МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА»
(САМАРСКИЙ УНИВЕРСИТЕТ)

*Е.В. СИМОНОВА*

# МОДЕЛИРОВАНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ

## *Часть I. Адаптивное управление сложными системами на основе мультиагентных технологий*

Рекомендовано редакционно-издательским советом федерального государственного автономного образовательного учреждения высшего образования «Самарский национальный исследовательский университет имени академика С.П. Королева» в качестве учебного пособия для обучающихся по основной образовательной программе высшего образования по направлению подготовки 09.04.01 Информатика и вычислительная техника

САМАРА
Издательство Самарского университета
2022

УДК 004.9(075)
ББК 32.97я7
С 375

Рецензенты: канд. техн. наук, доц. Л.С. Зеленко,
д-р техн. наук, проф. С.В. Смирнов

***Симонова, Елена Витальевна***

**С375 Моделирование информационных систем. Часть I. Адаптивное управление сложными системами на основе мультиагентных технологий** : учебное пособие / *Е.В. Симонова*. – Самара : Издательство Самарского университета, 2022. – 204 с. : с ил.

**ISBN 978-5-7883-1757-1 (ч. 1)**
**ISBN 978-5-7883-1759-5**

Учебное пособие предназначено для использования при изучении курса «Моделирование информационных систем». Пособие посвящено рассмотрению вопросов управления сложными адаптивными системами с использованием мультиагентных технологий. Описывается методология управления, методы и средства построения мультиагентных систем, типы архитектуры и стандарты в области построения мультиагентных систем, а также агентные платформы и языки программирования агентов

Предназначено для обучающихся по направлению подготовки 09.04.01 Информатика и вычислительная техника.

Подготовлено на кафедре информационных систем и технологий.

УДК 004.9(075)
ББК 32.97я7

ISBN 978-5-7883-1757-1 (ч. 1)
ISBN 978-5-7883-1759-5

# ОГЛАВЛЕНИЕ

# ВВЕДЕНИЕ

Разделы №1–№4 посвящены исследованию природы растущей сложности окружающего мира и предлагают новую методологию адаптивного управления сложными системами, применение которой позволяет на практике решать сложные задачи в самых различных сферах применения: от планирования и оптимизации ресурсов – до понимания текстов и извлечения знаний из данных.

Адаптивное управление сложными системами на практике требует создания распределенных систем, части (элементы) которых обладают высокой степенью автономности, и потому способны в любой момент времени быстро реагировать на непредвиденные события, самостоятельно принимать решения и согласованно взаимодействовать с себе подобными, чтобы достигать того уровня качества и эффективности функционирования, которые демонстрируют нам устойчиво развивающиеся окружающие нас разнообразные биологические и социальные системы.

В этой связи Разделы №5–№8 посвящены мультиагентным технологиям, позволяющим создавать интеллектуальные системы нового поколения, базирующиеся на принципах самоорганизации и эволюции, для решения сложных задач, где существующие классические математические модели или имеют очень ограниченное применение, или вовсе не применимы.

Разработанные на основе мультиагентных технологий новые модели, методы и алгоритмы решения сложных задач находят свое обобщение Разделе №4, описывающем конкретные алгоритмы реализации разработанного подхода в прикладных интеллектуальных системах «эмерджентного (вспыхивающего) интеллекта», отвечающего за высокую адаптивность создаваемых приложений.

В заключении обсуждаются перспективы развития данного направления, и показывается возможный облик будущих систем, появление которых ожидается уже в самое ближайшее время.

Пособие, в первую очередь, предназначено для студентов направления 09.04.01 Информатика и вычислительная техника. Учебное пособие может быть полезно широкому кругу читателей, включая инженеров и специалистов, ученых, аспирантов и студентов в самых различных сферах деятельности, на практике сталкивающихся с вызовами современной глобальной экономики и информационного общества и быстро растущей сложностью решаемых задач, а также всем тем, кто интересуется перспективами развития новых информационных технологий.

Мультиагентные системы призваны помогать в понимании природы растущей сложности окружающего мира и умении гибко адаптировать свое поведение «на лету» по событиям в реальном времени.

При подготовке Разделов №1–№6 использовались материалы из книги Ржевский Г.А., Скобелев П.О. Как управлять сложными системами? Мультиагентные технологии для создания сложных систем управления предприятиями. – Самара: Офорт, 2015. – 290 с.

# 1. ПОНЯТИЕ СЛОЖНОЙ СИСТЕМЫ

## 1.1. Век растущей сложности

Частое возникновение непредсказуемых событий, влияющих на наши планы, создаёт *неопределенность*.

