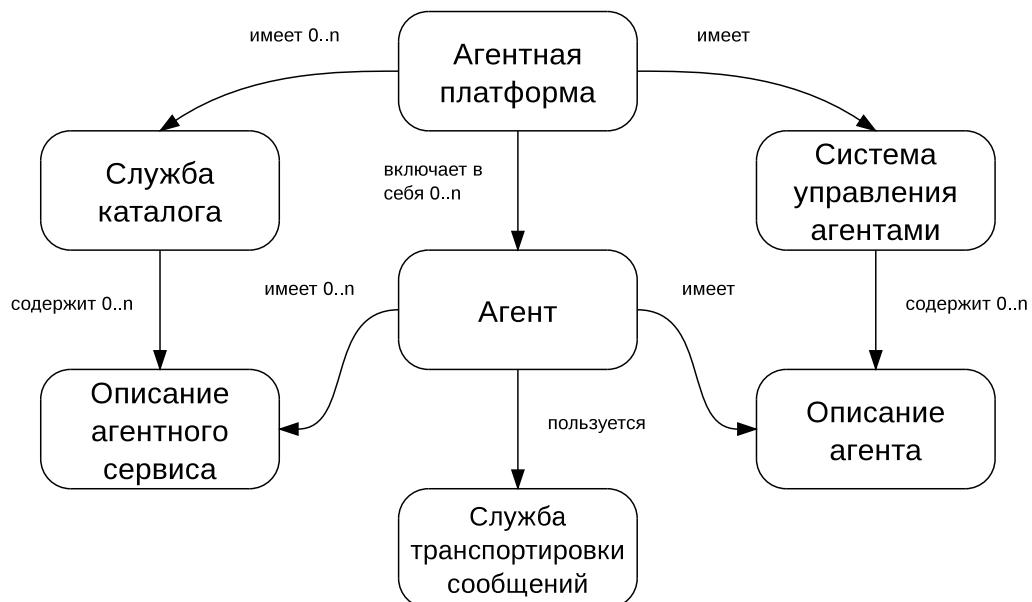


Рис. 3.8. Референсная модель управления агентами

Агентная платформа (АП) Обеспечивает физическую инфраструктуру, в которой будет развёрнута МАС. Включает в себя вычислительное оборудование, операционные системы, компоненты управления агентами

FIPA (см. ниже), самих агентов и вспомогательное программное обеспечение. Реализация самой MAC не регламентируется FIPA, что позволяет разработчикам создавать системы в соответствии с текущими потребностями производственной или технологической среды. АП может иметь блочную структуру, компоненты которой физически располагаются в разных узлах вычислительной сети, единственное требование — несколько агентов не могут одновременно находиться на одном хосте АП. FIPA рассматривает только способ реализации коммуникаций между «родными» для данной АП агентами и внешними или динамически регистрирующимися на АП агентами.

Агент Агент является ключевым компонентом АП, агент представляет собой некоторый вычислительный процесс²⁰, являющийся частью распределённой АП. Как правило, агент может предоставлять один или несколько вычислительных сервисов в *унифицированной и интегрированной исполнительной модели*, которая может включать доступ к внешнему программному обеспечению, интерфейс пользователя и коммуникационные возможности.

Структура и реализация данных сервисов может быть произвольной, FIPA регламентирует только структуру и кодировку сообщений, используемых для взаимодействия агентов. Агент должен иметь как минимум одного владельца (например, на основе принадлежности к организации или человеку) и иметь некоторый уникальный идентификатор, позволяющий однозначно выделить его внутри агентного универсума. Также агент может регистрироваться с множеством транспортных адресов, по которым с ним можно контактировать, а также быть брокером ресурсов (разд. 2.2.3) для доступа к программному обеспечению.

²⁰ В соответствии с ГОСТ 19781-90 [108] процесс обработки данных (computational process) есть система действий, реализующая определенную функцию в системе обработки информации и оформленная так, что управляющая программа данной системы может перераспределять ресурсы этой системы в целях обеспечения мультипрограммирования.

Служба каталога (СК) Служба каталога является optionalным компонентом АП, который предоставляет сервис *жёлтых страниц* всем агентам МАС. СК хранит точный и полный список агентов системы и по требованию должна выдавать актуальную информацию о них. АП может включать в себя несколько СК, при этом СК регистрируют друг друга, образуя федерацию.

Каждый агент, который хочет заявить о предоставляемом сервисе (сервисах), должен найти СК и *оставить запрос на регистрацию*. Факт регистрации агента в службе каталога не подразумевает никаких обязательств с его стороны. Агент в любой момент может *отправить запрос с просьбой удаления или модификации* записей о себе из каталога и служба каталога обязана удовлетворить его. Кроме того, агент может *отправить запрос на поиск* интересующих его сервисов, но, исходя из сказанного выше, ответ СК может быть недостоверным. СК имеет право временно ограничить доступ к информации в своём каталоге агентам, пытающимся изменить данные о себе и проверить их полномочия доступа.

Система управления агентам (9999999999999999) СУА является обязательным компонентом АП и отвечает за управления такими операциями АП как создание и удаление агентов, а также контроль за перемещениями агентов в/из АП. Каждый агент обязан зарегистрироваться в СУА и получить свой уникальный идентификатор, который сохраняется в каталоге СУА вместе с текущим состоянием агента (например, активен, работа приостановлена, в режиме ожидания) и его транспортными адресами. Записи об агентах могут изменяться только с разрешения СУА.

Жизненный цикл агента ([рис. 3.9](#)) завершается в момент *снятия его с учёта* в СУА, при этом его уникальный идентификатор удаляется из ката-

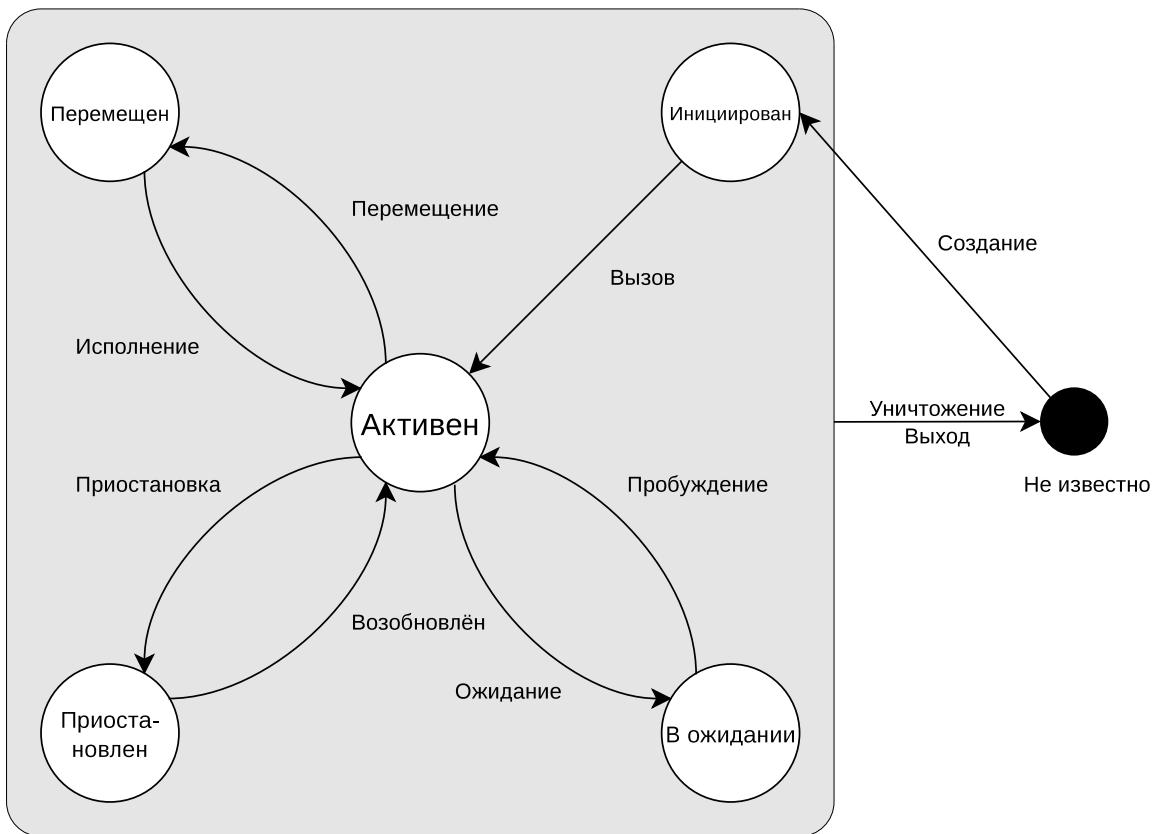


Рис. 3.9. Диаграмма жизненного цикла агента

лога и становится доступным для вновь регистрирующихся агентов²¹. СУА предоставляет доступ к своему каталогу для поиска записей о состояниях агентов АП. СУА может потребовать, чтобы агент выполнил определённое административное задание, например, прекратил свою работу. Также СУА обладает полномочиями принудительного выполнения заданной административной операции в случае, если её запрос игнорируется. В АП может быть только одна СУА.

Служба транспортировки сообщений (СТС) СТС отвечает за транспортировку сообщений FIPA-ACL между агентами внутри одной АП, а также между агентами разных АП. Сообщения помещаются в конверт, содержащий в себе различную информацию об отправителе, получателе, протоколе и т. д. Общая структура сообщения представлена на рис. 3.10.

²¹ В разд. 4.3 автором будет предложен способ, позволяющий генерировать одноразовые идентификаторы агентов, что позволит отказаться от необходимости подобного поведения СУА.



Рис. 3.10. Структура сообщения FIPA

Стороннее программное обеспечение (СПО) СПО представляет собой:

1. Совокупность неагентного исполняемого кода, доступного через интерфейс агента.
2. Не относится к АП.
3. Агенты могут обращаться к СПО для того, чтобы, например:
 - добавить новые сервисы;
 - приобрести новые коммуникационные протоколы;
 - приобрести новые протоколы/алгоритмы безопасности;
 - приобрести новые протоколы переговоров;
 - обратиться к инструментам поддерживающим миграцию;
 - и т. д.

3.4.4. Абстрактной архитектура FIPA

В дополнение к стандартам, описывающим взаимодействие агентов и управление агентной платформой в период с 2000 по 2002 годы FIPA была описана абстрактная агентная платформа (ААП). ААП позволила более

чётко формализовать наиболее важные механизмы агентного взаимодействия. Конечная цель данного подхода — дать возможность создавать такие МАС, которые могли бы быть легко включены в любую вычислительную среду посредством создания в ней дополнительного (внешнего по отношению к среде) агентного слоя.

МАС, созданные в соответствии с разработанными ранее стандартами FIPA'97 и '98 могут с некоторыми ограничениями взаимодействовать с системами, построенными на базе ААП через специализированные шлюзы. МАС, использующие архитектуру FIPA'2000 также должны использовать шлюзы, но уже без каких-либо ограничений.

Конкретная МАС, созданная на основе ААП может быть реализована разными способами, например, как компонент на Java, как СОМ объект, самостоятельной Lisp программой или TCL скриптом. Он может исполняться как нативный процесс на некотором физическом компьютере под управлением операционной системы, или поддерживаться интерпретатором, таким как Java Virtual Machine или TCL системой. Взаимосвязь между агентами и их вычислительными контекстами специфицируется в процессе жизненного цикла агента. ААП не затрагивает жизненный цикл агента, т. к. часто он реализуется по разному в конкретных вычислительных средах, реализация ААП должна описывать эти моменты.

3.5. Прикладные библиотеки разработки

многоагентных систем

В общем случае, прикладная библиотека является программным обеспечением промежуточного уровня (middleware), т. е. программным обеспечением для создания программного обеспечения, в рассматриваемой предметной области — многоагентных систем технологического назначения. С целью

сохранения единства терминологии для обозначения данного класса ПО автором будет использоваться определение *Многоагентная платформа* (МП²²).

На сегодняшний день существует огромное количество прикладных библиотек разработки МАС. Подробный и всесторонний анализ большинства из них можно найти в работе [123]. В рамках представленного исследования автором проведён анализ лишь трёх систем моделирования и разработки МАС. Это обусловлено тем, что подавляющая часть программных средств разработки либо выпускаются под несвободной (проприетарной) лицензией²³, либо реализует функциональность МАС лишь в узкой предметной области, не позволяющей использовать её для разработки полноценной ИУП ТПП. Выбранные для исследования многоагентные платформы представляют три разных подхода к моделированию МАС, что позволяет проследить основные современные направления в развитии данного ПО и выбрать наиболее подходящую библиотеку для разработки прототипа ИУП ТПП.

3.5.1. JADE

JADE (Java Agent Development Framework²⁴) — это программное обеспечение промежуточного слоя, разработанное компанией TILAB, предназначеннное для создания распределённых многоагентных приложений на основе транспортной архитектуры «точка-точка». И интеллект, и инициатива, и информация, и ресурсы, и контроль могут быть полностью распределены по мобильным терминалам так же, как и по компьютерам выделенной сети. Среда может динамично взаимодействовать с узлами, которые в терминологии JADE называются агентами. Агенты то появляются, то исчезают в

²² По аналогии с многоагентной средой ([разд. 2.2.1](#)) и многоагентной системой ([разд. 2.2.2](#)).

²³ Что противоречит требованию создания открытой системы, см. . . .

²⁴ англ. *Фреймворк* (см. определение на с. 58) для разработки агентов на языке Java.

системе в соответствии с потребностями и требованиями программной среды. Коммуникации между узлами не зависят от типа сети (проводная, беспроводная). Они являются полностью симметричными, и каждый узел может, как инициировать запросы, так и отвечать на них [113].

МП JADE полностью разработана на языке Java. Перечислим основополагающие принципы платформы:

- *Функциональная совместимость* — продукт JADE разработан в соответствии со спецификациями FIPA. Как следствие JADE-агенты могут взаимодействовать со сторонними агентами, поддерживающими этот стандарт.
- *Портируемость и единообразность* — продукт JADE предоставляет гомогенный набор прикладных программных интерфейсов (API), которые не зависят ни от базового устройства сети, ни от версии платформы Java.
- *Простота использования* — набор простых и интуитивно-понятных интерфейсов API прячет сложную логику ПО промежуточного слоя от пользователя.
- *Принцип «разрабатывать по средствам»* — программистам нет необходимости использовать все возможности, которые предоставляет ПО промежуточного слоя. От программистов не требуется знать что-либо о неиспользуемых функциях платформы. Ни одна из незадействованных функций не создает дополнительные накладные вычислительные расходы.

Архитектурная модель

МП JADE включает как программные библиотеки, требуемые для разработки прикладных агентов, так и среду исполнения, которая предоставляет

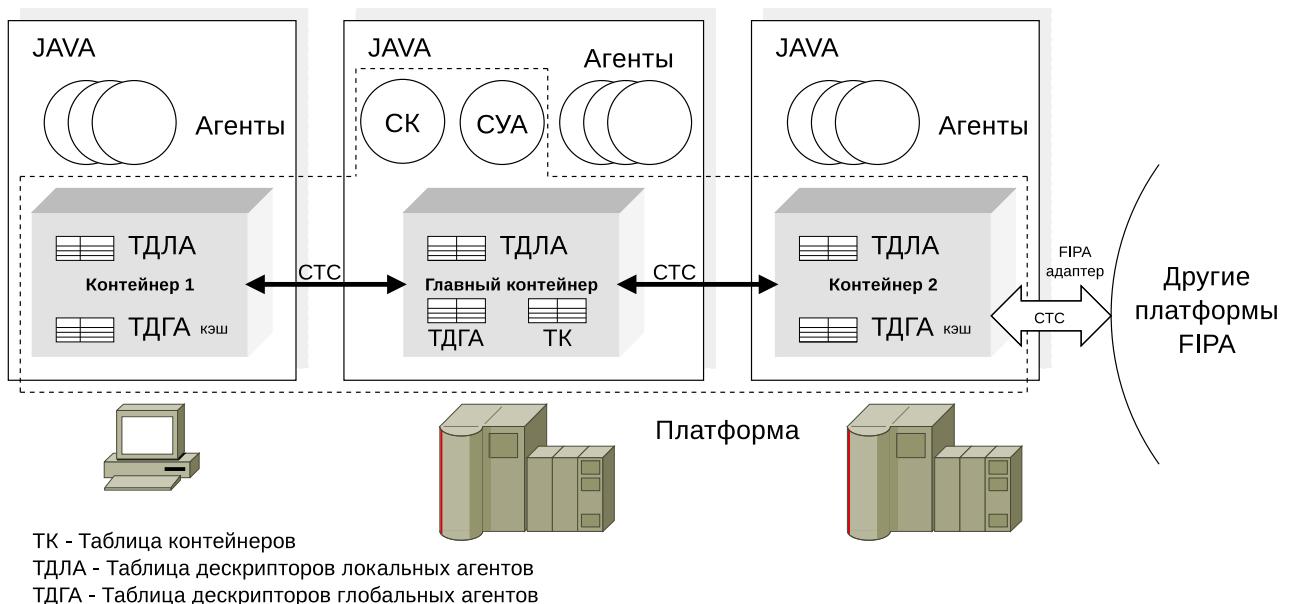


Рис. 3.11. Архитектура платформы JADE

базовые службы и которая должна быть активна на устройстве до того, как на нем будет исполняться агент. Каждый экземпляр JADE во время исполнения называется контейнером (так как он «содержит» агентов). Набор всех контейнеров называется платформой и предоставляет однородный слой, который прячет от агентов (а также от разработчиков приложений) сложность и распределённый характер механизмов, расположенных на более низком уровне ([рис. 3.11](#) и [рис. 3.12](#)).

В процессе исполнения JADE занимает около 100 КБ оперативной памяти, но этот размер может быть уменьшен до 50 КБ за счет совместной компиляции JVM и JADE. Продукт JADE является универсальным и более того, он не только удовлетворяет ограничениям сред с ограниченными ресурсами, но он интегрирован в сложные архитектуры, такие как .NET и J2EE, где JADE становится службой для исполнения прикладных проактивных приложений с несколькими взаимодействующими сторонами.

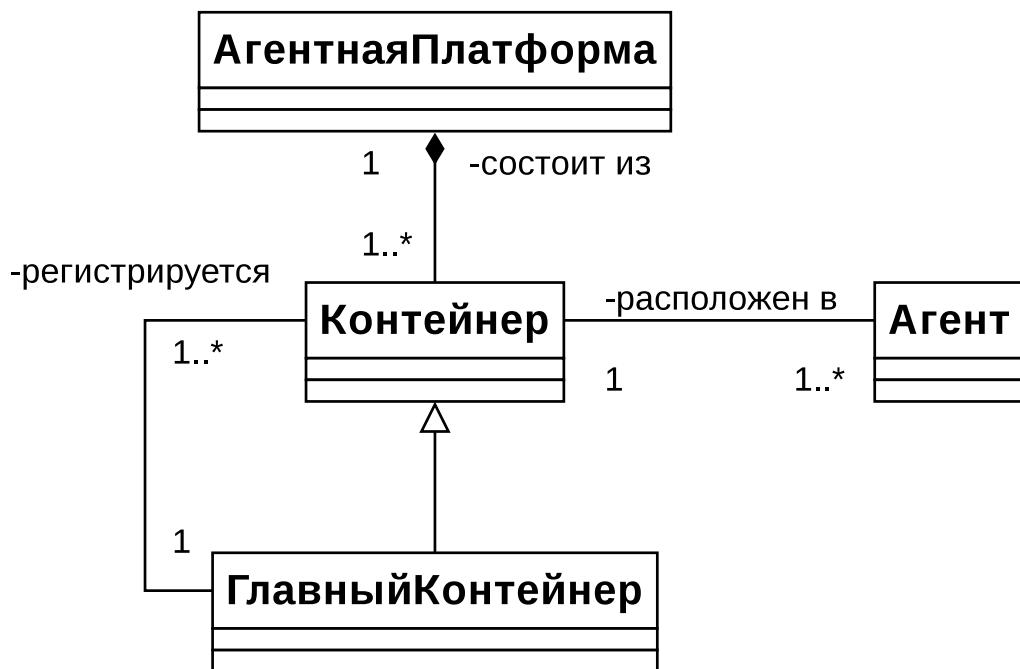


Рис. 3.12. UML-диаграмма взаимодействия базовых элементов платформы JADE

Функциональная модель

С функциональной точки зрения программный инструмент JADE предоставляет базовые службы, необходимые для распределённых одноранговых приложений типа «точка-точка» в стационарных и мобильных средах. Он позволяет каждому агенту динамически обнаруживать других агентов и обмениваться с ними сообщениями в соответствии с парадигмой распределённых многоагентных систем. С точки зрения приложения каждый агент идентифицируется уникальным именем и предоставляет набор служб. Он может регистрировать и модифицировать свои службы и/или искать агентов, предоставляющих данные службы. Также агент способен контролировать свой жизненный цикл и, в частности, общаться с другими агентами.

Агенты общаются путём обмена асинхронными сообщениями, модель коммуникаций почти повсеместно применима для распределённых и слабосвязанных коммуникаций, т. е. коммуникаций между гетерогенными сущностями, которые ничего не знают друг о друге. Для осуществления коммуникации

агент только посыпает сообщение получателю. Идентификатором служит имя агента и, как следствие, нет временной зависимости между общающимися агентами. Не требуется, чтобы отправитель и получатель были запущены в одно и тоже время. Получатель даже может не существовать (или еще не существовать) или может быть не известен явно отправителю.

Несмотря на такой тип коммуникаций, при коммуникациях сохраняется безопасность передачи сообщений. Для приложений, которым требуются безопасные каналы, платформа JADE предоставляет надлежащие механизмы для аутентификации и проверки прав, назначенных агентам. Следовательно, когда требуется, приложение может проверить истинность идентификатора отправителя сообщения и предотвратить выполнение недозволенных действий (например, агенту разрешено получать сообщения от агента-босса, но запрещено посыпать ему сообщения). Все сообщения, которыми обмениваются агенты, передаются внутри обёртки, которая включает только необходимую транспортному уровню информацию. Что позволяет, помимо всего прочего, шифровать содержимое сообщения отдельно от его обёртки.

Структура сообщения задается при помощи языка FIPA-ACL, определенного спецификациями FIPA, и включает такие поля, как переменные, определяющие контекст, получателя сообщения и время, в течение которого будет ожидаться ответ, нацеленные на поддержку сложных взаимодействий и множественные параллельные беседы. Для того, чтобы поддерживать реализацию сложных коммуникаций в дальнейшем, программное обеспечение JADE предоставляет набор каркасов типичных паттернов взаимодействий для выполнения определенных задач, таких как ведение переговоров, аукционы и делегирование задач. Используя эти скелетоны (реализованные как абстрактные классы Java), программисты смогут избежать таких проблем, как задачи синхронизации, таймауты, условия ошибок, и, в общем, всех тех аспектов, которые тесно не связаны с логикой приложения.

Для того чтобы облегчить реализацию механизмов создания и обработки содержимого сообщений, агентная платформа JADE предоставляет поддержку для автоматического прямого и обратного конвертирования в формат, подходящий для обмена контентом, включая XML и RDF, и формат, подходящий для управления контентом (например, объекты Java). Эта поддержка встроена в некоторые инструменты создания онтологий, позволяя программистам создавать графическое представление онтологий.

Для того чтобы расширить масштабируемость или удовлетворить ограничения сред с ограниченными ресурсами, ПО JADE предоставляет возможность выполнения множественных параллельных задач в одном потоке Java. Несколько элементарных задач, таких как коммуникации, могут быть соединены для формирования более сложных задач, представленных как конечные автоматы.

В средах J2SE и Personal Java программа JADE поддерживает мобильность кода и мобильность состояния. То есть агент может остановить работу на хосте и мигрировать на другой удаленный хост (нет необходимости копировать код или устанавливать агента на этом хосте заранее), а затем возобновить исполнение с того места, где он был остановлен. Эта функциональность позволяет, к примеру, распределение вычислительной нагрузки в режиме исполнения путем перемещения агентов на менее загруженные машины без какого-либо воздействия на приложение.

Платформа также содержит службу имен (которая проверяет, что каждый агент имеет уникальное имя) и службу «желтых страниц», которая может быть распределена по нескольким хостам. Можно создать графы федерации, чтобы определить организованные области агентных служб. Другая очень важная функция заключается в доступности богатого набора графических инструментов, поддерживающих и стадию отладки, и стадию управления/наблюдения жизненного цикла приложения. Посредством этих инструментов

стало возможно удалённо управлять агентами (даже в том случае, когда они уже развернуты и запущены).

Технические/функциональные характеристики

- Распределенные, многосторонние приложения с коммуникациями «peer-to-peer»
- Соответствие стандарту FIPA
- Управление жизненным циклом агента Службы «белые страниц» и «желтых страниц» с возможностью создания графов федераций в онлайн режиме
- Графические инструменты, поддерживающие стадии отладки, управления, мониторинга
- Поддержка миграции агентного кода и запущенного состояния
- Поддержка сложных протоколов взаимодействия (например, contract-net)
- Поддержка создания и управления контентом сообщений, в т.ч. в формате XML и RDF
- Поддержка интеграции со страницами JSP
- Поддержка безопасности на уровне приложения (на данный момент – только для платформы J2SE)
- Возможность выбора транспортных протоколов в режиме исполнения
- Доступные транспортные протоколы: JAVA-RMI, JICP (используется по-умолчанию), HTTP и ИОР
- Сетевое окружение Проведены испытания на базе Bluetooth, GPRS, W-LAN, Internet.
- Терминалы Все терминалы, поддерживающие среду Java MIDP 1.0, или Personal Java, или J2SE.

3.5.2. Repast

Repast (Recursive Porus Agent Simulation Toolkit²⁵) — многофункциональная МП проектирования многоагентных систем. Программный комплекс Repast перенял много идей от инструментального средства Swarm (Swarm – это инструментальная система моделирования на основе агентов, написанная на языке Objective C). Платформа Repast имеет различные реализации для моделирования с использованием различных языков программирования.

²⁵ англ. Инструментарий моделирования разреженных рекурсивных агентов.

Прикладная библиотека Repast — бесплатное инструментальное средство с открытым исходным кодом. Repast поддерживает разработку чрезвычайно гибких многоагентных систем, ориентированных в первую очередь на развитую систему межагентного взаимодействия.

На сегодняшний день созданы три реализации инструментария Repast: для языка программирования Java (Repast J), для платформы Microsoft .NET (Repast.NET) и Repast для интерпретируемого языка Python (Repast Py).

МП Repast поддерживает большое количество различных функций, в том числе:

- В стандартную библиотеку входит большое количество шаблонов агентов, но несмотря на это инженеру по знаниям предоставляется полный набор инструментов по настройке и конфигурированию проектируемых агентов. Это позволяет достичь высокой гибкости и минимизировать объём программного кода.
- Платформа полностью соответствует базовым принципам разработки объектно-ориентированных систем.
- Repast включает в себя планировщик дискретных событий, поддерживающий как последовательные, так и параллельные операции.
- Наличие графических инструментов разработки, симуляции и визуализации работы агентной системы.
- Возможность динамически просматривать и изменять свойства, поведение, а также модели свойств агентов.
- В состав Repast входят модули разработки генетических алгоритмов, нейронных сетей, генерации случайных чисел и специализированных математических выражений и встроенные средства моделирования динамики многоагентных систем.

- Все инструментальные средства Repast поддерживает работу с операционными системами Windows, Mac OS, Linux. Она может работать как на персональных компьютерах, так и на масштабных научных вычислительных кластерах.

С 2007 года состав инструментария Repast расширился включением в него пакета Repast Symphony (Repast S), основанного на Repast. Repast S существенно расширяет базовые возможности агентной платформы, предоставляя среду разработки моделей и среду исполнения.

Также к возможностям Repast S можно отнести:

- Совместное использования языков Java, Groovy, Python, .Net.
- Возможность интеграции с множеством сторонних пакетов, включая R statistics environment [124], плагин для работы с SQL-запросами, подключаемые модули анализа трафика *ORA [125] и Pajek [126], инженерный пакет визуализации VisAD [127], платформа интеллектуального анализа данных Weka [128], многие популярные форматы электронных таблиц, вычислительная математическая среда MATLAB, дизайнер визуальных отчётов iReport [129].
- Моделирование трехмерных сред
- Интеграция с библиотекой сетевого моделирования JUNG [130].
- Импорт данных из электронных таблиц Microsoft Excel.
- Поддержка сквозной имитации XML.
- Распределённые вычисления с помощью пакета Terracotta [131].

Инструментарий Repast S, в первую очередь представляет удобный графический интерфейс моделирования. Интерфейс среды исполнения Repast S (рис. 3.13) создан на основе каркаса Simphony Application Framework (SAF). SAF — это интегрированная платформа разработки приложений, созданная с целью упростить программирование графических настольных приложений.

Набор шаблонов SAF реализует функцию управления жизненным циклом и содержит другие полезные инструменты, такие как плавающие окна и вкладки.

МП Repast S содержит новые функции для определения поведения агента и возможность динамической сборки компонентов модели из обычных объектов Java, задаваемых пользователем. Тем не менее, следует отметить, что для создания таких объектов может использоваться и любой другой метод (начиная от прямого кодирования до создания классов-оберток (wrappers) над готовыми сторонними компонентами, установлением связей с распределенными информационными системами).

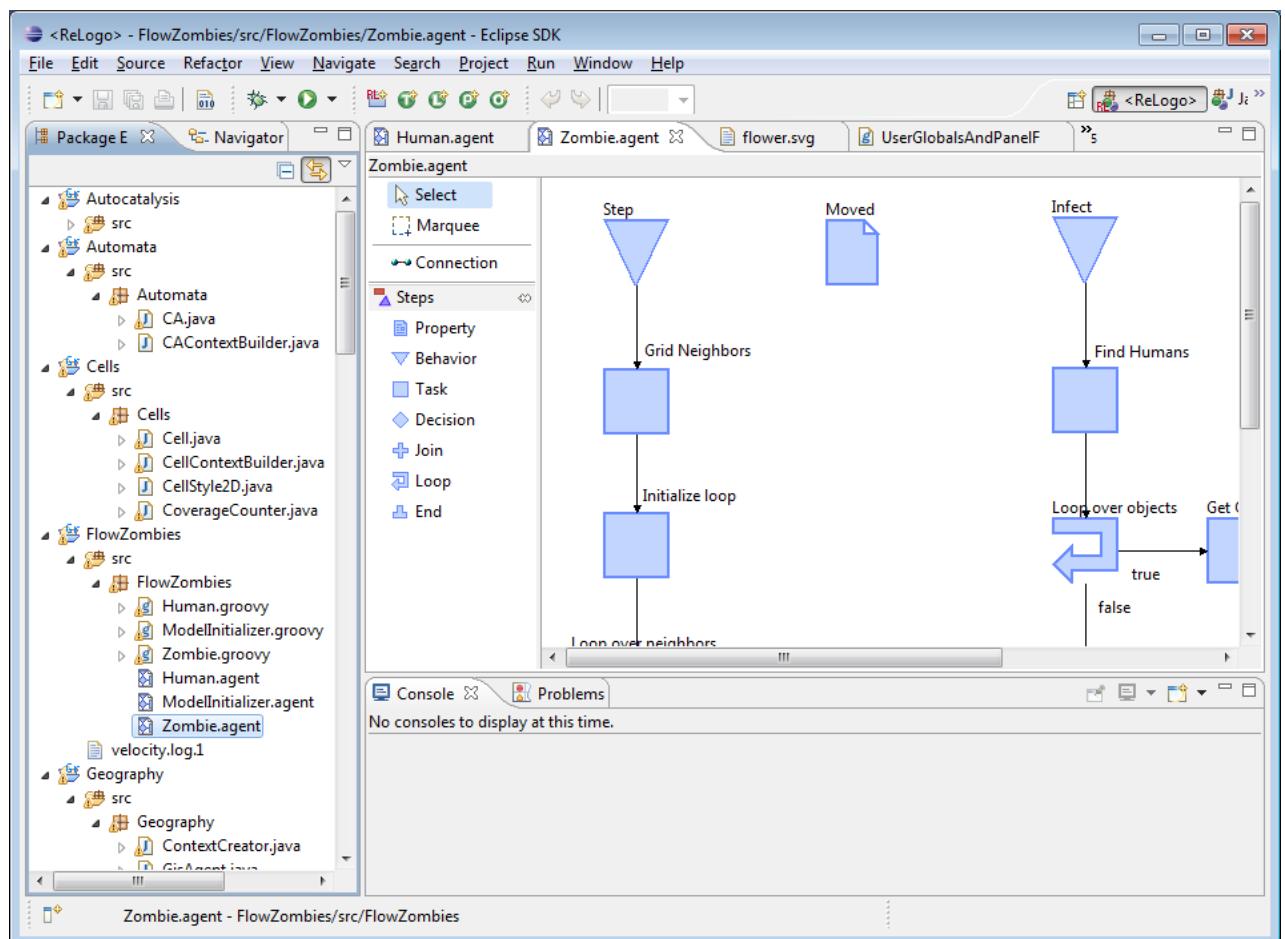


Рис. 3.13. Графический интерфейс среды Repast S

Аннотации и настройки

Программист, работающий с МП Repast S может использовать аннотации языка Java. Аннотации — это тэги метаданных, которые компилируются в файлы бинарного класса. Аннотации не выполняются также как код, поэтому похожи на обычные комментарии Java. Однако, в отличие от комментариев, аннотации хранятся в скомпилированных бинарных файлах. Это позволяет выполняющимся программам Java получать метаданные и использовать их в работе. Т. е. в процессе проектирования разработчики могут явно помечать (или аннотировать) код для специальной обработки средой исполнения Repast S. Данная функциональность используется, к примеру, для создания т. н. «сторожевого таймера» (watchdog). Используя данную функциональность можно создать агента, который будет активизироваться всякий раз, когда меняется какой-либо атрибут ближайшего к нему агента, т. е. можно будет реализовать механизм *поиска по необходимости*. Например, агент, связанный с системой управления ресурсами инструментального склада будет динамически отслеживать распределение инструмента по участкам, предотвращая перерасход инструмента.

МП Repast S использует два основных типа настроек:

- *Дескрипторы модели*, определяющие, что может содержаться в модели (типы агентов, допустимые связи между агентами, наблюдаемая информация).
- *Дескрипторы сценариев*, определяющие реальное содержимое модели: источники данных, визуализации, протоколирование.

Оба набора параметров хранятся в файлах XML. Дескрипторы модели создаются во время её разработки, а дескрипторы сценариев создаются во время исполнения. Интегрированная среда разработки Repast S предостав-

ляет удобные графические инструменты проектирования и редактирования дескрипторов модели.

Базовые инструменты моделирования

Помимо средств общего назначения (упоминавшихся выше) среда разработки Repast S содержит множество специальных инструментов поддержки. Рассмотрим некоторые из них более подробно.

Контексты Базовая структура данных в Repast S называется контекстом. Контекст — это просто контейнер, представляющий собой некоторое подобие математического множества. В контекст может быть помещен объект любого типа с оговоркой, что в контексте может содержаться только один экземпляр любого объекта. С точки зрения моделирования, контекст представляет абстрактную группу. Объекты в контексте являются элементами группы модели. Необходимо отметить, что контекст не предоставляет никакого механизма для взаимодействия между агентами модели. Тем не менее, он обеспечивает базовую инфраструктуру для определения группы и характера взаимодействий в ней. Т. е. контекст формирует многоагентную среду ([разд. 2.2.1](#)).

Каждый контекст обладает своим собственным состоянием. Это состояние может содержать данные различных типов. В состоянии контекста могут содержаться простые значения, например, время, либо более сложные структуры данных: многомерные поля, массивы и т. д.

Контексты могут содержать подконтексты. Т. е. контексты при необходимости могут быть разделены на подкомпоненты в соответствии с принципом холонической декомпозиции ([разд. 2.2.1](#)). Членство в контекстах наследуется, т. е., если объект является элементом подконтекста, то, очевидно, что он является и элементом родительского контекста. Каждый подконтекст находится в

своём собственном состоянии, не зависящим от состояния родительского контекста. По аналогии с МС, агенты могут перемещаться между различными контекстами/подконтекстами.

Проекции Проекциями называются специальные структуры данных, определяющие и обеспечивающие коммуникации между агентами внутри заданного контекста.

Проекции формируют новую структуру в рамках заданной группы контекста. Данная структура определяет и обеспечивает связи в группе агентов, используя семантику, заданную в проекции. С практической точки зрения это означает, что проекции добавляются в контекст для того, чтобы обеспечить взаимодействие агентов внутри платформы. В одном контексте может быть сразу несколько проекций, из чего следует, что агенты могут образовать произвольное количество типов связей. Каждая проекция может взаимодействовать с любыми объектами. Для того чтобы работать с проекциями не требуется изменений кода агента, проекции создаются независимо от агентов. Модель создания проекций является гибкой, поиск связей между объектами или связанных элементов осуществляется специальными поисковыми объектами. Эти объекты специализируются на конкретных типах проекций. Тем не менее, все поисковые объекты реализуют один и тот же интерфейс, который возвращает список объектов, которые удовлетворяют критерию определенного контекста. Поисковые объекты могут объединяться и образовывать новые запросы произвольной сложности. Например, можно создать запрос к контексту, используя различные содержащиеся в нем проекции или атрибуты объектов. Аналогично, создав базовые набор запросов, соединив их с анализатором выражений, разработчик модели, не модифицируя код, может создать систему, в которой множества связей, запрашиваемых агентами, могут быть созданы и реорганизованы при помощи декларативного языка запросов.

Рисунок 19. Контекст, содержащий агентов (слева), проекции (сеть, карту, сетку - справа), подконтекст (внизу) Контекстно-зависимое поведение

Контексты предоставляют способ создания и поддержки среды с популяцией. Проекции позволяют определить широкий спектр связей при помощи объектов Queries. Однако они не учитывают поведениеprotoагентов или то, как protoагенты будут использовать информацию, полученную из контекста и его проекций. Локализованное поведение – это поведение, которое protoагент проявляет только в определенных условиях. Эти условия определяются контекстом, в который помещен protoагент.

Для того чтобы реализовать контекстно-зависимое поведение protoагентов, создаются «часовые» или триггеры. Проектировщик определяет конкретные условия для выполнения определенного поведения. Эти условия учитываются «часовым». Используя «часовых» дизайнер может изменять поведение protoагента в зависимости от контекста и обстоятельств. Название Repast – Recursive Porous Agent Simulation Toolkit, Repast Symphony Разработчик Argonne National Laboratory Сайт <http://repast.sourceforge.net/> Язык Java: J2EE, J2SE, Groovy, Python, семейство языков платформы .NET(C Sharp, VB и др.) Технические/функциональные характеристики

Планировщик событий (выполняет последовательные, параллельные операции с дискретными событиями) Встроенные средства записи результатов(в текстовые файлы, в файлы формата XML) Используются методы Монте-Карло Поддержка двухмерных, трехмерных визуализаций Динамическое управление агентами во время выполнения Библиотеки генетических алгоритмов, нейронных сетей, генерации случайных чисел, спец. мат. выражений Встроенные средства моделирования системной динамики Инструменты для моделирования социальных сетей Поддержка геоинформационных систем Операционные системы: Windows, Mac OS, Linux Работа как на персональных компьютерах, так и на научных вычислительных кластерах Моделирование

при помощи блок-схем Поддержка Java, Groovy Интеграция с множеством различных внешних утилит, включая R statistics environment, подключаемые модули анализа трафика *ORA и Pajek, плагин для работы с SQL-запросами, инженерный пакет визуализации VisAD, платформа интеллектуального анализа данных Weka, многие популярные форматы электронных таблиц, вычислительная математическая среда MATLAB, дизайнер визуальных отчетов iReport; интеграция с библиотекой сетевого моделирования JUNG импорт данных из электронных таблиц Microsoft Excel, файлов UCINET DL Опциональное сквозное моделирование на базе XML Распределенные вычисления с помощью продукта Terracotta

Сетевое окружение – Терминалы Все терминалы, поддерживающие среду J2SE, Groovy либо содержащие интерпретатор языка Python, либо установленную платформу .NET

Таблица 3. Основные технические характеристики инструментария Repast, Repast S Средство мультиагентного моделирования MASON

3.5.3. SPADE

SPADE (Smart Python multi-Agent Development Environment²⁶) – представляет собой прикладную инструментальную библиотеку для проектирования МАС на базе *Расширяемого протокола обмена сообщениями и информацией о присутствии* (Extensible Messaging and Presence Protocol – XMPP). Библиотека полностью реализована на языке Python, что делает её очень гибкой и легко настраиваемой для любой проектируемой МАС, в т. ч. технологического назначения.

К основным особенностям МП SPADE можно отнести:

- Полное соответствие стандартам проектирования FIPA.

²⁶ англ. *Среда разработки интеллектуальных многоагентных систем на языке Python.*

- Реализация четырёх протоколов межагентного взаимодействия: XMPP, P2P, HTTP и SIMBA.
- Поддержка двух языков описания содержимого: FIPA-ACL и RDF.²⁷
- Агенты SPADE достигают своих целей посредством инициирования делиберативных или реактивных процессов.
- SPADE предоставляет удобный web-интерфейс для управления платформой.
- Платформа SPADE имеет все возможности для интеграции с другими многоагентными платформами, созданными на любых языках программирования и работающие под управлением любых ОС.
- Наличие подсистемы уведомлений позволяет системе определять текущее состояние всех агентов в реальном времени.
- Платформа SPADE позволяет объединять агентов в т. н. *организации* для решения непересекающихся задач, что даёт возможность создавать на её основе сложные агент-холонические системы управления ТПП (разд. 2.3.2).

Архитектурная модель

Архитектурная модель МП SPADE представлена на рис. 3.14. Не трудно заметить, что эта архитектура не требует, чтобы все агенты МАС создавались с помощью библиотеки SPADE. Платформа лишь представляет инструменты взаимодействия агентов внутри МАС и даёт возможность интеграции с другими платформами, соответствующими стандартам FIPA. Единственное требование: наличие специализированного адаптера, дающего возможность

²⁷ англ. *Resource Description Framework* — это разработанная Консорциумом всемирной паутины (World Wide Web Consortium — W3C) модель для представления данных, в особенности — метаданных. RDF представляет утверждения о ресурсах в виде, пригодном для машинной обработки [118].