До последнего времени мы не уделяли должного внимания происходящим вокруг нас неожиданным событиям. Полагаясь на учение Ньютона, мы привыкли к определенности. Мы ждем, что рано или поздно будет открыт новый фундаментальный закон природы, который объяснит все и позволит предсказывать будущее. Конечно, очень хорошо было бы строить планы и с уверенностью наблюдать за их воплощением. Неопределенность, и тем более, неожиданные события, напротив, пугают нас. Сталкиваясь с ними, мы задаемся многочисленными вопросами: «Что происходит? Кто за этим стоит? Что мне теперь делать?» К сожалению, готовых ответов на эти вопросы и, тем более, универсальных решений для неожиданно возникающих проблем часто не существует.

Перечислим примеры только некоторых глобальных проблем, которые мы не можем решить сегодня:

- глобальное потепление, механизм которого мы не понимаем;
- бедность, неравенство и преступность – до сих пор все наши попытки искоренить их терпели неудачу;
- увеличение пропасти между богатыми и бедными вследствие глобализации;
- другие.

Все упомянутые выше проблемы являются *сложными*, и это не значит, что они были созданы намеренно или возникли вследствие

чьей-то ошибки; они появились в результате миллионов взаимодействий, как мы далее будем говорить, *эмерджентно* – как непредвиденные последствия бесчисленных решений, которые мы сами принимаем каждый день.

Такие проблемы могут быть решены только с применением новой методологии, основанной на науке о сложности. Основы этой науки не так давно были заложены бельгийским ученым проф. И. Пригожиным [1–2], русским по происхождению, получившим Нобелевскую премию по химии в 1978 г. за исследование процессов самоорганизации и явления автокаталитических реакций в растворах органических жидкостей. Эти работы были развиты исследователями из Института Санта-Фе в США, в частности, проф. Кауфманом [3] и проф. Холландом [4–5], продолжившими изучение феноменологии самоорганизующихся систем в окружающей нас неживой и живой природе, а также в социально-экономических системах.

Однако, мы упорно продолжаем решать проблемы теми же методами и средствами, как делали это в прошлом, совершенно упуская из вида тот факт, что новые задачи требуют и новых подходов.

Вот некоторые примеры, показывающие, к чему приводит консервативный образ мышления:

- мы по-прежнему создаем большие компьютерные системы, которые, как правило, разрабатываются с задержками и превышением бюджета, и многие из них работают совсем не так, как задумывались;
- мы создаем большие организации и объединяем их в еще более громоздкие корпорации и концерны с вертикальным управлением, уповая на мудрость и прозорливость, а также бесконечный ресурс здоровья, сил и времени их руководителей, которым платим огромные зарплаты, несмотря на тот всем очевидный факт, что небольшие организации являются

более открытыми, гибкими и эффективными, и, в конечном счете, более прибыльными, ориентированными на заказчиков, и комфортными для своих сотрудников.

- мы постоянно увеличиваем масштабы научных и учебных учреждений, хотя известно, что небольшие образовательные школы предоставляют более качественные услуги и в академическом плане являются более успешными.

Возможно, стоит остановиться на мгновение и поразмыслить над прозорливым утверждением Эйнштейна: «*Ты никогда не решишь проблему, если будешь думать так же, как те, кто её создали».*

Важно понимать, что большие системы были экономически эффективны в стабильном и предсказуемом мире, где критическим фактором успеха была *экономия на масштабе* (снижение затрат на единицу продукции при укрупнении производства.). Недавние изменения, в частности, нарастающая сложность рынков и нашей социальной среды, выявили низкую эффективность крупных систем, чрезмерно громоздких и жестких для адаптации к частым изменениям.

Стоит лишь кратко взглянуть на процессы, идущие в современном обществе, и мы сразу увидим, как их сложность постоянно растёт, порождая проблемы, решение которых требует нового образа мышления, а именно, понимания *природы сложных систем*.

Часто цитируемое утверждение выдающегося физика Стивена Хокинга, сделанное им в конце 20-го века, лучше всего иллюстрирует то, как важно уметь управлять сложными системами: *«Я думаю, следующий век будет веком сложных систем»*.

Рассмотрим, как наука о сложных системах может помочь в решении сложных проблем.

## 1.2. Определение сложной системы

Казалось бы, мы интуитивно понимаем, что подразумевается под сложностью.

Одной из важных характеристик сложных систем является их нелинейность, когда при малом входном воздействии система неожиданно реагирует в ответ большими изменениями на выходе, равно как и наоборот, причем возможно, с существенными задержками, и даже колебаниями или другими неожиданными явлениями, часто ставящими в тупик наблюдателя.

Мы страшимся нелинейности не случайно – ведь с эпохи Возрождения и до 2-й половины XX века в нашей науке царило «линейное мышление», в котором считается, что результат суммарного воздействия на систему обычно равен сумме воздействий. Известные законы во многих областях линейны, например, Ньютона (механика), Ома (электричество), Гука (теория упругости), Мальтуса (рост популяций), Максвелла (электродинамика) и другие. Из математических свойств линейных систем следует однозначный детерминизм: следствие однозначно определяется причиной, существует одно правильное решение.

Линейная наука с успехом изучает устойчивые процессы, воспроизводимые в эксперименте, но ведь далеко не все явления природы линейны, устойчивы и воспроизводимы, например, живые системы (от клетки – до человечества), зарождение атмосферных вихрей, образование галактик и пятен планктона в океане, творческая деятельность и другие подобные процессы.

Можно ли написать уравнения, описывающие работу производственного цеха машиностроительного предприятия или цепочки поставок товаров от фабрик через склады в магазины и другие похожие системы?

К сожалению, феномену сложности нет никакого точного определения. Многие ученые, работающие в науке о сложных системах,

разочарованы этим фактом, но, с другой стороны, так и должно быть: мы никогда не сформулируем точного определения сложности, потому что сложность неоднозначна по своей природе.

По той же причине у нас нет точных определений для таких понятий как знание, разум, эмоции или творчество. Но отсутствие точного определения совершенно не препятствует тому, чтобы мы эффективно использовали эти понятия.

Сложность можно определить следующим образом:

***Сложность*** *является свойством открытой системы, которая состоит из большого числа разнообразных, частично автономных, активно взаимодействующих элементов, называемых агентами. Сложная система не имеет централизованного управления, а ее поведение определяется взаимодействием агентов, и поэтому, не будучи хаотичным, является неопределенным (недетерминированным), поскольку в каждой ситуации определяется свободой выбора агентов и зависит от принимаемых ими решений*.

Ключевыми характеристиками сложной системы являются:

- открытость, предопределяющая активное взаимодействие со средой;
- разнообразие, частичная автономность и взаимосвязь агентов;
- отсутствие централизованного управления;
- эмерджентность поведения – решение системы возникает спонтанно, в непредвиденный момент времени, как результат цепочки взаимодействий агентов для нахождения согласия (консенсуса) между ними.

В нашем понимании, сложная система – это самоорганизующаяся система, построенная на поиске и поддержании баланса (гармонии) интересов агентов.

Интеллект такой системы проявляет себя цепочками взаимодействий и согласованных решений для достижения консенсуса

агентов, которые развиваются подобно автокаталитическим реакциям в растворах органических жидкостей в терминах нелинейной термодинамики И. Пригожина.

Вся феноменология поведения рассматриваемых систем связана с «устойчивыми неравновесиями» или «неустойчивыми равновесиями» в согласии агентов.

Не правда ли, слово «гармония» для технических систем звучит как-то неожиданно?

## 1.3. Виды сложных систем

В английском языке существуют два похожих слова, обозначающих сложность, но имеющих совершенно разные понимания или смысловые значения.

Первое слово *Complex* – по смыслу переводится как «сложный в своем поведении», а второе – *Complicated,* переводится как «состоящий из многих частей». Для того, чтобы более точно выделить класс интересующих нас сложных систем и определить его главные отличия, мы предлагаем говорить о классах «сложных адаптивных» и «сложных неадаптивных» систем.

***Сложная неадаптивная система*** (complicated system) – состоит из множества частей, каждая из которых, в свою очередь, возможно, обладает множеством своих составных частей, т.е. имеется много уровней вложенности, однако, состав элементов и связи между ними не меняются динамически в результате взаимодействий. Располагая достаточным запасом времени, мы можем разобраться в том, как работает такая система, для чего обычно достаточно разделить ее на вложенные подсистемы, что обычно с успехом и реализуется в системном анализе. Если подсистемы также являются составными, мы можем далее делить каждую из них на подсистемы следующих уровней и продолжать такое деление до

тех пор, пока не достигнем уровня, на котором поведение системы станет полностью однозначным и понятным исследователю.

Поведение такой сложной системы является полностью предсказуемым (детерминированным), даже если для описания модели ее работы требуются сложнейшие системы дифференциальных или алгебраических уравнений. Типичными системами такого рода являются самолет, реактивный двигатель, компьютер, алгоритм, технологическая линия (конвейер) массового производства или иерархически организованная компания, действующая по жестким бизнес-процессам.

Такие системы обычно строятся «сверху-вниз», реализуя некоторый общий замысел и план действий своего создателя.