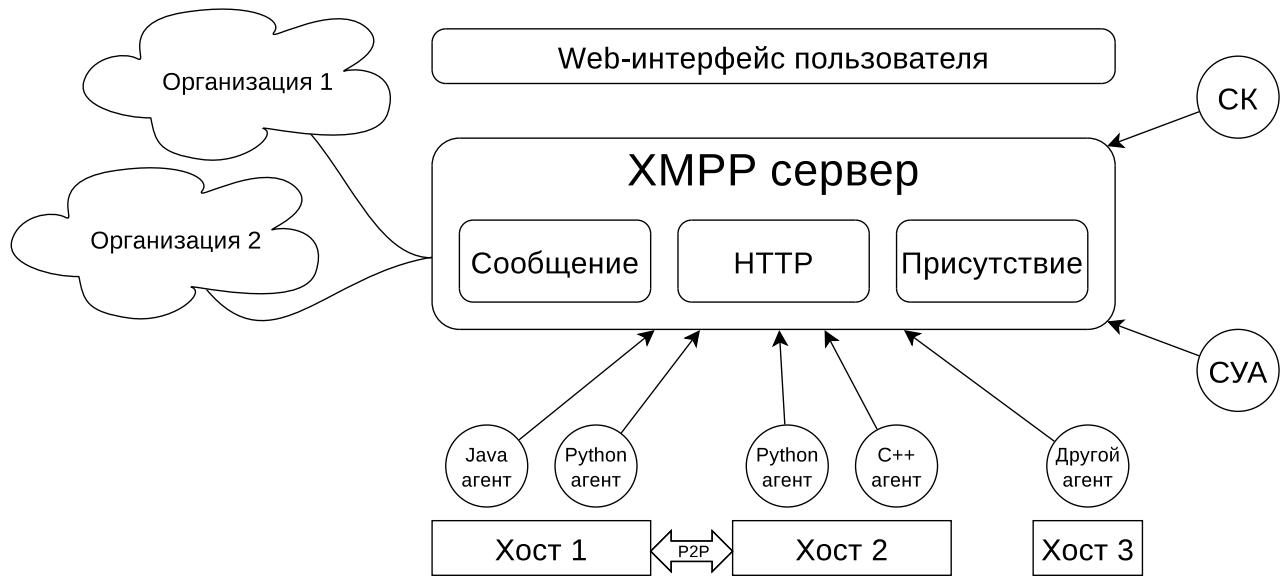


Рис. 3.14. Архитектура платформы SPADE

сторонним агентам работать с сообщениями инкапсулированными по протоколу XMPP.

Как уже было отмечено выше, всё взаимодействие агентов на уровне платформы осуществляется по протоколу XMPP. Изначально этот протокол использовался для организации сервисов передачи коротких сообщений. На сегодняшний день он находит все более широкое применение в любых системах, где требуется оперативный обмен информацией в режиме, близкому к режиму реального времени.

3.5.4. Сравнительный анализ рассмотренных средств проектирования МП

Резюмируя всё высказанное, следует отметить, что все исследованные МП подходят для разработки многоагентной ИУП ТПП. Однако, каждое из рассмотренных инструментальных средств имеет свою специфику, см. табл. 3.4.

Платформа JADE на сегодняшний день является самой комплексной и проработанной системой построения МАС. Она идеально подходит для

Таблица 3.4
Основные характеристики исследуемых многоагентных платформ

Хар-ка \ МП	JADE	REPAST	SPADE
Поддерживаемые языки	Java (J2EE, J2SE, J2ME)	Java (J2EE, J2SE, Groovy), Python, .Net (C#, VB и др.)	Python
Совместимость с FIPA	Полная	Полная	Полная
Графический интерфейс	Отсутствует	Интегрированная среда	Web-интерфейс
Область применения	Крупные телекоммуникационные MAC	Быстрое прототипирование MAC	Создание гибких, легкоинтегрируемых MAC

разработки крупных технологических систем на предприятиях со сложной организационной структурой, имеющей отраслевые подразделения и региональные филиалы. Данная система может стать единой телекоммуникационной платформой не только для интеграции АСТПП, но и других информационных и управляющих систем предприятия. Подобная функциональность явно избыточна для решения задач, являющихся предметом представленного исследования, ведь одна из его целей — создание максимально простой, быстровнедряемой многоагентной и легкорасширяемой технологической системы, ориентированной в первую очередь на специалистов, не обладающих глубокими специальными знаниями в области программирования комплексных распределённых систем.

Удобный графический интерфейс и большое количество поддерживаемых языков разработки МП Repast, безусловно, является её огромным

плюсом. Но, с другой стороны, данная система больше направлена на моделирование и планирование MAC, а не на промышленное использование, что ставит её в один ряд с такими популярными системами моделирования как Adonis или Rational Rose.

МП SPADE сочетает в себе лучшие черты двух предыдущих систем. С одной стороны, это гибкая и расширяемая система, написанная на языке Python (этот язык по праву считается наиболее лёгким для использования непрофессиональными программистами), а с другой — в ней реализованы все основные функции, присущие промышленной МП. По мнению автора именно платформа SPADE является наиболее предпочтительной для разработки многоагентной ИУП ТПП.

3.6. Выводы к третьей главе

В данной главе был сделан краткий обзор инструментальных средств построения многоагентных систем.

Глава 4

Реализация ТПП изделий из полимерных композиционных материалов на основе многоагентной системы

Для отработки методов организации многоагентной технологической системы автором рассмотрена задача технологической подготовки производства изделий из полимерных композиционных материалов.

Композиционные материалы (от лат. *composition* — составление) представляют собой многокомпонентные материалы, состоящие из полимерной, металлической, углеродной, керамической или другой основы, которая называется матрицей, армированной наполнителями из волокон, нитевидных кристаллов, тонкодисперсионных частиц и др. Путём подбора состава и свойств наполнителя и матрицы, их соотношения, ориентации наполнителя можно получить новые материалы с требуемым сочетанием эксплуатационных и технологических свойств. Именно сочетание разнородных веществ приводит к созданию нового материала, свойства которого количественно и качественно отличаются от свойств каждого из его составляющих.

Технологическая подготовка производства изделий из ПКМ сводится к проектированию технологии изготовления конструкции изделия и проектированию материала, удовлетворяющего требованиям, предъявляемым к данной конструкции, т. е. создание материала и изделия совмещаются, при этом сразу получается готовое изделие заданной формы. Подобное разделение существенно увеличивает количество инженерных изысканий, проводимых в процессе ТПП, что в конечном итоге, приводит к необходимости

мости задействования большего количества средств автоматизации ТПП, а также их интеграции.

По мнению автора, основная сложность проектирования технологии изготовления изделий из ПКМ заключается в параллельности и многоитерационности данного процесса. После получения исходных данных для проектирования в виде модели будущего изделия и набора его характеристик (требований), на первом этапе необходимо сформировать проект, используя для этого систему управления данными об изделии (PDM). Систематизированные в PDM-системе данные передаются для анализа специалистам по проектированию материалов (здесь и далее см. рис. 4.1). На этом этапе должны быть задействованы специализированные экспертные модули, позволяющие из простого набора параметров изделия составить примерную модель будущего материала, представленную в виде фреймов. На основании имеющихся характеристик происходит первичный отбор компонентов ПКМ (либо полуфабрикатов: препротов, премиксов и т. д.) из сторонней БД полимерных материалов, которые также заносятся в PDM-систему. При этом проект из фазы начального проектирования переходит в фазу проектирования материала (в соответствии с используемым на предприятии регламентом).

Полученные данные об элементах композиции передаются в специализированную систему моделирования материала, где создаётся математическая модель композиции. Далее модель передаётся системе (либо САЕ-модулю/решателю) конечноэлементного анализа, где определяется ориентация волокон и строится матрица жесткости. Данные параметры помещаются в базу PDM системы, при этом проект переводиться в состояние «*Проверка полученного материала*». После этого геометрическая модель разрабатываемого изделия и характеристики материала (в виде данных) направляются в систему компьютерного моделирования для проверки соответствия разработанного материала (и изделия) требованиям, полученным на первом этапе. В случае

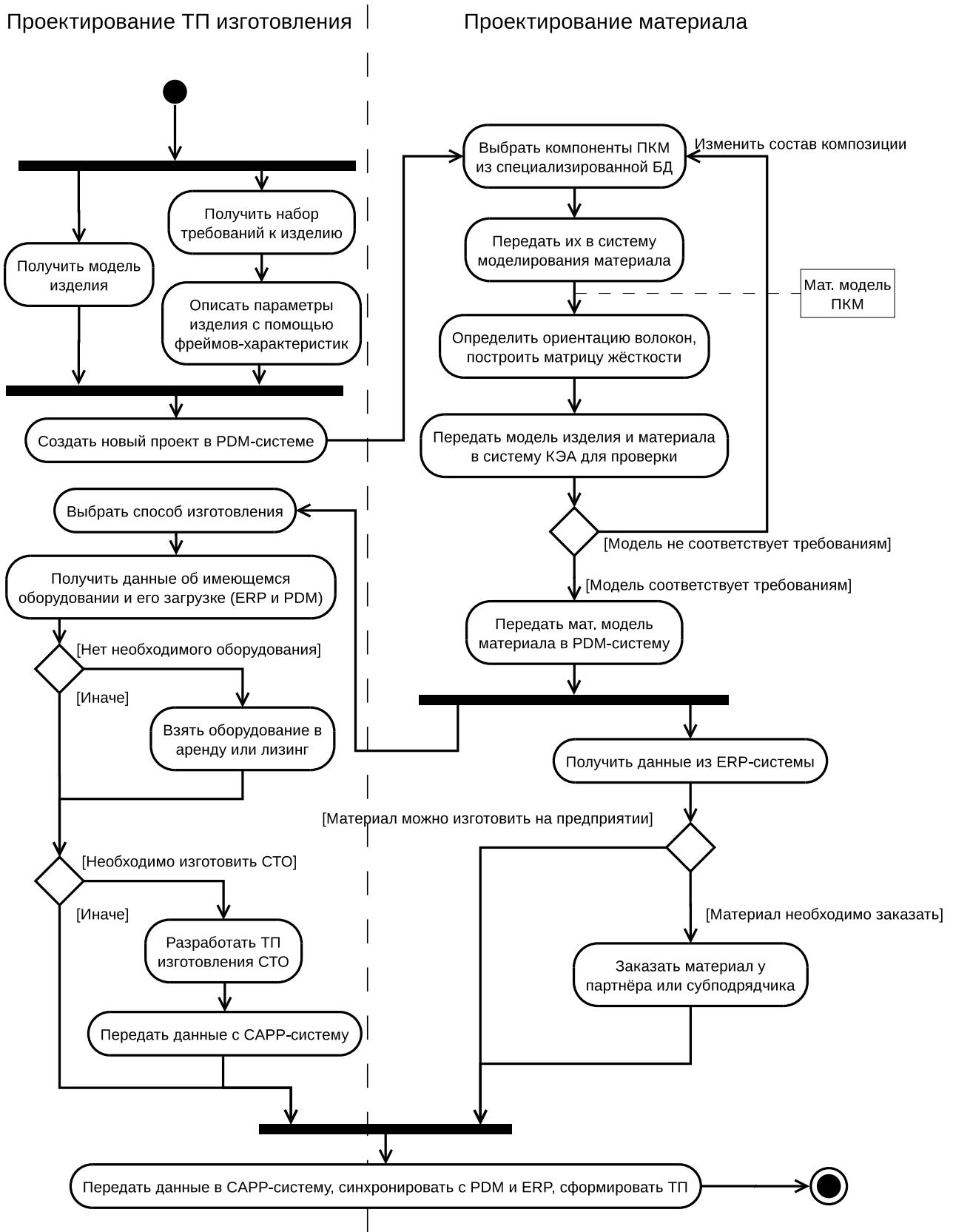


Рис. 4.1. UML-диаграмма деяельности ТПП изделий из ПКМ

неудачи создаётся новая версия проекта, после чего уточняется состав композиции и проводится повторная проверка.¹

Результат этих действий (данные о составе полимерной композиции и её свойствах) сохраняется в PDM системе,² после чего проект переводится в фазу «Проектирование ТП изготовления».

На этом этапе происходит разделение ТПП. С одной стороны, начинается работа по созданию технологии производства материала, с другой — технологии изготовления продукции. Первая задача требует постоянного обмена данными с системой управления ресурсами предприятия (ERP). Ведь компоненты разработанного ПКМ могут быть как на складах предприятия, так и на складах предприятий-партнёров, либо должны быть заказаны (все, либо только часть). Также должен быть учтён вариант обращения к субподрядчикам, у которых можно заказать готовую полимерную композицию в соответствии с разработанным составом.

Проектирования ТП изготовления начинается с анализа элементов разработанной композиции и выбора технологии производства. Это может быть прессование, экструзия, пултрузия, вакуумное формование, «мокрая» или «сухая» намотка, автоклавное формование и т. д. На этом этапе также требуется постоянный обмен информацией между PDM- и ERP-системами с целью выбора оборудования, либо обращения к партнёрам (субподрядчикам).

Следующим этап — проектирование средств технологического оснащения (СТО). Здесь необходимо учитывать информационные потоки, возникающие между PDM-системой, системой (системами) проектирования управляющих программ и системой проектирования геометрии оснастки (CAD).

¹ Если результат стал ещё хуже, происходит возврат к предыдущей версии.

² В дополнение к этому он может быть перенесён в специальную базу материалов-аналогов, в которой хранятся данный о составе вновь полученного материала, его характеристики и область применения.

На последнем этапе технологический процесс окончательно формализуется с использованием системы автоматизированного проектирования технологических процессов (САРР): определяется последовательность операций, оформляется комплект документации, осуществляется составления плана производства, нормирование (что опять же требует работы с ERP-системой) и т. д.

В следующих разделах автором будет описана процедура одноранговой многоагентной интеграции технологических данных и знаний, возникающих на всех вышеописанных этапах ТПП изделий из ПКМ.

4.1. Описание интегрируемых систем автоматизации технологической подготовки производства

В первую очередь необходимо описать упрощённую схему информационных потоков ТПП изделий из ПКМ (рис. 4.2). Из представленной схемы видно, что ТПП изделий из ПКМ можно условно разделить на два больших подэтапа: *проектирование материала и разработка ТП изготовления* конечного изделия. Отсюда следует, что для построения минимальной модели ИУП ТПП необходимо в рамках единого информационного пространства как минимум обеспечить интеллектуальное взаимодействие:

- Программных систем, предназначенных для решения различных инженерных задач: расчётов, анализа и симуляции физических процессов (САЕ). Для решения задач проектирования ТПП изделий из ПКМ были задействованы следующие системы: **Moldex3d** (система моделирования процессов заливки материала, выдержки под давлением, охлаждения, усадки, коробления и т. д.), **DIGIMAT** (платформа для полномасштабного конечно-элементного моделирования нелинейного поведения ПКМ

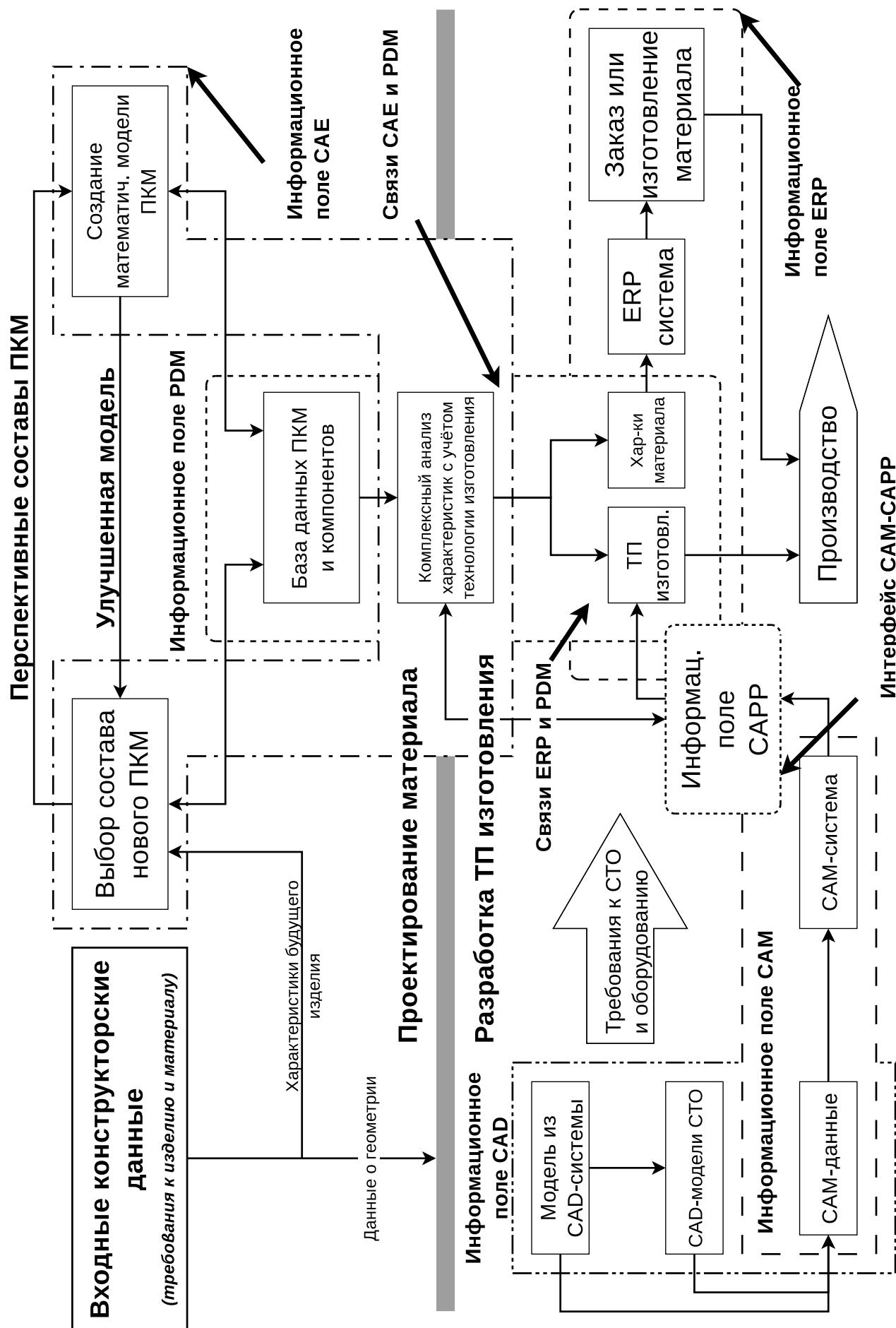


Рис. 4.2. Схема информационных потоков ТПП изделий из ПКМ

и композитных структур), *Samcef* (программное обеспечение для расчётов методом конечных элементов), а также набор инструментальных средств для общеинженерных расчётов *SALOME*.

- Организационно-технической системы, обеспечивающей управление всей информацией об изделии (PDM). Для интеграции была выбрана система *ENOVIA SmarTeam*, используемая для управления жизненным циклом проектируемых изделий, а также в качестве интеллектуальной базы проектируемых ПКМ.
- Интегрированной системы управления внутренними и внешними ресурсами приборостроительного предприятия (ERP). Была использована система *OpenERP*.
- Системы автоматизированной технологической подготовки оборудования с числовым программным управлением (САМ). Использовалась базовая по своей функциональности система *RyCAM*, позволяющая создавать управляющие программы для трехкоординатных фрезерных станков, чего оказалось достаточно для отработки методов интеграции подобных систем.
- Системы автоматизированной подготовки (написания) технологических процессов (САПР). Была использована система *Вертикаль 2011*.
- Вспомогательных баз данных хранящих различную справочную информацию и не являющихся частью ни одной из рассмотренных выше систем. Была использована система управления базами данных *PostgreSQL*, а также информационные адаптеры для web-ориентированных баз по материалам, таких как: *M-Base*, *CAMPUS*, *MatWeb*.

Более подробное описание возможностей указанных автоматизированных систем технологической подготовки производства представлено в [прил. Г](#). В процессе моделирования на РВИС для каждой системы автоматизации

ТПП, участвующей в интеграционной сети, создана виртуальная машина, для каждого специалиста — виртуальное рабочее место. Организационная модель интеграционной сети представлена на рис. 4.3.

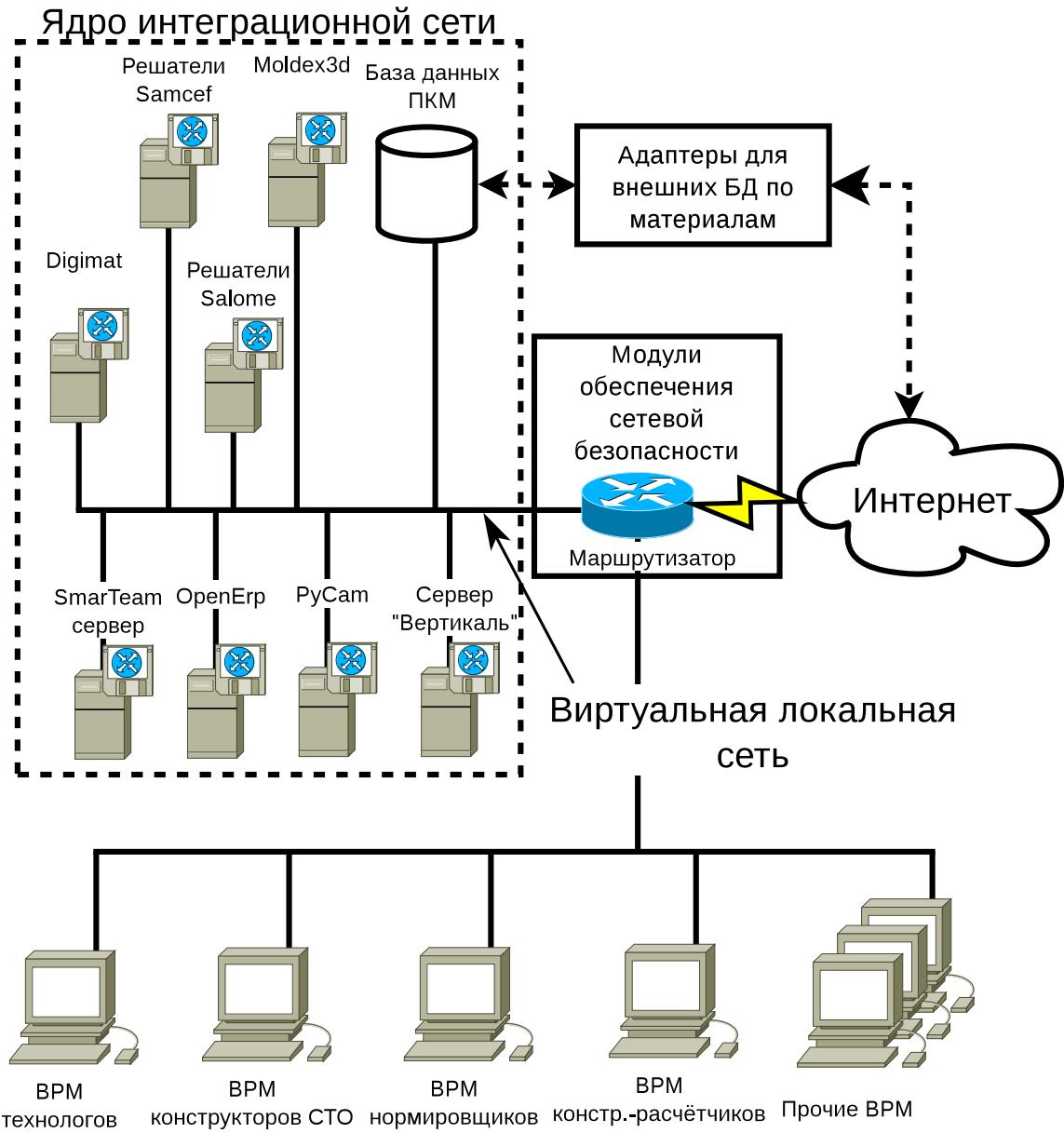


Рис. 4.3. Организационная модель интеграционной сети ИУП ТПП

Все виртуальные машины объединены в локальную сеть с доступом в Интернет через общий шлюз (на рисунке обозначен как *маршрутизатор*). Безопасность и конфиденциальность данных достигается за счёт использования специализированных модулей шифрования, обеспечивающих требуемый уровень защищённости коммуникационных каналов.

Как уже отмечалось в разд. 3.2 доступ к виртуальным рабочим местам осуществляется через web-интерфейс, т. е. не требует создания какого-либо дополнительного клиентского программного обеспечения. Пример ВРМ, позволяющего работать с САЕ-системой Samcef, представлен на рис. 4.4.

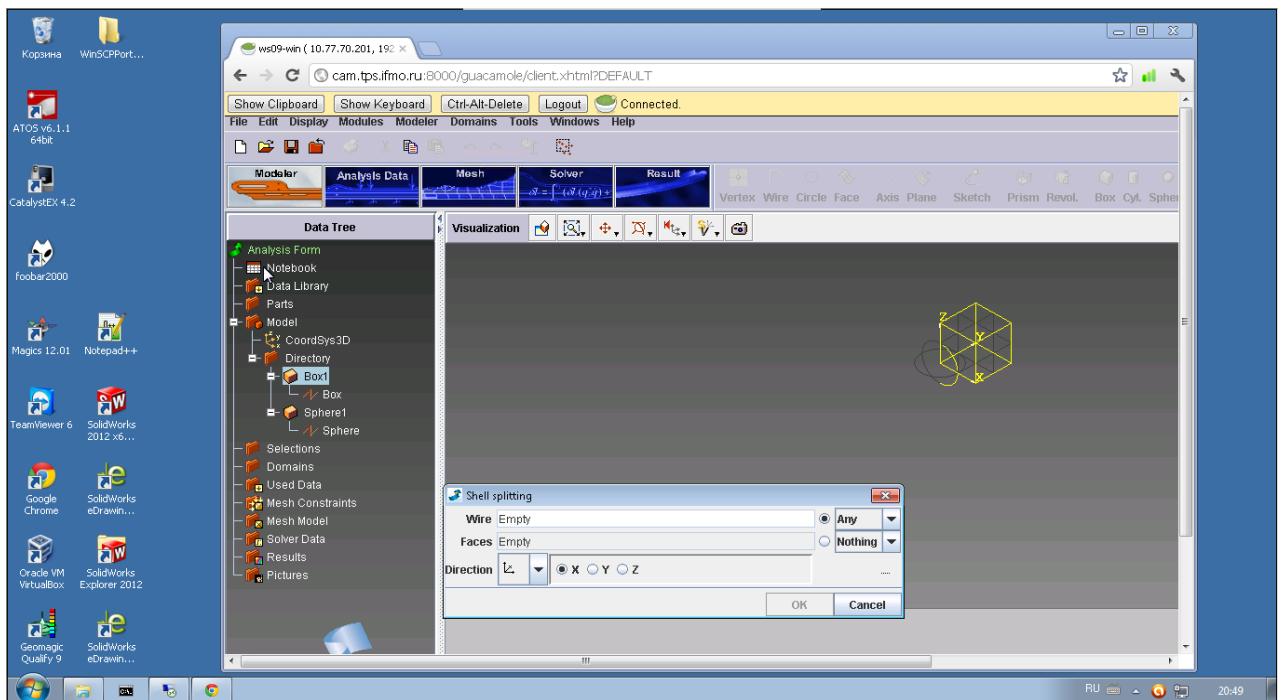


Рис. 4.4. Web-интерфейс виртуального рабочего места

4.2. Описание агентов, моделей поведения и целей

Каждый агент созданной МАС представляет собой независимый исполняемый модуль (скрипт), базовые методы которого наследуются³ от агента-прототипа (разд. 2.2.2). Каждый класс агентов обладает своим собственным поведением (конфигурацией, позволяющей ему работать с определённым классом автоматизированных систем технологической подготовки производства или специалистом, иначе говоря — *контекстом*, см. разд. 3.1.2),

³ Следует отметить, что в языке Python нет понятия класса в привычном его понимании. Классы, по сути, тоже являются объектами, а «настоящие» классы скрыты от программиста. Поэтому методы, действительно, могут быть унаследованы от объекта-прототипа [132–134].

обусловленным некоторым планом и определёнными целями. Пример реализации простейшего агента приведён в прил. Д.

Для сохранения максимальной простоты и гибкости в проектируемой системе автором определены всего два базовых класса агентов:

1. Класс А — *Агенты-преобразователи*.
2. Класс Б — *Агенты-интерфейсы*.

Агенты первого класса связаны с одной из информационных систем, использующихся в процессе технологической подготовки производства, агенты второго класса должны взаимодействовать с пользователями. Также определены два сервисных агента: *агент системы управления* и *агент службы каталога* (рис. 4.5). К основным функциям агента системы управления относятся: хранение транспортных адресов агентов и маршрутизация внутри агентной среды. Основная функция агента службы каталога — хранение актуального списка общесистемных и пользовательских сервисов (абстрактных ресурсов), предоставляемых агентами. Общесистемные сервисы позволяют агентам взаимодействовать, получая друг у друга данные и знания. Пользовательские сервисы предоставляют оператору некоторый формализованный диалог, в процессе работы с которым могут быть получены новые технологические данные или знания. Примерами таких сервисов могут служить *расчёт режимов резания*, *подбор материала по параметрам*, *расчёт усадок* и т. д. Каждый сервис описывается транспортным адресом агента и триплексной строкой параметров, которые может обработать данный сервис.

4.2.1. Агент-преобразователь

Агент-преобразователь отвечает за создание, управление и поддержание в надлежащем состоянии онтологии, относящейся к какой-либо конкретной подобласти информационного поля технологической подготовки

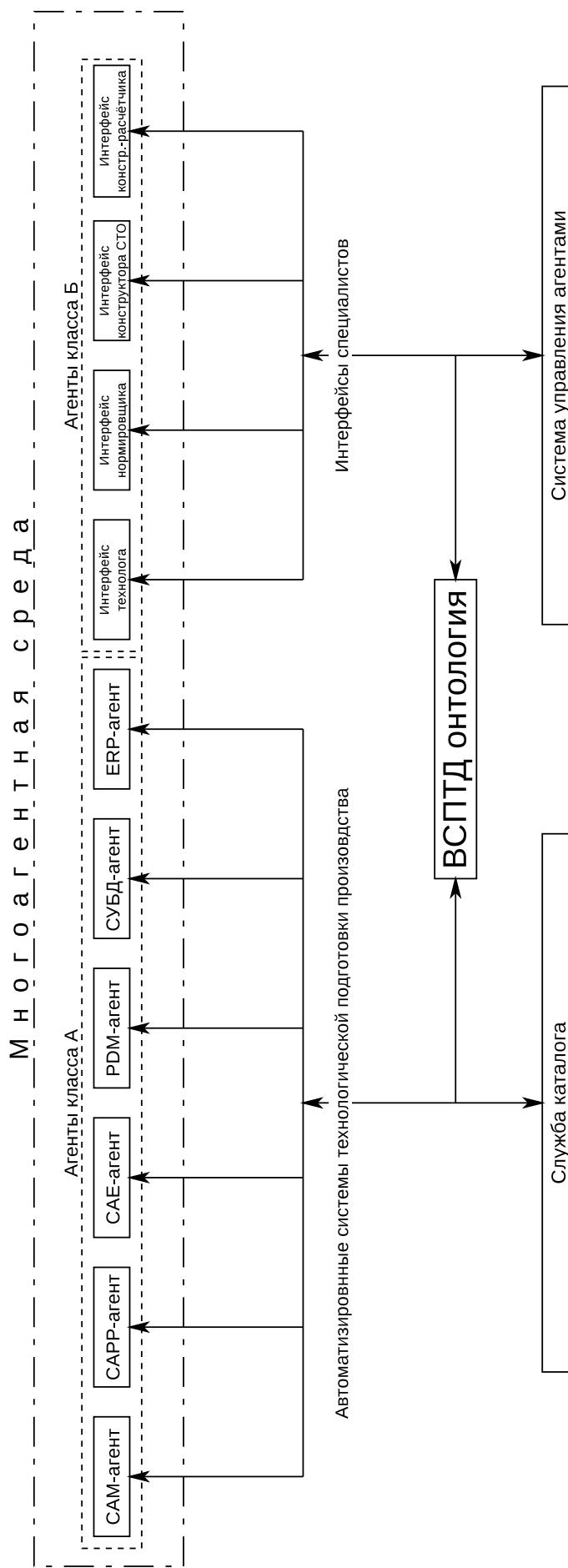


Рис. 4.5. Схема взаимодействия агентов ИУП ТПП

производства. Агент-преобразователь связан с одной или несколькими информационными сущностями (в рассматриваемой упрощённой схеме — с системами технологической подготовки производства изделий из ПКМ). Агент-преобразователь имеет интерфейс для работы с инженером по знаниям, отвечающим за наполнение внутренней онтологии агента и подключение к агенту различных информационных модулей, реализующих сервисы агента. Сконфигурированный для работы с конкретной системой автоматизации, агент-преобразователь ожидает запросы от других агентов.

Вследствие высокой сложности рассматриваемой предметной области, в онтологии её описания не обойтись без наследования одних понятий другими, иными словами, многие информационные сущности технологической подготовки производства должны образовывать строгие иерархии. Как уже отмечалось в [разд. 2.3](#), для образования подобных структур внутри гетерархической сети, агенты-преобразователи могут образовывать постоянные или временные холархии внутри домена кооперации, при этом агент (в данном случае холон), отвечающий за базовые определения конкретного класса понятий, становится координатором.

Например, в процессе подбора параметров полимерных композиций происходит активный обмен информацией между несколькими САЕ-системами, при этом на каждом шаге этого многоитерационного процесса соответствие параметров изделия заданным характеристикам проверяется в системе конечноэлементного анализа Samcef. Агент данной системы становится временным координатором, управляющим данным процессом и принимающим решение об изменении состава полимерного композиционного материала (т. е. о возврате на один шаг назад) или переходе на следующий этап. Соответственно, когда оптимальный состав найден, данный временный кооперационный кластер агентов распадается.

4.2.2. Агент-интерфейс

Агент-интерфейс с одной стороны взаимодействует со специалистом, а с другой — с агентами-преобразователями. Таким образом, агент-интерфейс помогает пользователю напрямую работать с незнакомыми ему системами, например, технолог может напрямую обращаться к ERP-системе, получая при этом упрощённый интерфейс, в котором будет отражена только та информация, которая соответствует онтологии понятий технологического процесса, а все остальные будут либо переведены в понятную для технолога форму, либо сконвертированы в соответствии с правилами перевода онтологии, либо опущены за ненадобностью. С технической точки зрения, агент-интерфейс взаимодействует с пользователем через web-интерфейс, т. е. является «насыщенным» интернет-приложением (Rich Internet Application — RIA⁴), с которым пользователь работает через интернет-браузер (просмотрщик).

Работа с агентом-интерфейсом может осуществляться в трёх режимах:

1. В режиме *свободного поиска*, когда агент-преобразователь проводит полнотекстовый поиск в онтологическом словаре, например по запросу «материал липол плотность» система вернёт значение найденного параметра, а также предложит пользователю просмотреть дополнительные результаты, найденные в онтологии: другие параметры материала, входимость этого материала в состав различных ПКМ, изделия, созданные из этого материала, оборудование и т. д. При этом поиск будет осуществляться во всём информационном пространстве, что достигается наличием единого метахранилища всех технологических данных и знаний, т. е. онтологического словаря.

⁴ Приложение (компьютерная программа), доступное через сеть интернет и обладающее функциональностью традиционных настольных приложений [135, 136].

2. В режиме *поиска сервиса*, в котором по требованию пользователя система вернёт либо список всех доступных агентам ИУП ТПП сервисов, разбитых по категориям, либо предложит воспользоваться свободным поиском сервиса, аналогично предыдущему варианту.
3. В режиме *работы с сервисом*, в котором агент-интерфейс динамически формирует кумулятивный пользовательский диалог (автоматически сгенерированную web-форму) на основании фрейма-терминала ВСПТД. Отличительной особенностью данного диалога является то, что он даёт возможность пользователю работать с данными и знаниями сразу от нескольких систем. Пример интерфейса, сформированного агентом при работе с сервисом PDM-системы, представлен на рис. 4.6.

The screenshot shows a Google Chrome window titled "Form - Google Chrome" with the URL "form.com/form/20532528264". The form is for selecting a material. It includes a field for "Идентификатор *" with value "MT-000027". Below it is a section titled "Параметры материала" containing a table with four rows: "Плотность", "Температура по Вика", "Электропроводность", and "Твёрдость". The table has columns for "Очень важно", "Важно", "Желательно", and "Не важно". The "Твёрдость" row has radio buttons selected in the "Очень важно" and "Важно" columns. Below the table are dropdown menus for "Марка" (Derlin 100) and "Производитель" (Dupont). At the bottom left is a button for "Дополнительные параметры" which opens a large empty text area. At the bottom right are "OK" and "Clear Form" buttons.

Рис. 4.6. Пример интерфейса при работе с сервисом PDM-системы

4.3. Описание взаимодействия агентов ИУП ТПП

Взаимодействие агентов ИУП ТПП на уровне представления осуществляется путём обмена сообщениями на языке FIPA-ACL. Содержимое каждого сообщения представляет собой триплексную строку, содержащую передаваемые данные или знания. Онтология передаваемого сообщения описывается с помощью словаря метаданных ВСПТД ([разд. 3.1.3](#)).

Рассмотрим типовой сценарий взаимодействия агентов. После установки вычислительного модуля интеллектуального агента на целевую платформу (систему, обеспечивающую функционирование того или иного средства автоматизации ТПП) происходит его инициализация. В процессе инициализации агент должен «осознать» себя в среде информационно-управляющей платформы технологической подготовки производства. Первым делом, ИУ проверяет наличие у него доступа к сети (в данном случае имеется в виду вычислительная сеть предприятия, поверх которой строится многоагентная сеть). Для этого агенту необходимо послать тестовый запрос (функция `ping()`, здесь и далее см. [рис. 4.7](#)) к службе управления агентами ([разд. 4.2](#)). Тестовое сообщение представляется специализированным триплетом виртуального строкового пространства вида:

Листинг 4.1. Пример тестового запроса к СУА

```
1 $__SERVICE . TEST = :0 ;
```

В данном случае, СУА был передан триплет цели, помеченный специализированным системным префиксом `__SERVICE`.⁵ Параметру `TEST` присваивается значение 0. Затем триплет помещается в контейнер и передаётся СУА по протоколу XMPP ([разд. 3.5.3](#)).

⁵ Системные префиксы ВСПТД начинаются с символа «двойное подчёркивание», они также хранятся в онтологическом словаре метаданных, но по умолчанию скрыты.

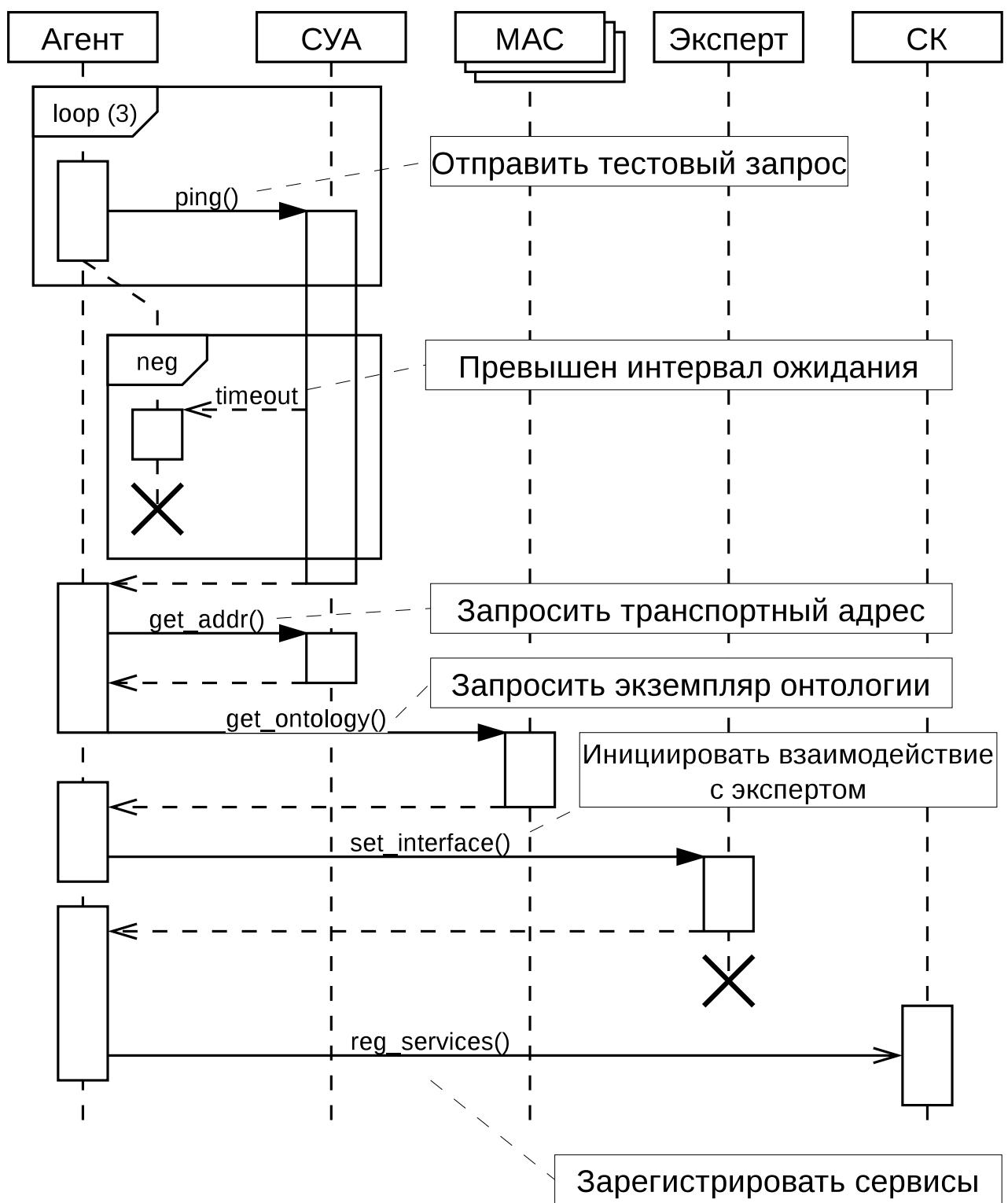


Рис. 4.7. UML-диаграмма последовательности действий при инициализации агента

Если сеть доступна, и в конфигурации агента был задан правильный транспортный адрес СУА, данный триплет цели будет обработан, и агенту вернётся ответ, содержащий уже триплет факта вида `$_SERVICE.TEST=1;`. Если СУА в данный момент не может обработать запрос от агента, значение триплета изменено не будет. В этом случае агент должен повторить свой запрос через некоторое время. Если после трех попыток ответ не получен, агент сообщает об ошибке и прекращает свою работу.

На следующем этапе агент должен отправить СУА запрос на получение транспортного адреса (функция `get_addr()`):

Листинг 4.2. Пример запроса на получение транспортного адреса

```
1 $_SERVICE.TRANSPORT_ADDRESS=:
2 "00000000000000000000000000000000";
```

Значением триплета цели является пустой MD5-хэш,⁶ который обрабатывается СУА и преобразуется в триплет факта:

Листинг 4.3. Транспортный адрес агента

```
1 $_SERVICE.TRANSPORT_ADDRESS=
2 "1bc29b36f623ba82aaf6724fd3b16718@tps.ifmo.ru";
```

Символы до знака «@» являются непосредственно адресом агента, а символы после — доменным именем сервера, на котором работает СУА. Получив транспортный адрес, агент посыпает третье (после тестового и запроса адреса) обязательное сервисное сообщение: запрос онтологического словаря. Данное сообщение также содержит сервисный триплет цели, но, в отличие от предыдущих, данное сообщение рассыпается всем агентам МАС, адреса которых известны СУА (функция `get_ontology()`). Далее агент переходит

⁶ Под хэшированием понимается преобразование входного массива данных произвольной длины в выходную битовую строку фиксированной длины, что позволяет создавать уникальный идентификатор агента без использования алгоритмов генерации псевдослучайных чисел.

в фазу ожидания, ему необходимо дождаться пока все агенты пришлют ему свою копию онтологического словаря. Это необходимо для того, чтобы агент на момент инициализации имел наиболее полную версию словаря.

В процессе работы МАС, онтология может меняться, для поддержания её в актуальном состоянии агенты время от времени синхронизируются, и в этот момент целостность онтологии нарушается. Именно поэтому инициализируемый агент дожидается, пока *каждый* из агентов МАС пришлёт ему свою копию. Также стоит отметить, что на момент получения запроса онтологии все агенты приостанавливают синхронизацию.

Получение копии онтологического словаря — последний этап инициализации агента, после которого он переходит в режим взаимодействия с экспертом (функция `set_interface()`). На этом этапе эксперту необходимо определить какие модули (подмодули)⁷ автоматизированной системы технологической подготовки производства должны быть подключены к внутреннему интерфейсу агента, определить, необходимо ли внести исправления (дополнения) в онтологический словарь, а также определить какие сервисы будет предоставлять агент. Последнее очень важно, ведь именно на переговорах и соглашениях о предоставлении тех или иных ресурсов (в данном случае программных сервисов) и строится взаимодействие агентов внутри МАС, и именно от них зависит полезность каждого агента ([разд. 2.2.3](#)). Конкретные примеры агентных сервисов зависят от предметной области и степени автоматизации решаемых задач. Например, для рассматриваемой проблемы проектирования технологии производства изделий из полимерных композиционных материалов, могут быть определены следующие сервисы: «сделать

⁷ Как уже отмечалось выше, построить многоагентную систему технологической подготовки производства, не создавая при этом специализированных модулей-адаптеров, невозможно. Задачей автора было максимально упростить этот процесс, сделав его доступным для рядовых сотрудников предприятия, и тем самым, отказаться от необходимости привлекать профессиональных программистов, см. [разд. 3.1.2](#).

комплексную выборку из нескольких БД на основании фрейма-запроса», «проанализировать полученную в результате проектирования композицию», «дать количественную оценку соответствия материала заданным требованиям», «сформировать отчёт по текущей загрузке оборудования» и т. д. Полный перечень сервисов может быть довольно обширным, поэтому детальное их рассмотрение выходит за рамки проводимого исследования.

Сформированный перечень сервисов передаётся службе каталога (СК) агентной системы, где происходит их регистрация и привязка к тому агенту, который о них заявил (функция `reg_services()`). Агент, которому необходимо получить некоторый сервис должен обратиться к СК и посмотреть зарегистрирован ли хотя бы один сервис, который может удовлетворить его запрос (возможно, только частично⁸). Найдя подходящий сервис, агент получает его транспортный адрес, после чего начинается этап переговоров.

В процессе переговоров агенты обмениваются речевыми актами, представленными с помощью языка FIPA-ACL, инкапсулированными в сообщения XMPP (в соответствии с требованиями агентной библиотеки SPADE). В качестве языка описания содержимого используется триплексная строка виртуального строкового пространства технологических данных.

Рассмотрим пример, в котором агент-преобразователь ERP-системы обращается к агенту-преобразователю PDM-системы, желая получить определённые параметры композиционного материала для нормирования. С целью упрощения данного примера будет описана процедура получения всего одного параметра *Плотность*. Следует отметить, что, несмотря на кажущуюся простоту данной операции, её реализация напрямую и штатными

⁸ Процесс выбора сервиса сводится к сопоставлению триплетов цели, которые агент желает перевести в триплеты фактов, с триплексной строкой, описывающей предоставляемый сервис. Соответственно, чем больше фактов будет получено, тем выше будет функция полезности агента, предоставляющего сервис.

методами невозможна. Использованная в процессе технологической подготовки производства изделий из ПКМ PDM-система (SmarTeam, см. прил. Г.2) использует объектную базу данных, обращение к которой возможно только через внутренний COM API системы, в то время как ERP-система (OpenERP, см. прил. Г.3) хранит свои записи в реляционной базе PostgreSQL (прил. Г.6), причём из соображений безопасности не позволяет обращаться к БД напрямую, а использует высокоуровневый API на языке Python. Следовательно, такая, казалось бы, простая операция как получение одного параметра *Плотность* требует создания модуля, который, как минимум, обращался бы к API системы OpenERP, вызывал функцию выборки и передавал ей заранее подготовленный SQL-запрос, затем получал результат этого запроса, преобразовывал его в соответствии со своей внутренней логикой, вызывал API системы SmarTeam, формировал бы запросы к объектной БД, выполнял его, получал результат и делал его обратное преобразование для размещения в БД OpenERP. Работа с подобным «чёрным ящиком» может вызвать ряд проблем:

1. Понятие *Плотность* в разных системах может обозначаться по-разному (например, в ERP-системе может быть поле, называемое DENSITY, а PDM — объект, именуемый «*Плотность материала*»).
2. Описываемый модуль может связать только две системы, установленные на одной машине, в противном случае, ему придётся использовать сетевые вызовы COM, что необоснованно усложнит его внутреннюю реализацию.
3. Любые, даже незначительные, изменения в структуре данных или API описываемых систем повлекут за собой необходимость изменения кода модуля и его перекомпиляции.

Предлагаемый автором подход позволяет существенно упростить подобную интеллектуальную интеграцию. Во-первых, агенты МАС не оперируют с понятиями, обозначающими данные напрямую, все они хранятся в он-

тологическом словаре в виде концептов, относящимися к определённому классу и обладающих некоторыми атрибутами, связями и т. д. Так, в рассматриваемом примере, агент ERP-системы обратится к онтологии понятий, относящихся к полимерным композиционным материалам для получения концепта по его синониму:⁹

Листинг 4.4. Получение концепта по его синониму

```
1 erp.set_ontology('ПКМ')
2 erp.attrs += mas.get_by_syn('DENSITY')
```

В результате агент получает триплексную строку с выборкой по синониму:

Листинг 4.5. Представление онтологического синонима

```
1 $PCM.PLOTN=" _S_ ,$0A=Плотность' ;$1A=' DENSITY' ;
2                 $2A=' Плотность материала' ;"
```

В данной записи 0A — ссылка на агент-прототип, хранящий базовое определение онтологического концепта ([разд. 4.2.1](#)), в списке синонимов значение от агента 0A является главным для внутренней работы ВСПТД; 1A — идентификатор объекта агента ERP; 2A — идентификатор объекта агента PDM; _S_ — системная строка, указывающая на тип «значения-синонимы».¹⁰

Получив обозначение концепта, ERP-агент формирует триплет цели и помещает его в виртуальное строковое пространство технологических данных: \$PCM.PLOTN=:;. Таким же образом могут быть обработаны все «статические данные», необходимые агенту для взаимодействия, после чего начинается фаза работы со знаниями. Агенту необходимо преобразовать семантическое понятие «запрос параметра», представленное с помощью языка SQL в форму, используемую в рассматриваемой технологической МАС. Для этого агент вновь обращается к онтологическому словарю, по синониму получает концепт «фрейм-запрос»

⁹ Предполагается, что концепт обладает этим атрибутом и его значение заполнено экспертом (инженером по знаниям) на этапе инициализации агента.

¹⁰ В тех случаях, когда коллизии с синонимами отсутствуют, и значения величин характеристик одинаковы для всех агентов, работа с триплетами ВСПТД идёт в обычном режиме.

(данний параметр является системным, поэтому не добавляется в виртуальное строковое пространство технологических данных агента), к которому присоединена процедура преобразования подобных семантических конструкций.¹¹

Получив таким образом фрейм-запрос, агент ERP-системы может сформировать перформатив (речевой акт) Request и передать его агенту PDM-системы ([листинг 4.6](#)).

Листинг 4.6. Пример речевого акта Request

```

1 {"запрос": {
2     "отправитель": {
3         "идентификатор": [
4             "1bc29b36f623ba82aa6724fd3b16718@tps.ifmo.ru"
5         ],
6         "Получатель": {
7             "идентификатор": [
8                 "8a8884ee553cceaa82aa6724fd3b167@tps.ifmo.ru"
9             ],
10            "содержимое": [
11                "$PCM.PLOTN=:; $PCM.CODE=09857865;",
12            ],
13            "протокол": "запрос-ответ",
14            "язык": "всптд",
15            "онтология": "полимерные композиционные материалы",
16        }
}

```

После получения запроса (Request) агент PDM-системы должен проанализировать его и дать ответ принимает ли он его (Agree) или нет (Refuse). Если запрос принят, агенту PDM-системы необходимо обработать его (т. е. аналогичным образом обратиться к онтологии, привести в соответствие свои внутренние понятия и т. д.) и вернуть один из следующих перформативов ([рис. 4.8](#)):

- Failure, в случае возникновения ошибок.
- Inform-done:inform, в случае, если запрос просто обработан (не требуется возвращать какой-либо результат).

¹¹ Преобразование осуществляется за счёт использования модели, представляющей *Объектно-реляционную проекцию* (Object-relational mapping — ORM), данный механизм подробно описан в работе [137].

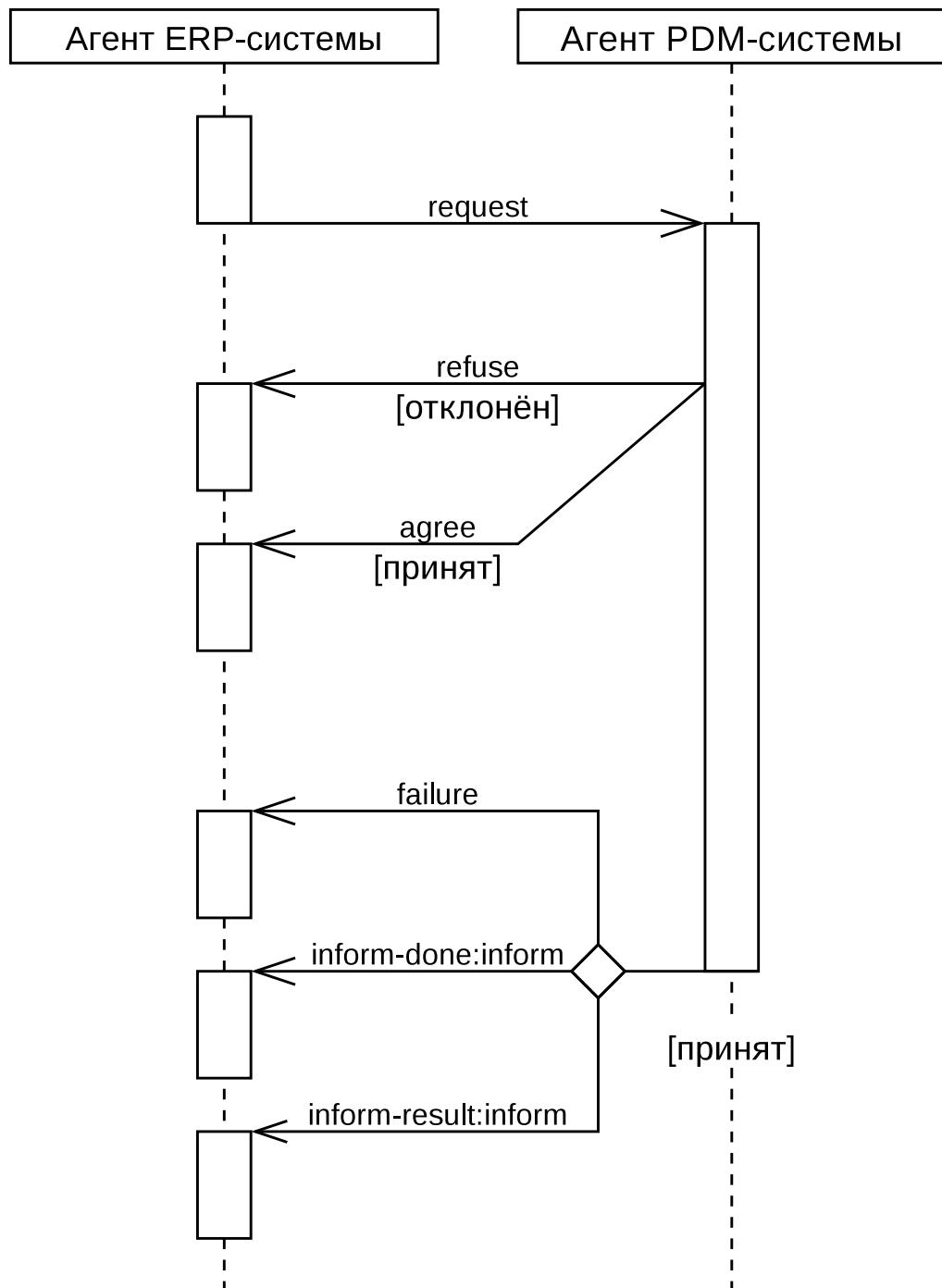


Рис. 4.8. AUML-диаграмма взаимодействия двух агентов при запросе ресурса

- `Inform-result:inform`, в случае, если запрос успешно обработан и получен некоторый результат, который необходимо вернуть запрашивающему агенту.

В рассматриваемом примере агенту ERP-системы будет передано сообщение, содержимым которого будет триплексная строка, описывающая запрашиваемый параметр, который будет преобразован к внутреннему представлению системы и помещён в базу. После этого оба агента вернутся в фазу ожидания.

Рассмотренный пример описывает самый простой из вариантов взаимодействия агентов внутри информационно-управляющей платформы технологической подготовки производства — взаимодействие по схеме «запрос-ответ», тем не менее даже он позволяет осуществить интеллектуальную интеграцию средств автоматизации в едином информационном пространстве технологической подготовки изделий из полимерных композиционных материалов. Использование же более сложных методов (описанных в [разд. 3.4](#)) позволит создать децентрализованную сеть управления ТПП, способную работать не только в рамках одного предприятия, но и выполнять задачи по интеграции внутри целого производственного кластера.

4.4. Выводы и результаты по четвёртой главе

Показана реализация многоагентной информационно-управляющей платформы технологической подготовки производства изделий из полимерных композиционных материалов. Выбор направления интеграции обусловлен высокой сложностью автоматизации данной задачи вследствие необходимости задействования дополнительных автоматизированных систем технологической подготовки производства, используемых в процессе проектирования состава полимерной композиции и анализа характеристик изготавливаемого

из него изделия, а также многоитерационностью этого процесса. В процессе конфигурирования информационно-управляющей платформы технологической подготовки производства:

1. Описана модель взаимодействия информационных потоков, возникающих в процессе технологической подготовки производства изделий из полимерных композиционных материалов.
2. Описаны конкретные автоматизированные системы и их роль в процессе выбора и проектирования нового материала, а также разработки технологии изготовления изделия из этого материала.
3. С помощью распределённого виртуального испытательного стенда создана модель интеграционной сети информационно-управляющей платформы технологической подготовки производства изделий из полимерных композиционных материалов, учитывающая специфику средств аппаратного обеспечения и позволяющая взаимодействовать с АСТПП посредством использования удалённых рабочих мест.
4. Разработаны и программно реализованы базовые классы многоагентной системы, описаны их функции и основные характеристики. Особое внимание уделено пользовательскому интерфейсу каждого агента, позволяющему строить динамические диалоговые формы, основанные на фреймах-терминалах виртуального строкового пространства технологических данных.
5. Описан жизненный цикл технологического агента информационно-управляющей платформы, а также программно реализованы архитектура агента и процесс взаимодействия агентов для осуществления интеллектуальной интеграции при решении конкретных задач технологической подготовки производства изделий из полимерных композиционных материалов.

Моделирование многоагентной информационно-технологической платформы технологической подготовки производства средствами распределённого виртуального испытательного стенда позволило оценить достоверность разработанного метода многоагентной интеграции, а также сделать вывод о том, что рассматриваемая технология может быть перенесена в реальные производственные условия.

Заключение

В работе выполнен комплекс научных исследований и разработок, основной целью которых являлось совершенствование существующих методов автоматизации решения задач технологической подготовки приборостроительного производства за счёт внедрения многоагентной системы, упрощающей интеграцию средств информационного обеспечения АСТПП.

В процессе выполнения диссертационной работы были получены следующие основные результаты:

1. Показана актуальность применения методов теории многоагентных систем для решения задач интеграции в рамках единого информационного пространства технологической подготовки производства.
2. Описаны математические модели многоагентной системы и многоагентной среды, базирующиеся на принципах теории множеств и теории алгебраических систем.
3. Описан специализированный язык взаимодействия технологических агентов, основанный на принципах теории виртуального строкового пространства и позволяющий агентам обмениваться технологическими данными и знаниями.
4. На базе сконфигурированного серверного кластера реализован распределённый виртуальный испытательный стенд для моделирования взаимодействия автоматизированных систем в едином информационном пространстве технологической подготовки производства.
5. Предложена модель взаимодействия автоматизированных систем технологической подготовки производства изделий из полимерных композиционных материалов.

6. Разработан и программно реализован прототип многоагентной системы, являющейся интеграционным ядром информационно-управляющей платформы для решения технологической подготовки производства.

Спроектированная многоагентная система позволяет существенно упростить внедрение современных систем автоматизации технологической подготовки производства за счёт их более тесной интеграции, а также снизить накладные расходы, связанные с промышленной эксплуатацией подобных систем. Предложенные методы могут быть адаптированы для решения технологических задач не только в приборостроении, но и в смежных отраслях.

Список литературы

1. Маталин А. А. Технология машиностроения: Учебник. 2 изд. СПб: Издательство «Лань», 2008. 512 с.
2. Scallan P. Process planning: the design/manufacture interface. Oxford, UK: Butterworth-Heinemann, 2003. 483 р.
3. Митрофанов С. П., Куликов Д. Д., Миляев О. Н., Падун Б. С. Технологическая подготовка гибких производственных систем / Под ред. С. П. Митрофanova. Л.: Машиностроение, 1987. 352 с.
4. Аверченков В. И., Каштальян И. А., Пархутин А. П. САПР технологических процессов, приспособлений и режущих инструментов. Мн.: Высшая школа, 1993. 288 с.
5. Горанский Г. К., Владимиров Е. В., Ламбин Л. Н. Автоматизация технического нормирования работ на металлорежущих станках с помощью ЭВМ. М.: Машиностроение, 1970. 224 с.
6. Цветков В. Д. Система автоматизации проектирования технологических процессов. М.: Машиностроение, 1972. 240 с.
7. Цветков В. Д. Системно-структурное моделирование и автоматизация проектирования технологических процессов. Мн.: Наука и техника, 1979. 264 с.
8. Зарубин В. М., Капустин Н. М., Павлов В. В. Автоматизированная система проектирования технологических процессов механосборочного производства. М.: Машиностроение, 1979. 248 с.
9. Корсаков В. С., Капустин Н. М., Темпельгоф К.-Х., Лихтенберг Х. Автоматизация проектирования технологических процессов в машиностроении. М.: Машиностроение, 1985. 304 с.
10. Вальков В. М., Вершин В. Е. Автоматизированные системы управления технологическими процессами. 3-е изд. Л.: Политехника, 1991. 269 с.

11. Яблочников Е. И., Маслов Ю. В. Автоматизация ТПП в приборостроении. Учебное пособие. СПб: СПб ГИТМО (ТУ), 2003. 104 с.
12. Зильбербург Л. И., Молочник В. И., Яблочников Е. И. Информационные технологии в проектировании и производстве. СПб: Политехника, 2008. 304 с.
13. Норенков И. П. Основы автоматизированного проектирования: Учеб. для вузов. Сер. Информатика в техническом университете. 2-е изд. М.: Изд-во МГТУ им. Н. Э. Баумана, 2002. 336 с.
14. Гаврилов Д. А. Управление производством на базе стандарта MRP II. СПб: Питер, 2003. 352 с.
15. Nasr E. A., Kamrani A. K. Computer-Based Design and Manufacturing: An Information-Based Approach. New York, NY, USA: Springer Science, 2007. 344 p.
16. ГОСТ Р ИСО 10303–2002. Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными; Введ. 20.12.2002. М.: Госстандарт России, 2003. 52 с.
17. Голицына Т. Д. Принципы организации и интерфейс унифицированного модуля интеграции pdm- и cad-систем // Научно-технический вестник Санкт-Петербургского государственного университета информационных технологий, механики и оптики. 2008. № 48. С. 186–190.
18. Афанасьев М. Я., Грибовский А. А. Организация единого информационного пространства виртуального предприятия // Научно-технический вестник Санкт-Петербургского государственного университета информационных технологий, механики и оптики. 2011. № 76. С. 113–118.
19. Фомина Ю. Н. Построение информационно-управляющей среды для технологической подготовки производства виртуального предприятия: Дис... канд. техн. наук: 05.11.14 / СПбГУ ИТМО. СПб, 2009. 147 с.

20. Саломатина А. А. Алгоритмы функционирования технологической подготовки производства в информационной среде виртуального предприятия: Дис. . . канд. техн. наук: 05.11.14 / СПбГУ ИТМО. СПб, 2011. 157 с.
21. Wood M. F., DeLoach S. A. An overview of the multiagent systems engineering methodology // First international workshop, AOSE 2000 on Agent-oriented software engineering. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2001. P. 207–221.
22. McDonald J. T., Talbert M. L., DeLoach S. A. Heterogeneous Database Integration Using Agent-Oriented Information Systems // Proceedings of the International Conference on Artificial Intelligence (2000. CSREA Press, 2000. P. 26–29.
23. Валетов В. А., Орлова А. А., Третьяков С. Д. Интеллектуальные технологии производства приборов и систем. Учебное пособие. СПб: СПбГУ ИТМО, 2008. 134 с.
24. Bai Q., Fukuta N. Advances in Practical Multi-Agent Systems. Studies in Computational Intelligence. Springer-Verlag, 2010. 474 p.
25. Staron R., Tichý P., Šindelář R., Maturana F. Methods to Observe the Clustering of Agents Within a Multi-Agent System // Holonic and Multi-Agent Systems for Manufacturing / Ed. by V. Mařík, V. Vyatkin, A. Colombo. Berlin, Heidelberg: Springer, 2007. Vol. 4659 of Lecture Notes in Computer Science. P. 127–136.
26. Heragu S. S., Graves R. J., Kim B.-I., Onge A. S. Intelligent agent based framework for manufacturing systems control // IEEE Transactions on Systems, Man, and Cybernetics, Part A. 2002. Vol. 32, no. 5. P. 560–573.
27. Энгельке У. Д. Как интегрировать САПР и АСТПП. М.: Машиностроение, 1990. 320 с.

28. Anumba C. J., Ugwu O. O., Ren Z. Agents and Multi-agent Systems In Construction. Spon Press, 2005. 329 p.
29. Рассел С., Норвиг П. Искусственный интеллект: современный подход: Пер. с англ. 2-е изд. М.: Издательский дом «Вильямс», 2006. 1408 с.
30. Wooldridge M., Jennings N. R. Agent Theories, Architectures, and Languages: A Survey // Intelligent Agents: ECAI-94 Workshop on Agent Theories, Architectures, and Languages / Ed. by M. J. Wooldridge, N. R. Jennings. Berlin: Springer, 1994. P. 1–39.
31. Городецкий В. И., Грушинский М. С., Хабалов А. В. Многоагентные системы (обзор) // Новости искусственного интеллекта. 1998. № 2. С. 41–61.
32. Brooks R. A. Intelligence Without Representation // Artificial Intelligence. 1991. Vol. 47, no. 1–3. P. 139–159.
33. Genesereth M. R., Nilsson N. J. Logical foundations of artificial intelligence. San Francisco: Morgan Kaufmann, 1987. 405 p.
34. Bratman M. E. Intention, Plans, and Practical Reason. Cambridge, MA, USA: Harvard University Press, 1987. 208 p.
35. Rao A. S., Georgeff M. P. BDI-agents: From Theory to Practice // Proceedings of the First Intl. Conference on Multiagent Systems. San Francisco: 1995. P. 704–710.
36. Корепанов В. О. Модели рефлексивного группового поведения и управления. М.: ИПУ РАН, 2011. 127 с.
37. Gerber C., Siekmann J., Vierke G. Holonic Multi-Agent Systems: Tech. Rep. RR-99-03. Germany: DFKI, Kaiserslautern, 1999.
38. Wooldridge M. An Introduction to Multiagent Systemes. John Wiley & Sons Limited, 2002. 348 p.
39. Singh M. P. Multiagent Systems: A Theoretical Framework for Intentions, Know-how, and Communications. Secaucus, NJ, USA: Springer-Verlag New

- York, Inc., 1994. 168 p.
40. Bussmann S., Jennings N. R., Wooldridge M. On the identification of agents in the design of production control systems // First international workshop, AOSE 2000 on Agent-oriented software engineering. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2001. P. 141–162.
 41. Vieira R., Moreira A., Wooldridge M., Bordini R. H. On the Formal Semantics of Speech-Act Based Communication in an Agent-Oriented Programming Language // J. Artif. Int. Res. 2007. — June. Vol. 29. P. 221–267.
 42. Fischer K., Schillo M., Siekmann J. Holonic Multiagent Systems: A Foundation for the Organisation of Multiagent Systems // Proceedings of the First International Conference on Applications of Holonic and Multiagent Systems (HoloMAS'03). 2003.
 43. Tharumarajah A., Wells A. J., Nemes L. Comparison of emerging manufacturing concepts // Proceedings of the IEEE International Conference on Systems, Man and Cybernetics. Vol. 1. 1998. P. 325–331.
 44. Афанасьев М. Я., Грибовский А. А. Реализация модуля управления виртуальным предприятием в PDM-системе ENOVIA-SmarTeam // Сборник тезисов докладов конференции молодых учёных, Выпуск 2. Труды молодых учёных / Под ред. В. О. Никифорова. СПб: СПбГУ ИТМО, 2011. С. 258–259.
 45. Афанасьев М. Я., Саломатина А. А., Алёшина Е. Е., Яблочников Е. И. Применение многоагентных технологий для реализации системы управления виртуальным предприятием // Научно-технический вестник Санкт-Петербургского государственного университета информационных технологий, механики и оптики. 2011. № 75. С. 105–111.
 46. Bečvář P., Charvát P., Pospíšil M. P. J. et al. ExPlanTech/ExtraPLANT: Production Planning and Supply-Chain Management Multi-Agent Solution // EXP – in search of innovation. 2003. Vol. 3, no. 3. P. 116–125.

47. Шереметов Л. Б. Методы и модели конфигурирования адаптивных сетей поставок на основе многоагентных коалиционных систем: Автореф. дис. . . д-ра техн. наук: 05.13.01 / СПИИРАН. СПб, 2010. 39 с.
48. Cooperative Control of Distributed Multi-Agent Systems / Ed. by J. Shamma. John Wiley & Sons Limited, 2007. 435 p.
49. Sun D. Synchronization and Control of Multiagent Systems. Automation and Control Engineering Series. CRC Press, 2011. 177 p.
50. Intelligent Manufacturing Systems [Электронный ресурс]. URL: <http://www.ims.org/> (дата обращения: 30.08.2011).
51. Wang L., Nee A. Collaborative Design and Planning for Digital Manufacturing. Springer, 2009. 413 p.
52. Koestler A. The ghost in the machine. The danube, 2. impr edition. London: Hutchinson, 1979. 384 p.
53. Скобелев О. П. Открытые мультиагентные системы для холонических предприятий // Искусственный интеллект. 2001. № 3. С. 98–119.
54. Mella P. The holonic revolution. Holons, holarchies and holonic networks. The ghost in the production machine. Pavia: Pavia University Press, 2009. 124 p.
55. Giret A., Botti V. Analysis and Design of Holonic Manufacturing Systems // Proceedings of the 18th International Conference on Production Research. 2005.
56. Giret A. A multi agent methodology for holonic manufacturing systems // Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems. AAMAS '05. New York, NY, USA: ACM, 2005. P. 1375–1385.
57. Giret A., Botti V. From system requirements to holonic manufacturing system analysis // International Journal of Production Research. 2006. — October. Vol. 44, no. 18–19. P. 3917–3928.

58. Suda H. Future Factory System Formulated in Japan // Japanese Journal of Advanced Automation Technology. 1989. Vol. 2, no. 10. P. 15–25.
59. Babiceanu R., Chen F. Development and Applications of Holonic Manufacturing Systems: A Survey // Journal of Intelligent Manufacturing. 2006. Vol. 17. P. 111–131.
60. Borangiu T., Gilbert P., Ivanescu N.-A., Rosu A. An implementing framework for holonic manufacturing control with multiple robot-vision stations // *Eng. Appl. Artif. Intell.* 2009. — June. Vol. 22. P. 505–521.
61. Borangiu T., Raileanu S., Rosu A. et al. Semi-Heterarchical Distributed Control of a Holonic Manufacturing Cell // Journal of Control Engineering and Applied Informatics. 2009. Vol. 11. P. 14–21.
62. Botti V., Giret A. ANEMONA: A Multi-agent Methodology for Holonic Manufacturing Systems. Springer series in advanced manufacturing. London, UK: Springer-Verlag, 2008. 214 p.
63. Bussmann S. An Agent-Oriented Architecture for Holonic Manufacturing Control // Proceedings of First International Workshop on IMS. 1998.
64. Bussmann S., Schild K. An agent-based approach to the control of flexible production systems // Proceedings of the 8th IEEE Int. Conf. on Emergent Technologies and Factory Automation. 2001. P. 169–174.
65. Bussmann S., Schild K., Ag D. Self-Organizing Manufacturing Control: An Industrial Application of Agent Technology // Proceedings of the Fourth International Conference on MultiAgent Systems (ICMAS-2000). Washington, DC, USA: IEEE Computer Society, 2000. P. 87–94.
66. Bussmann S., Sieverding J. Holonic control of an engine assembly plant an industrial evaluation // Proceedings of the IEEE International Conference on Systems, Man and Cybernetics. Vol. 5. 2001. P. 3151–3156.
67. Christensen J. H. Holonic Manufacturing Systems: Initial Architecture and Standards Directions // Proceedings of First European Conference

- on Holonic Manufacturing Systems, European HMS Consortium. 1994. P. 53–88.
68. Fletcher M., Garcia-Herreros E., Christensen J. H. et al. An Open Architecture for Holonic Cooperation and Autonomy // Proceedings of the 11th International Workshop on Database and Expert Systems Applications. DEXA '00. Washington, DC, USA: IEEE Computer Society, 2000. P. 224–230.
 69. Fletcher M., Deen S. M. Fault-Tolerant Holonic Manufacturing Systems // Concurrency and Computation Practice & Experience. 2001. Vol. 13. P. 43–70.
 70. Fletcher M., Brennan R. Designing a holonic control system with IEC 61499 function blocks // Proceedings of the International Conference on Intelligent Modeling and Control. 2001.
 71. International Standard IEC 61499-1: Function Blocks — Part 1: Architecture. Geneva, Switzerland: IEC Tech. Comm. 65, Working group 6, Bureau Central de la Commission Electrotechnique Internationale (International Electrotechnical Commission), 2005.
 72. Дубинин В. Н. Концептуальное моделирование систем управления на основе функциональных блоков IEC 61499 // Вестник ТГТУ. 2009. Т. 15, № 3. С. 467–477.
 73. Елькин И. В., Кустарев П. В. Модель абстрактных функциональных блоков // Научно-технический вестник СПбГИТМО (ТУ). Выпуск 10. Информация и управление в технических системах. С. 55–61.
 74. International Standard IEC 61131-3: Programmable Controllers — Part 3. Geneva, Switzerland: Bureau Central de la Commission Electrotechnique Internationale (International Electrotechnical Commission), 1993.
 75. Deen S. M., Fletcher M. Temperature Equilibrium in Multi-Agent Manufacturing Systems // Proceedings of the 11th International Workshop on

- Database and Expert Systems Applications / Ed. by A. M. Tjoa, R. R. Wagner, A. Al-Zobaidie. Washington, DC, USA: IEEE Computer Society, 2000. DEXA '00. P. 259–270.
76. Brennan R. W., Norrie D. H. From FMS to HMS // Agent-Based Manufacturing: Advances in the Holonic Approach / Ed. by S. M. Deen. Berlin, Heidelberg: Springer-Verlag, 2003. P. 31–49.
77. Brennan R., Hall K., Mařík V. et al. A Real-Time Interface for Holonic Control Devices // Holonic and Multi-Agent Systems for Manufacturing / Ed. by V. Mařík, D. McFarlane, P. Valckenaers. Berlin, Heidelberg: Springer, 2003. Vol. 2744 of Lecture Notes in Computer Science. P. 1088–1098.
78. Fischer K. An agent-based approach in holonic manufacturing systems // Proceedings of the 3rd IEEE/IFIP international conference on Intelligent systems for manufacturing : multi-agent systems and virtual organizations: multi-agent systems and virtual organizations. Norwell, MA, USA: Kluwer Academic Publishers, 1998. P. 3–12.
79. Müller J. The design of intelligent agents: a layered approach. New-York, NY: Springer-Verlag, 1996. Vol. 1177 of Lecture notes in computer science. 227 p.
80. Maturana F. P., Norrie D. H. Distributed decision-making using the contract net within a mediator architecture // *Decis. Support Syst.* 1997. — May. Vol. 20. P. 53–64.
81. Van Brussel H., Wyns J., Valckenaers P. et al. Reference architecture for holonic manufacturing systems: PROSA // *Comput. Ind.* 1998. — November. Vol. 37. P. 255–274.
82. Verstraete P., Germain B. S., Valckenaers P. et al. Engineering manufacturing control systems using PROSA and delegate MAS // *Int. J. Agent-Oriented Softw. Eng.* 2008. — January. Vol. 2. P. 62–89.

83. Hadeli, Valckenaers P., Kollingbaum M., Van Brussel H. Multi-agent coordination and control using stigmergy // *Comput. Ind.* 2004. — January. Vol. 53. P. 75–96.
84. Grobbelaar S., Ulieru M. Holonic Stigmergy as a Mechanism for Engineering Self-Organizing Applications // Proceedings of the Third International Conference on Informatics in Control, Automation and Robotics. 2006. P. 5–10.
85. Leitão P., Restivo F. Holonic adaptive production control systems // Proceedings of special session on Agent-based Intelligent Automation and Holonic Control Systems of the 28th Anual Conference of the IEEE Industrial Society. Vol. 4. 2002. P. 2968–2973.
86. Leitão P., Restivo F. Identification of ADACOR holons for manufacturing control // Proceedings of 7th IFAC Workshop on Intelligent Manufacturing Systems. 2003. P. 109–114.
87. Chirn J. L., Mcfarlane D. C. A holonic component-based architecture for manufacturing control systems // Proceedings of MAS '99. 1999. P. 219–223.
88. Iwamura K., Taimizu Y., Sugimura N. A study on Real-Time Scheduling Methods in Holonic Manufacturing systems // Knowledge and Skill Chains in Engineering and Manufacturing / Ed. by E. Arai, F. Kimura, J. Goossenaerts, K. Shirase. Boston: Springer, 2005. Vol. 168 of IFIP International Federation for Information Processing. P. 301–312.
89. Iwamura K., Okubo N., Tanimizu Y., Sugimura N. Real-time scheduling for holonic manufacturing systems based on estimation of future status // *International Journal of Production Research.* 2006. — October. Vol. 44, no. 18–19. P. 3657–3675.
90. Iwamura K., Nakano A., Tanimizu Y., Sugimura N. A Study on Real-Time Scheduling for Holonic Manufacturing Systems — Simulation for Estimation of Future Status by Individual Holons // *Holonic and Multi-Agent Systems*

- for Manufacturing / Ed. by V. Mařík, V. Vyatkin, A. Colombo. Berlin, Heidelberg: Springer, 2007. Vol. 4659 of Lecture Notes in Computer Science. P. 205–214.
91. Iwamura K., Kuwahara S., Tanimizu Y., Sugimura N. A Real-time Scheduling Method Considering Human Operators in Autonomous Distributed Manufacturing Systems // Manufacturing Systems and Technologies for the New Frontier / Ed. by M. Mitsuishi, K. Ueda, F. Kimura. London, UK: Springer, 2008. P. 283–288.
 92. Leitão P., Colombo A., Restivo F. An Approach to the Formal Specification of Holonic Control Systems // Holonic and Multi-Agent Systems for Manufacturing / Ed. by V. Mařík, D. McFarlane, P. Valckenaers. Berlin, Heidelberg: Springer, 2003. Vol. 2744 of Lecture Notes in Computer Science. P. 59–70.
 93. Multi-Agent Programming. Languages, Tools and Applications / Ed. by R. Bordini, M. Dastani, J. Dix, A. Seghrouchni. Springer, 2009. 296 p.
 94. Цейтин Г. С. О соотношении естественного языка и формальной модели // Вопросы кибернетики. 1982. С. 28–34.
 95. Филиппов А. Н. Разработка и исследование методов экспертных систем в САПР ТП механической обработки: Дис. . . канд. техн. наук. Л., 1991. 148 с.
 96. Филиппов А. Н. Автоматизированная система проектирования технологических процессов с использованием подхода экспертных систем // Актуальные проблемы современного программирования. Сборник ЛИИАН. 1989. С. 113–122.
 97. «Техком-SERVICE Plus», V 3.5. — Описание применения. СПб: НТЦ Техком, 1993. 205 с.
 98. Сисюков А. Н. Разработка и применение специализированных экспертизных систем для САПР технологических процессов механической

- обработки заготовок: Дис... канд. техн. наук. СПб, 2007. 150 с.
99. Грибовский А. А., Афанасьев М. Я. Декомпозиция структуры трехмерных моделей на наборы конструктивных элементов с использованием примитивов // Сборник тезисов докладов конференции молодых ученых, Выпуск 2. Труды молодых ученых / Под ред. В. О. Никифорова. СПб: СПбГУ ИТМО, 2011. С. 281.
 100. Афанасьев М. Я. Использование библиотеки Open CASCADE для параметрического 3D моделирования // Сборник тезисов докладов конференции молодых ученых, Выпуск 3. Труды молодых ученых / Под ред. В. О. Никифорова. СПб: СПбГУ ИТМО, 2010. С. 117–118.
 101. Афанасьев М. Я. Вероятностная модель рассуждений в машине логического вывода экспертной системы «TexAssистент» // Сборник трудов конференции молодых учёных, Выпуск 2. Биомедицинские технологии, мехатроника и робототехника / Под ред. В. Л. Ткалич. СПб: СПбГУ ИТМО, 2009. С. 296–299.
 102. Тамм Б. Г., Кюттнер Р. А. Реализация специализированных инструментальных систем программирования для САПР ТП // Автоматизированное проектирование и производство в машиностроении / Под ред. Ю. М. Соломенцева, В. Г. Митрофанова. М.: Машиностроение, 1986. С. 230–244.
 103. Заде Л. Понятие лингвистической переменной и его применение к принятию приближенных решений. М.: Мир, 1976. 166 с.
 104. Афанасьев М. Я., Филиппов А. Н. Применение методов нечёткой логики в автоматизированных системах технологической подготовки производства // Изв. вузов. Приборостроение. 2010. Т. 53, № 6. С. 38–42.
 105. Бруевич Н. Г., Боброва И. В., Челищев Б. Е. Математические основы автоматизации проектирования процессов механической обработки деталей // Изв. АН СССР. Техническая кибернетика. 1978. № 1. С. 134–148.

106. Minsky M. A Framework for Representing Knowledge // Mind Design: Philosophy, Psychology, Artificial Intelligence / Ed. by J. Haugeland. Cambridge, MA: MIT Press, 1981. P. 95–128.
107. Гаврилова Т. А., Хорошевский В. Ф. Базы знаний интеллектуальных систем. СПб: Питер, 2000. 384 с.
108. ГОСТ 19781–90. Обеспечение систем обработки информации программное. Термины и определения; Введ. 01.01.92. М.: Стандартинформ, 2005. 16 с.
109. de Rivera G. G., Ribalda R., de Castro A., Garrido J. Performance of an Open Multi-Agent Remote Sensing Architecture Based on XML-RPC in Low-Profile Embedded Systems // PAAMS / Ed. by Y. Demazeau, J. Pavón, J. M. Corchado, J. Bajo. Vol. 55 of Advances in Intelligent and Soft Computing. Springer, 2009. P. 520–528.
110. Gray D. N., Hotchkiss J., LaForge S. et al. Modern Languages and Microsoft's Component Object Model // Commun. ACM. 1998. Vol. 41, no. 5. P. 55–65.
111. Оберг Р. Д. Технология COM+. Основы и программирование. М.: Издательский дом «Вильямс», 2000. 480 с.
112. Bollacker K., Evans C., Paritosh P. et al. Freebase: a collaboratively created graph database for structuring human knowledge // SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data. New York, NY, USA: ACM, 2008. P. 1247–1250.
113. Bellifemine F., Caire G., Greenwood D. Developing Multi-Agent Systems with JADE. John Wiley & Sons Limited, 2007. 286 p.
114. Searle J. Speech acts: an essay in the philosophy of language. Cambridge, UK: Cambridge University Press, 1969. 212 p.
115. Austin J. How to do things with words / Ed. by J. Urmson, S. Marina. 2 edition. Cambridge, MA, USA: Harvard University Press, 1975. 192 p.

116. ГОСТ Р ИСО/МЭК 7498-1-99. Информационная технология. Взаимодействие открытых систем. Базовая эталонная модель. Часть 1. Базовая модель; Введ. 18.03.99. М.: Госстандарт России, 1999. 58 с.
117. ГОСТ Р ИСО/МЭК 7498-2-99. Информационная технология. Взаимодействие открытых систем. Базовая эталонная модель. Часть 2. Архитерктура защиты информации; Введ. 18.03.99. М.: Госстандарт России, 1999. 36 с.
118. Powers S. Practical RDF. Sebastopol, CA, USA: O'Reilly & Associates, Inc., 2003. 331 p.
119. Germain S. B., Valckenaers P., Brussel V. H. et al. An agent-based approach to the control of flexible production systems // Proceedings of the 17th International Federation of Automatic Control (IFAC) Workshop. 2003. P. 207–212.
120. Antonopoulos N., Gillam L. Cloud Computing: Principles, Systems and Applications. Computer Communications and Networks. 1st edition. London, UK: Springer-Verlag, 2010. 396 p.
121. Adamczyk Z., Kociołek K. CAD/CAM technological environment creation as an interactive application on the Web // Journal of Materials Processing Technology. 2001. Vol. 109. P. 222–228.
122. Bauer B., Müller J. P., Odell J. Agent UML: A Formalism for Specifying Multiagent Interaction // in: Ciancarini, P.; Wooldridge, M. [Eds.], Agent Oriented Software Engineering. 2001. P. 91–103.
123. Швецов А. Н. Агентно-ориентированные системы: от формальных моделей к промышленным приложениям [Электронный ресурс] // Всероссийский конкурсный отбор обзорно-аналитических статей по приоритетному направлению «Информационно-телекоммуникационные системы». 2008. URL: http://window.edu.ru/window/library?p_rid=56179 (дата обращения: 02.01.2012).

124. Heiberger R. M., Holland B. Statistical Analysis and Data Display: An Intermediate Course with Examples in S-Plus, R, and SAS. Springer Texts in Statistics. Springer, 2004. 739 p.
125. Carley K. M., Reminga J., Storrick J., De Reno M. ORA User's Guide 2009: Tech. rep.: Institute for Software Research, School of Computer Science, Carnegie Mellon University, 2009.
126. Batagelj V., Mrvar A. Pajek — analysis and visualization of large networks // Graph Drawing Software, Ed. by M. Juenger, P. Mutzel. Berlin: Springer, 2003. P. 77–103.
127. Hibbard W., Rueden C., Emmerson S. et al. Java distributed components for numerical visualization in VisAD // Commun. ACM. 2005. — March. Vol. 48. P. 98–104.
128. Hall M., Frank E., Holmes G. et al. The WEKA data mining software: an update // SIGKDD Explor. Newsl. 2009. — November. Vol. 11. P. 10–18.
129. Ahammad S. iReport 3.7. 1st edition. Packt Publishing, 2010. 221 p.
130. Madadhain J., Fisher D., Smyth P. et al. Analysis and visualization of network data using JUNG // Journal of Statistical Software. 2005. Vol. 10. P. 1–35.
131. Cicirelli F., Furfaro A., Giordano A., Nigro L. HLA_ACTOR_REPAST: An approach to distributing RePast models for high-performance simulations // Simulation Modelling Practice and Theory. 2011. Vol. 19, no. 1. P. 283–300.
132. Лутц М. Программирование на Python: Пер. с англ. 2-е изд. СПб: Символ-Плюс, 2002. 1136 с.
133. Лутц М. Изучаем Python: Пер. с англ. 3-е изд. СПб: Символ-Плюс, 2009. 848 с.
134. Саммерфилд М. Программирование на Python 3. Подробное руководство: Пер. с англ. СПб: Символ-Плюс, 2002. 608 с.

135. Farrell J., Nezlek G. Rich internet Applications the Next Stage of Application Development // Information Technology Interfaces, 2007. ITI 2007. 29th International Conference on / IEEE. 2007. P. 413–418.
136. Fraternali P., Comai S., Bozzon A., Toffetti Carughi G. Engineering Rich Internet Applications With a Model-driven Approach // [ACM Trans. Web.](#) 2010. Vol. 4, no. 2. P. 1–47.
137. Афанасьев М. Я., Филиппов А. Н. Создание динамических моделей баз данных технологического назначения на языке Python // Изв. вузов. Приборостроение. 2010. Т. 53, № 6. С. 59–62.

Приложение А

Представление языка ИУП ТПП в расширенной форме Бэкуса–Наура

A.1. Тип данных

ВещЧисло = [“+”|“-”] НатЧисло [“.”[НатЧисло]]
 [(“e”|“E”)] [“+”|“-”] НатЧисло].

НатЧисло = Цифра {Цифра}.

Стока = “”” {ЛатСимвол|КирСимвол|Цифра} “””.

Цифра = “0”|“1”|“2”|“3”|“4”|“5”|“6”|“7”|“8”|“9”.

ЛатСимвол = “a”|…|“z”|“A”|…|“Z”.

КирСимвол = “а”|…|“я”|“А”|…|“Я”.

Кортеж = “{” Элемент {“,,” Элемент} “}”.

Элемент = ВещЧисло|Стока|Переменная|Константа|
 Промежуток|ВероятПараметр.

Промежуток = (“[”|“(“) ВещЧисло “;” ВещЧисло (“]”|“)”).

ВероятПараметр = (ВещЧисло|Стока)“@”[“+”|“-”] НатЧисло.

A.2. Триплет

Триплет = “\$” Префикс “.” Имя “=” [“:]”
 [Значение] “;”.

Префикс = (ЛатСимвол|“_”) {ЛатСимвол|“_”}.

Имя = (ЛатСимвол|“_”) {ЛатСимвол|Цифра|“_”}.

Значение = ТипДанных|Функция|Триплет|Синтагма|Продукция.

A.3. Операция

МатОпер = “+”|“-”|“*”|“/”|“~”|“%”.
 ЛогОпер = “И”|“ИЛИ”.
 Отношения = “>”|“<”|“=”|“>=”|“<=”|“<>”.
 ОперПрисвоения = “:=”.
 ОперПеречисления = “,”.

A.4. Функция

Функция = ИмяФункции“(”Аргумент
 {ОперПеречисления Аргумент}“)”).
 ИмяФункции = ЛатСимвол {ЛатСимвол}
 Переменная = “#” Имя
 Аргумент = ВещЧисло|Строка|Кортеж|Переменная|
 Функция.

A.5. Фрейм

Фрейм = Триплет {Триплет} “//”.

A.6. Синтагма

Синтагма = “\$” Имя Реляция “\$” Имя Оценка “;”.
 Реляция = “_” ЛатСимвол {ЛатСимвол} “_”.
 Оценка = ЛогВыражение | Формула.

A.7. Продукция

Продукция = ‘‘ЕСЛИ’’ Условие
 ‘‘ТО’’ Действие {ОперПеречисления
 Действие} .

Условие = ЛогВыражение {ЛогОпер (ЛогВыражение |
 ‘‘(’’ Условие ‘‘)’’)} .

Действие = Переменная ОперПрисвоения Формула .

ЛогВыражение = Формула Отношение Формула .

Формула = Компонент {МатОпер (Элемент |
 ‘‘(’’ Формула ‘‘)’’)} .

Компонент = ВещЧисло | Стока | Кортеж | Переменная |
 Константа | Функция .

Приложение Б

Онтологический словарь ИУП ТПП

Листинг Б.1. Пример концептов онтологии ПКМ

```

1 {"-name"      : "Параметры полимерных композиционных материалов",
2 "-prefix"    : "PCM", // Определяет иерархическую связь
3 "-row"       : [
4   { "denom"     : "Объёмное сопротивление",
5   "format"    : "9999", // Формат записи
6   "name"      : "$PCM.VOLRES", // Имя концепта
7   "synonyms": ["Volume Resistivity",
8   "R_объём."],
9   "glinks"    : ["$PCM.RES"], // Общие связи
10  "procs"     : ["pcm.get_volres"] // Присоединённые процедуры
11  "params"    : [{"стандарт":"ASTM D 257",
12  "ед.изм.": ["Ом*см", "Omh*см"]},
13  {"стандарт":"IEC 60093",
14  "ед.изм.": ["Ом*м", "Omh*m"]}],
15 ] // Дополнительные параметры
16 },
17 { "denom"     : "Температура плавления",
18 "format"    : "999V99",
19 "name"      : "$PCM.MELTTEMP",
20 "synonyms": ["Melting Temperature",
21 "T_плавл."],
22 "glinks"    : ["$PCM.VICATTEMP"],
23 "procs"     : [],
24 "params"    : [{"стандарт":"ISO 11357-1/-3",
25  "ед.изм.": ["оС"]},
26  {"стандарт":"ASTM D 3418",
27  "ед.изм.": ["оС"]}],
28 ],
29 },
30 { "denom"     : "Температура размягчения по Вика",
31 "format"    : "999V99",
32 "name"      : "$PCM.VICATTEMP",
33 "synonyms": ["Vicat softening temperature",
34 "T_Вика."],
35 "glinks"    : ["$PCM.MELTTEMP"],
36 "procs"     : ["pcm.set_temp", "pcm.set_force"],
37 "params"    : [{"стандарт":"ISO 306",
38  "ед.изм.": ["оС"]},
39  {"нагрузка": [10Н, 50Н]}],
40 ],
41 },
42 { "denom"     : "Светопропускание",
43 "format"    : "999V9",

```

```
44 "name"      : "$PCM.LTR",
45 "synonyms": ["Light Transmittance"] ,
46 "glinks"    : [] ,
47 "procs"     : [] ,
48 "params"    : [{"стандарт":"ASTM D 1003"} ,
49 "ед.изм." : [%] ,
50 ]
51 },
52 { "denom"    : "Индекс плавления",
53 "format"   : "999V999",
54 "name"     : "$PCM.MFI",
55 "synonyms": ["Melt Flow Index","инд_плавл"] ,
56 "glinks"   : ["$PCM.MELTTEMP","$PCM.VICATTEMP"] ,
57 "procs"    : ["pcm.eval_mfi"] ,
58 "params"   : [{"стандарт":"ASTM D 1238"} ,
59 "ед.изм." : [г/мин,"g/min"] ,
60 "доп.парам." : ["температура","нагрузка"]
61 ]
62 },
63 { "denom"    : "Водопоглощение",
64 "format"   : "999V9",
65 "name"     : "$PCM.WABS",
66 "synonyms": ["Water absorption"] ,
67 "glinks"   : [] ,
68 "procs"    : [] ,
69 "params"   : [{"стандарт":"ASTM D 570"} ,
70 "ед.изм." : [%] ,
71 "время." : [24 ч.]
72 ]
73 },
74 .
75 .
76 .
77 ]
78 }
```

Листинг Б.2. Пример концептов онтологии СОЖ

```

1 { "-name"      : "Список концептов СОЖ",
2 "-prefix"     : "C",
3 "-row"        : [
4 { "denom"     : "ГОСТ или ТУ",
5 "format"     : "X(45)",
6 "name"       : "$C.IST",
7 },
8 { "denom"     : "Наименование",
9 "format"     : "X(15)",
10 "name"      : "$C.NM",
11 },
12 { "denom"    : "Наименование группы
13 обрабатываемого материала",
14 "format"   : "X(40)",
15 "name"     : "$C.NMM",
16 },
17 { "denom"    : "Обрабатываемый материал",
18 "format"   : "99",
19 "name"     : "$C.PGS",
20 },
21 { "denom"    : "Способ обработки",
22 "format"   : "99",
23 "name"     : "$C.PGS2",
24 },
25 { "denom"    : "Группы обработки",
26 "format"   : "X(50)",
27 "name"     : "$C.SOB",
28 },
29 .
30 .
31 .
32 ]
33 }

```

Приложение В

Стандарт FIPA

Таблица В.1
Ключевые спецификации FIPA

Индекс	Описание
00001	Спецификация абстрактной архитектуры (Abstract Architecture Specification)
00007	Спецификация языка описания содержимого (Content Languages Specification)
00008	Спецификация языка описания содержимого SL (SL Content Language Specification)
00023	Спецификация управления агентами (Agent Management Specification)
00024	Спецификация транспортировки сообщений агентов (Agent Message Transport Specification)
00025	Спецификация библиотеки протокола взаимодействия (Interaction Protocol Library Specification)
00026	Спецификация запроса протокола взаимодействия (FIPA Request Interaction Protocol Specification)
00037	Спецификация библиотеки коммуникативных актов (Communicative Act Library Specification)
00061	Спецификация структуры сообщения ACL (ACL Message Structure Specification)
00063	Спецификация агентного контроля (Control Agent Specification)
00063	Спецификация строкового представления сообщений языка FIPA ACL (FIPA ACL Message Representation in String Specification)
00082	Спецификация сетевого управления и резервирования (Network Management and Provisioning Specification)
00084	Спецификация протокола транспортировки сообщений агентов для HTTP (Agent Message Transport Protocol for HTTP Specification)
00086	Спецификация сервиса онтологий (Ontology Service Specification)
00094	Спецификация качества обслуживания (Quality of Service Specification)

Таблица В.2
Речевые акты FIPA

Речевой акт	Описание
Accept Proposal	Принятие предложения другого агента
Agree	Согласие выполнить какое-то действие
Cancel	Уведомление о прекращении выполнения действия
Call for Proposal	Запрос предложений на выполнение действия
Confirm	Уведомление об истинности некоторого суждения в случае сомнений получателя
Disconfirm	Уведомление о ложности некоторого суждения в случае, когда получатель считает его истинным
Failure	Уведомление о неудаче
Inform	Уведомление об истинности некоторого суждения
Inform If	Сообщение о том было ли суждение истинным или нет
Inform Ref	Уведомление о соответствии полученного объекта некоторому произвольному дескриптору
Not Understood	Осознание необходимости сделать что-либо при отсутствии понимания, что именно
Propagate	Сообщение о необходимости принять сообщение и передать его другим агентам в соответствии с заданным дескриптором
Propose	Предложение выполнить некоторое действие с учётом заданных условий
Proxy	Уведомление о необходимости передать сообщения другим агентам в соответствии с заданным дескриптором
Query If	Запрос о том было ли суждение истинным или нет
Query Ref	Запрос о соответствии имеющегося объекта некоторому дескриптору
Refuse	Отказ от выполнения некоторого действия без сообщения причины
Reject Proposal	Отклонение предложения выполнить некоторое действие во время проведения переговоров
Request	Запрос на выполнение некоторого действия
Request When	Запрос на выполнение некоторого действия в случае получения предложения
Request Whenever	Запрос на выполнение некоторого действия постоянно при получении предложений
Subscribe	Желание получать сообщения в соответствии с некоторым условием до тех пор, пока оно не изменится

Приложение Г

Описание интегрируемых систем автоматизации ТПП

Г.1. SALOME 6.4.0

Система SALOME 6.4.0 представляет собой набор инструментальных средств для решения инженерных задач, распространяемый под свободной лицензией. Основными функциями данного пакета являются:

- Взаимодействие с различными CAD-приложениями на уровне передачи данных об исследуемой модели.
- Позволяет с лёгкостью добавлять в свой состав новые модули и компоненты за счёт удобного программного интерфейса на языке Python, что даёт возможность без труда включить его в состав описываемой в данной работе гетерогенной ИУП ТПП.
- Обеспечивает удобный и эффективный пользовательский интерфейс, позволяющий существенно снизить временные задержки при проведения исследований ([рис. Г.1](#)).
- Создание/модификация, импорт/экспорт (IGES, STEP, BREP), восстановление/чистка CAD-моделей.
- Работа с сеточными моделями (MED, UNV, DAT, STL): редактирование, проверка качества для дальнейшей постобработки, импорт/экспорт.
- Обработка параметров специфицирующих исследуемые модели.
- Возможность осуществления расчётов с помощью одного или нескольких внешних решателей (например, таких свободных программных библиотек как *Code_Saturn*, *Code_Aster*, *OpenFOAM* и др.).

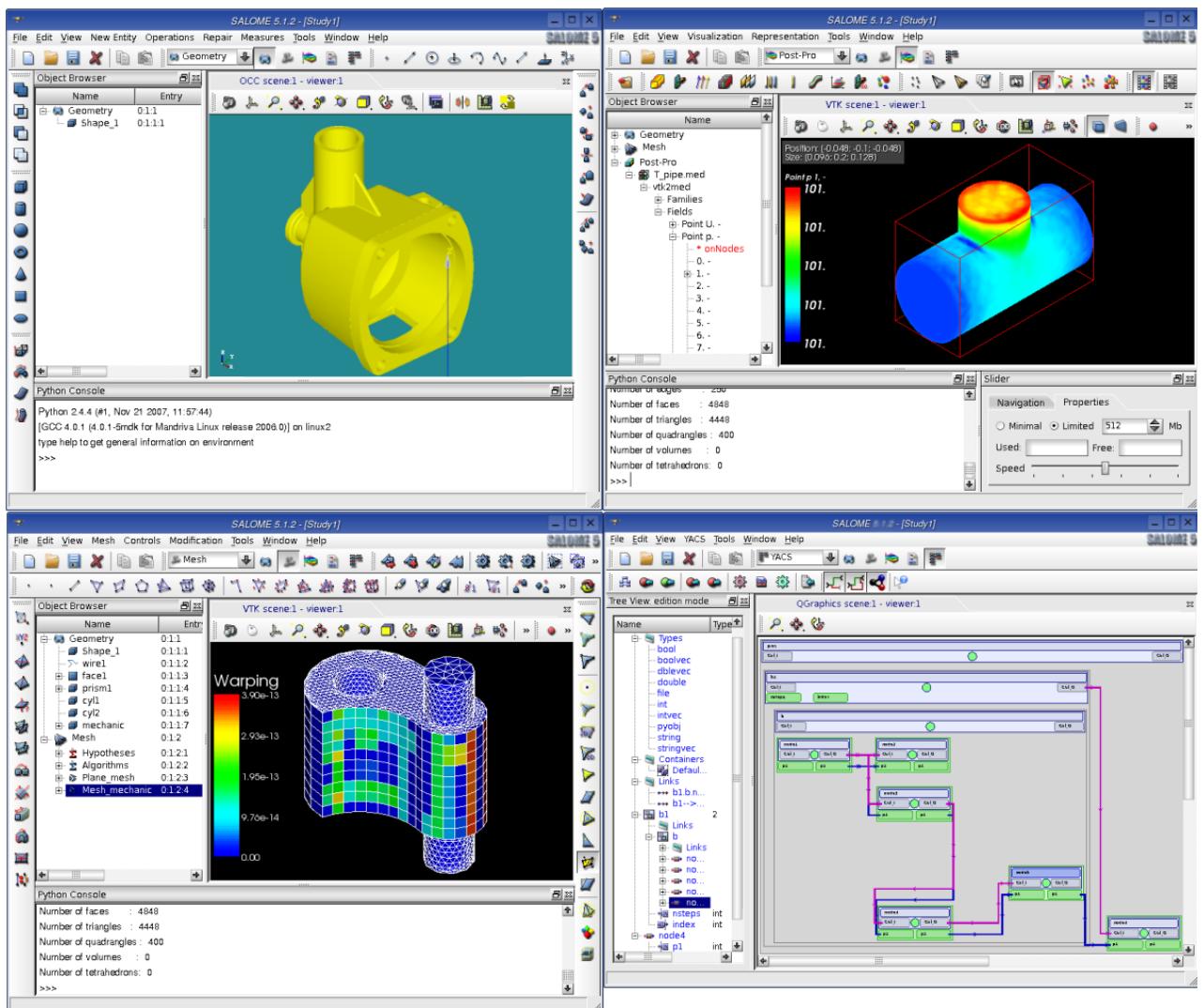


Рис. Г.1. Интерфейс системы SALOME

Г.2. SmarTeam V5R20

PDM-система SmarTeam представляет собой быстро внедряемое и экономичное решение по управлению данными об изделии для средних и малых предприятий, построенное с использованием принципа обмена информацией на основе BOM (Bill of Materials) — спецификаций, включающих в себя структуру изделия со всеми сопутствующими документами и ссылками. Этот пакет отличается от аналогичных систем других разработчиков наличием полноценной интеграции со всеми наиболее известными и используемыми сегодня в мировой практике CAD-системами, простотой в освоении и быстрой

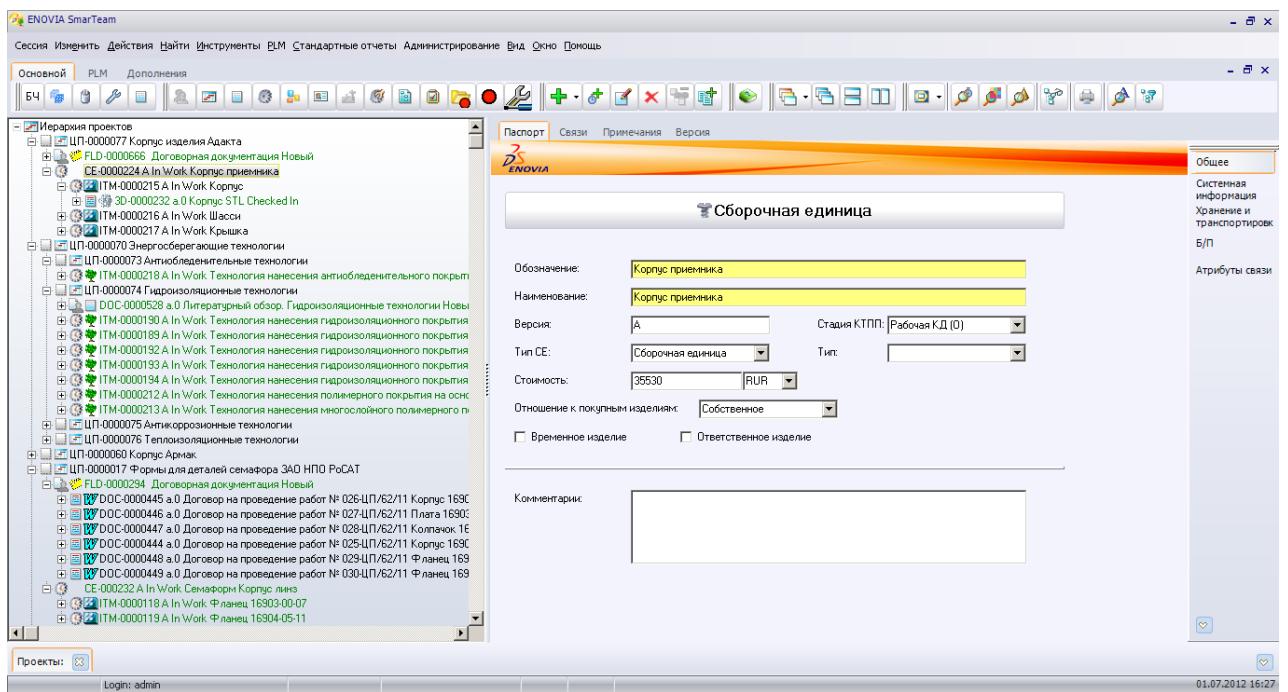


Рис. Г.2. Интерфейс PDM-системы SmarTeam V5R20

отдачей инвестиций после внедрения. Программный интерфейс приложения представлен на [рис. Г.2](#).

К основным функциям, реализуемым средствами PDM-системы SmarTeam в сфере проектирования и подготовки производства, относятся следующие [12]:

1. Ведение проектов: управление работами, процедурами и документами в составе проекта, контроль над выполнением проекта.
2. Планирование и диспетчирование работ.
3. Распределение прав доступа к информации между отдельными участниками проекта или их группами.
4. Организация и ведение распределённых архивов конструкторской, технологической и управлеченческой документации (электронные архивы).
5. Управление изменениями в документации: контроль версий документов, ведение протокола работы с документами, листов регистрации изменений и извещений.

6. Фиксирование стандартных этапов прохождения документов, контроль прохождения документов по этапам.
7. Интеграция с CAD/CAM-системами и их приложениями, используемыми при проектировании.
8. Контроль целостности проекта.
9. Поиск необходимой информации в проекте на основании запросов.

SmarTeam обеспечивает прием информации, создаваемой на различных этапах жизненного цикла изделий, причем ввод информации может выполняться либо в CAD-системах, либо в самой PDM. Хранение информации осуществляется в базе данных известных систем управления базами данных — например, Oracle, InterBase или MS SQL-Server. Средства, позволяющие создавать структуры баз данных и экранные формы представления информации в интерактивном режиме, позволяют легко адаптировать SmarTeam к условиям предприятия. Пользователи могут создавать базы данных стандартных и типовых деталей, используемых материалов, складов оснастки и др. Разработка программ для решения различных задач КТПП в среде SmarTeam выполняется с использованием *Интерфейса программирования приложений*, реализованного по технологии COM, что позволяет создавать расширения для него на любых языках программирования, поддерживающих данную технологию, в т. ч. и используемого в данной работе языке Python.

Г.3. OpenERP 6.0.3

Система OpenERP представляет собой модульную программную систему с открытым исходным кодом для управления ресурсами предприятия, написанную на языке Python. OpenERP является клиент-серверным приложением, взаимодействие в котором основано на протоколе XML-RPC, что позволяет

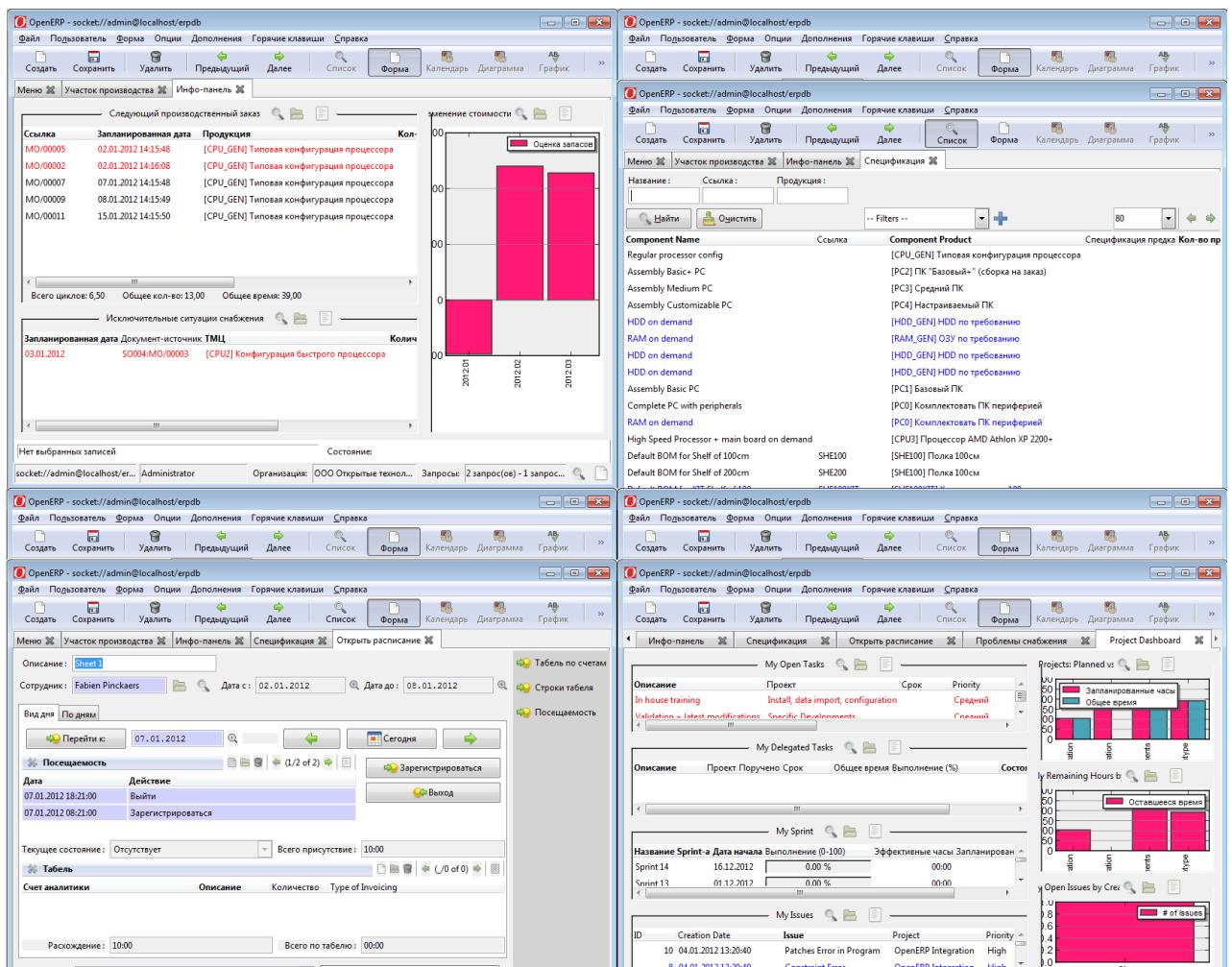


Рис. Г.3. Интерфейс системы OpenERP

обращаться к его функциям из любых приложений, поддерживающих работу по этому протоколу, а также даёт возможность бесшовной интеграции с ИУП ТПП. Серверная часть, в качестве СУБД использует PostgreSQL, клиентская часть выполнена как виде отдельного приложения (рис. Г.3), так и в виде web-сервиса, реализованного с использованием *Асинхронного JavaScript и XML* (Asynchronous Javascript and XML — AJAX). Основными модулями системы являются: Бухгалтерия, учёт активов, бюджет, система управления взаимоотношениями с клиентами, управление персоналом, готовая продукция, производство, продажи, закупки, запасы.

Г.4. PyCAM 0.5.1

PyCAM представляет собой систему генерации управляющих программ для трехкоординатных станков с числовым программным управлением, выпускаемую под свободной лицензией. В качестве входных данных система получает трёхмерную модель в формате STL или двумерный контур в форматах DXF или SVG и генерирует программу в формате *ISO-7bit* (последовательность *G-кодов*). Помимо непосредственно постпроцессинга PyCAM позволяет визуализировать путь инструмента (рис. Г.4).

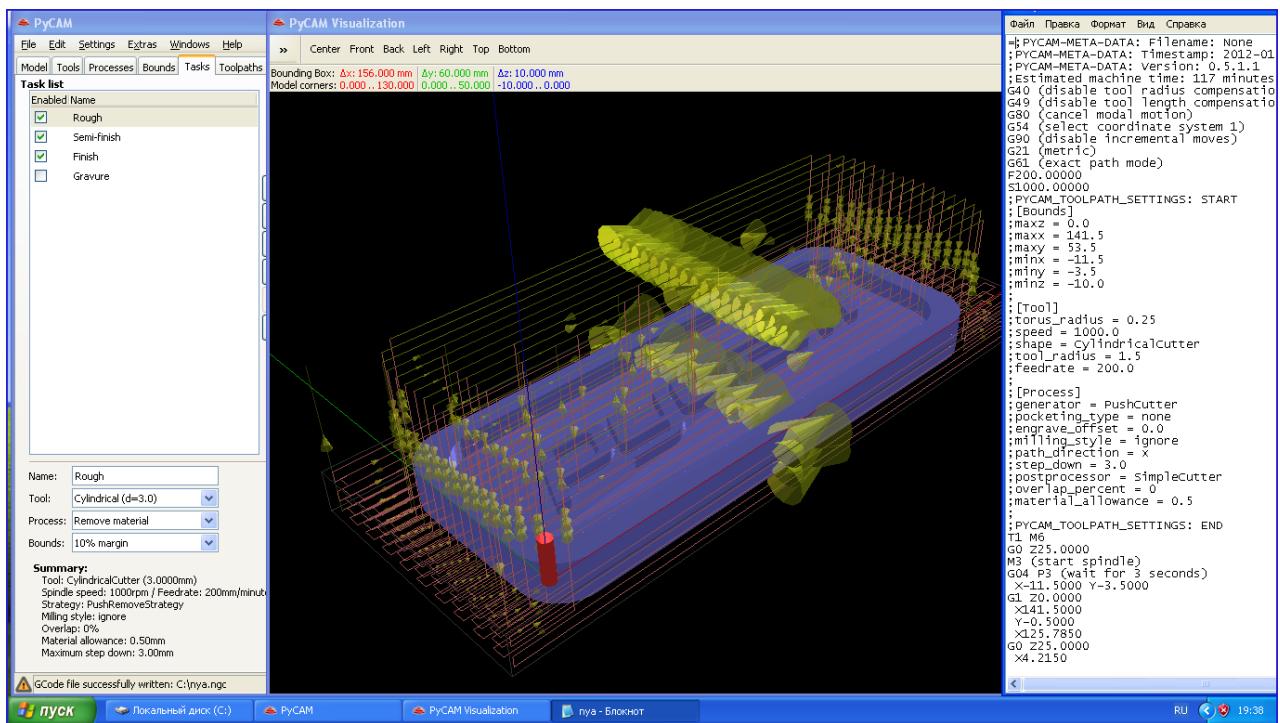


Рис. Г.4. Интерфейс CAM-системы PyCAM

Г.5. Вертикаль 2011

Система автоматизированного проектирования технологических процессов «Вертикаль 2011» является интеллектуальным редактором технологических документов (рис. Г.5), связанным с развитой базой данных типовых процессов, а также подроб-

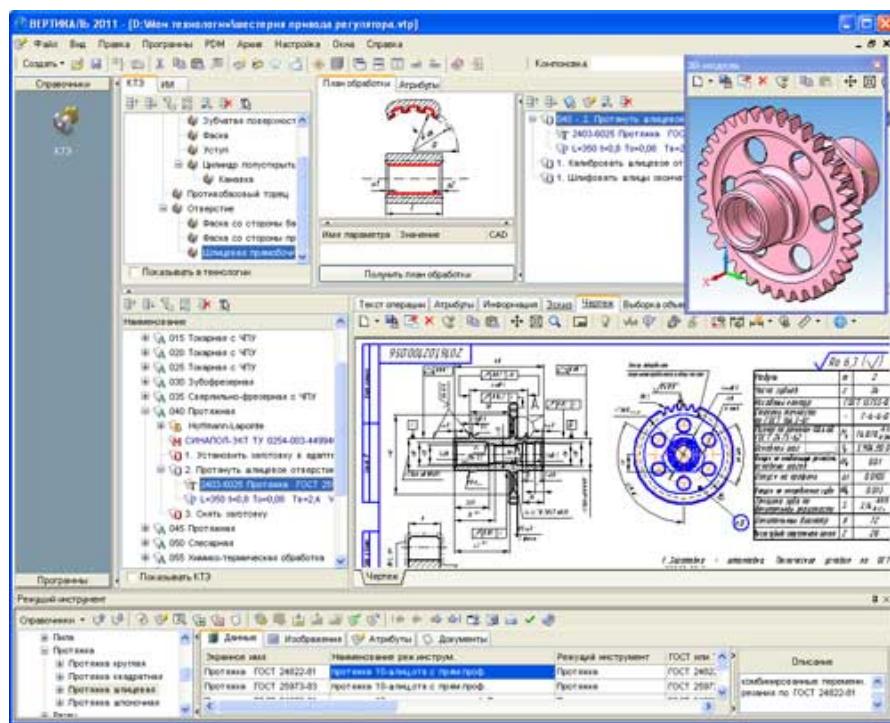


Рис. Г.5. Интерфейс САПР-системы Вертикаль 2011

ными технологическими и конструкторскими классификаторами. Система является проприетарным¹ ПО, разработанным отечественной компанией Аскон.

САПР ТП Вертикаль 2011 даёт возможность:

- разрабатывать технологические процессы в нескольких автоматизированных режимах;
- рассчитывать материальные и трудовые затраты на производство;
- формировать все необходимые комплекты технологической документации, используемые на предприятии;
- вести параллельное проектирование сложных и сквозных ТП группой технологов в реальном времени;
- формировать заказы на проектирование специальных средств технологического оснащения и создание управляющих программ;

¹ От англ. *proprietary* — собственнический, частный, патентованный.

- поддерживать актуальность технологической информации с помощью процессов управления изменениями.

Благодаря программируемому интерфейсу приложения, созданному по технологии СОМ (которая, как уже было отмечено выше, поддерживает работу с самыми разными языками, в т. ч. имеет реализацию для языка Python) система Вертикаль может быть легко интегрирована в описываемую многоагентную технологическую систему и стать одним из основных элементов разрабатываемой ИУП ТПП.

Г.6. PostgreSQL 9.1.1

PostgreSQL 9.1.1 представляет собой универсальную свободную объектно-реляционную систему управления базами данных, поддерживающую большинство функций, необходимых для создания на её основе информационной системы практически любой сложности.

Использование PostgreSQL в качестве единой базы, хранящей информацию о ПКМ обусловлено следующими факторами:

- Лёгкая интеграция СУБД PostgreSQL в структуру ИУП ТПП за счёт внутренней поддержки большинства современных языков программирования, включая Python.
- Поддержка триггеров и хранимых процедур, упрощающих работу с данными.
- Поддержка различных типов индексов.
- Поддержка большого набора встроенных типов (включая символьные типы произвольной длины, геометрические примитивы и XML-данные), а также возможность создания пользовательских типов.
- Репликации и возможность контроля целостности данных, что немало- важно при работе в сложной слабоструктурированной информационной среде предприятия.

Г.7. Samcef

Пакет SAMCEF имеет модульную структуру, которая позволяет решать такие специфические задачи как: статический и динамический анализ; расчёт собственных частот; нелинейный статический анализ; нелинейный анализ теплообмена (с переходным или стационарным режимом, включая воздействие радиации); структурная оптимизация; динамика ротора; динамика ротора с расчётом переходных состояний; анализ упругих механизмов; нелинейная динамическая характеристика; анализ абляционных и пиролизных явлений; механизмы усталостного разрушения; анализ явления вязкопластичности.

Все расчётные модули связаны с единым графическим пре- и пост- процессором BACON, который считывает геометрические данные о конструкции из CAD систем с помощью различных интерфейсов и позволяет создать сетку конечных элементов.

Г.8. Digimat

Вычислительный комплекс Digimat представляет собой совокупность следующих модулей (решателей):

1. Модуль Digimat-MF предназначен для аналитического прогнозирования нелинейного поведения материалов. Модуль предназначен для гомогенизации и позволяет аналитически предсказывать нелинейные механические, тепловые и электрические макроскопические свойства многокомпонентных материалов, основываясь на свойствах компонентов и данных о микроструктуре материала (объёмная концентрация компонентов, размеры и ориентация волокон).
2. Модуль Digimat-FE предназначен для точного прогноза локального/глобального нелинейного поведения многокомпонентных материалов на основе применения метода конечных элементов (МКЭ) и реалистичных

Представительных Элементов Объёма (ПЭО; Representative Volume Element — RVE).

3. Модуль Digimat-CAE — это набор интерфейсов для программ конечноэлементного анализа и моделирования процессов литья пластмасс, задач механики композитов и композитных структур.

Г.9. Moldex3D

Программный комплекс Moldex3D состоит из следующих основных модулей:

- *Модуль Flow* — моделирование процесса заливки термопластических материалов под давлением.
- *Модуль Pack* — моделирование выдержки под давлением, осуществляющее путём решения уравнения Навье-Стокса.
- *Модуль Cool* — моделирование охлаждения детали и обнаружение проблем в системе охлаждения и их оптимизации.
- *Модуль Warp* — моделирование усадки и коробления, а также контролировать другие возможные дефекты.
- *Модуль Fiber* — учёт волокон в материале детали, моделирование трёхмерной ориентации волокон при заполнении формы.
- *Модуль RIM* — моделирования литья реактопластов.

Приложение Д

Программная реализация базового агента

Листинг Д.1. Реализация базового агента

```

1 # Импорт агентной библиотеки
2 import spade
3 from spade.bdi import *
4 # Импорт библиотеки ВСПТД
5
6 # Установление типа агента -- BDI
7 agent = BDIAgent
8 # Определение адреса СУА
9 agent = ADS('1bc29b36f623ba82aaf6724fd3b16718@'
10             'tps.ifmo.ru')
11 agent.setDebugToScreen()
12 # Определение убеждений агена, т.е. триплетов онтологии
13 # понятий с которыми он работает
14 agent.addBelieve(expr('$VALUE=0;'))
15
16 # Определение функции достижения цели, т.е. перевода
17 # триплетов цели в разряд триплетов факта
18 g = Goal(expr())
19
20 class Serv1(Service):
21     """Первый агентный сервис (абстрактный ресурс)
22     и триплет цели, который он может заполнить
23     """
24
25     def run(self):
26         """Запускает сервис
27         """
28
29         print 'Service 1 running'
30         self.addBelieve(expr('$VALUE=:1;'))
31
32
33 class Serv2(Service):
34     """Второй агентный сервис (абстрактный ресурс)
35     и триплет цели, который он может заполнить
36     """

```

```

37
38     def run(self):
39         """Запускает сервис
40         """
41
42         print 'Service 2 running'
43         self.addBelieve(expr('$VALUE=1;'))
44
45
46 # Инициализация сервисов
47 s1 = Serv1(P=expr('$TRP=0;'), Q=expr('$TRP=1;'))
48 s2 = Serv2(P=expr('$TRP=2;'), Q=expr('$TRP=2;'))
49
50 # Определение плана поведения (очередности и полезности)
51 p = Plan(P=expr('$VALUE=0;'), Q=expr('$VALUE=2;'))
52
53 # Связь агента с определенными выше сервисами
54 p.appendService(s1)
55 p.appendService(s2)
56
57 # Назначение ему плана и целей
58 agent.addPlan(p)
59 agent.addGoal(g)
60
61 # Инициализация агента
62 agent.start()
63
64 # Переход к основному исполнительному циклу, т. е.
65 # ожиданию сообщений
66 import time
67 try:
68     while True:
69         time.sleep(1)
70 except:
71     agent.stop()
72
73 sys.exit(0)

```

Приложение Е

Акты внедрения результатов диссертационной работы

УТВЕРЖДАЮ

Генеральный директор



Козлова С.П.

Сергей

документ

2012 г.

АКТ

внедрения результатов диссертационной работы

соискателя Афанасьева Максима Яковлевича

«Разработка и исследование многоагентной системы для решения задач

технологической подготовки производства»

в производственный процесс ООО «Завод по переработке пластмасс имени
«Комсомольской правды»

Комиссия в составе: генерального директора Козловой С.П., руководителя центра прототипирования Кудрявцева А.В., руководитель проектов Курышев Е.С., составила настоящий акт о том, что результаты диссертационной работы «Разработка и исследование многоагентной системы для решения задач технологической подготовки производства» Афанасьева М.Я., представленной на соискание ученой степени кандидата технических наук:

- методика построения информационного пространства с использованием технологий виртуализации и «облачных» вычислений;
- методика управления технологическими процессами с применением многоагентной системы;
- классификатор материалов, являющийся базовым компонентом системы управления данными об изделии,

стали методологической основой реализации проекта создания высокотехнологичного производства изделий из полимерных материалов в виде интегрированной распределенной инжиниринговой и производственной структуры на базе ООО «Завод по переработке пластмасс имени «Комсомольской правды».

Руководитель центра прототипирования

А.В.
Кудрявцев А.В.

Руководитель проектов

Е.С.
Курышев Е.С.

Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
**«Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики»**



«УТВЕРЖДАЮ»

Ректор

д.т.н., проф. Васильев В.Н.

«01» апреля 2012 г.

АКТ

об использовании результатов диссертационной работы соискателя
Афанасьева Максима Яковлевича: «Разработка и исследование многоагентной системы для решения
задач технологической подготовки производства»
при выполнении Государственного контракта № П571 от 5 сентября 2008 г.
в период с 2008 по 2010 гг. «Разработка и реализация модели непрерывного повышения
квалификации педагогических кадров российских технических вузов в системе
«вуз – инжиниринговый центр – организация» для Федерального агентства по образованию РФ

Комиссия в составе: председателя Ю.Л. Колесникова, членов комиссии: В.М.Мусалимова, Д.Д. Куликова, рассмотрев представленные материалы:

1. Диссертационную работу Афанасьева М.Я.
2. Отчет по выполнению Государственного контракта № П571 от 5 сентября 2008 г. «Разработка и реализация модели непрерывного повышения квалификации педагогических кадров российских технических вузов в системе „вуз–инжиниринговый центр–организация“» и установила, что:
 - Предложенные в диссертационной работе Афанасьева М.Я. методы многоагентной интеграции различных программных подсистем автоматизации технологической подготовки производства были использованы в качестве методов автоматизации решения производственных задач при функционировании и развитии инжинирингового центра НИУ ИТМО.
 - Разработанные методы управления данными и знаниями в процессе информационного обмена внутри автоматизированной системы технологической подготовки производства в условиях единого информационного пространства приборостроительного предприятия использованы в качестве базовых средств обеспечения системности управления информацией в институциональной среде «вуз – инжиниринговый центр – организация».
 - Предложенная методика моделирования средств информационного обеспечения, основанная на концепциях «облачных» вычисления и виртуальных рабочих мест, а также распределённый виртуальный испытательный стенд применяются для эффективного использования университетских лабораторий и промышленных предприятий в процессе непрерывного повышения квалификации педагогических кадров и переподготовки специалистов предприятий.

Председатель комиссии,
проректор по учебно-организационной
и административной работе, д.ф.-м.н., профессор

Ю.Л. Колесников

Члены комиссии:
д.т.н., профессор

В.М. Мусалимов

д.т.н., профессор

Д.Д. Куликов