***Сложная адаптивная система*** (complex system) – состоит из автономных (самостоятельных) элементов (агентов), вступающих в связи или разрывающих эти связи в результате взаимодействий и собственных решений, предпочтений и ограничений. Иными словами, целостность сложной системы формируется «снизу-вверх» путем самоорганизации образующих ее элементов, что рассматривается как более высокая ступень в организации систем, предопределяя ее открытость, гибкость и эффективность, надежность и живучесть.

Невозможно разделить сложную систему способом, как это делается обычно в системном анализе, когда система разделяется на подсистемы и независимо анализируется поведение ее составных частей, поскольку разрыв связей между элементами сложной системы неминуемо повлечет изменение моделей и методов работы ее элементов, и, как следствие, поведения всей системы в целом.

Поведение сложной адаптивной системы трудно предсказать в связи с недетерминизмом в том смысле, что оно не диктуется одним заданным сверху общим «замыслом», алгоритмом или бизнес-процессом (пусть даже сколько угодно составным и ветвящимся),

а определяется взаимодействием ее элементов (агентов), вступающим в связи или разрывающими эти связи по собственной инициативе, с учетом изменений в среде или в отношениях между агентами, но без какого-либо принуждения со стороны.

На рис. 1 представлен пример изменения конфигурации открытой системы за счет присоединения еще одного нового узла в ходе взаимодействий, например, нового заказа или ресурса.

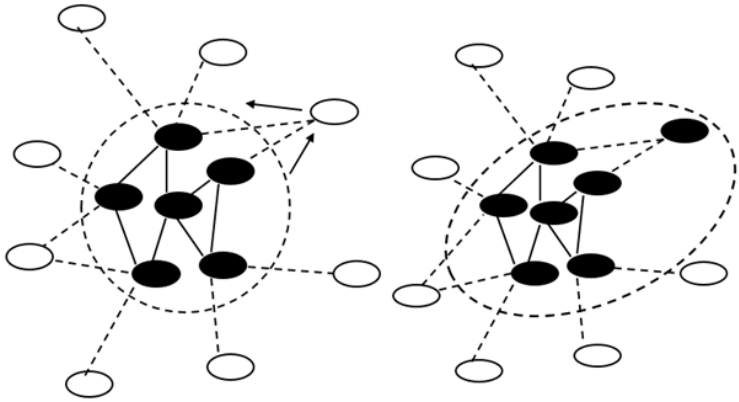



Рис. 1. Изменения состояния системы
за счет присоединения еще одного узла в ходе взаимодействий:
а – момент времени Т1; б – момент времени Т1+1

Примерами сложных адаптивных систем в окружающем нас мире являются экология, климат, мировой рынок, семья, команда, рой пчел, человеческий мозг, человеческое общество, эпидемии, террористические сети, дорожное движение, жизненный цикл самолета или автомобиля.

Примеры сложных адаптивных и неадаптивных систем [6], характеризующие их значимые свойства, приведены в табл. 1.

Более близким к технике примером может служить «умный автомобиль» будущего, который через датчики в шинах может почувствовать изменение сцепления с трассой при сходе с шоссе на грунтовку и предложить раме поднять подвеску, которая, в свою очередь, попросит двигатель сбросить обороты, и каждый из этих элементов самостоятельно, но согласованно примет решения, опираясь на результаты анализа развития текущей ситуации.

Таблица 1. **Виды сложных систем**

| **Сложные адаптивные системы** | **Сложные неадаптивные системы** |
|---|---|
| Мировой рынок, основанный на интернет-технологиях | Централизованная плановая экономика |
| Гибкая производственная система | Конвейер массового производства |
| Команда по междисциплинарному проекту | Начальник отдела и подчиненные |
| Адаптивная система планирования | Пакетная система планирования |
| Мультиагентная система | Большая монолитная компьютерная программа |
| Жизненный цикл самолета | Самолет |

Другой пример – «умный холодильник», который «на лету» сможет начать взаимодействовать с пакетом молока, который ставится в холодильник, что приведет к такому изменению режима работы холодильника, который обеспечит наилучшую сохранность молока, но с учетом интересов и всех тех продуктов, которые уже хранятся в холодильнике, а еще принимая во внимание постоянно уточняемый и пересчитываемый прогноз потребления продуктов хозяином, изменения в прогнозе погоды и ряд других факторов.

Нельзя не отметить и новую тенденцию тотального появления встроенных «умных» (smart) систем и образуемых ими систем «повсеместного интеллекта» (ambient intelligence), каждая из которых

по определению должна иметь возможности восприятия, принятия решений и коммуникации с другими.

## 1.4. Сложность и неопределенность

Неопределенность можно использовать в качестве одного из важных параметров, по которому можно отличать сложные адаптивные системы от неадаптивных систем, как это показано в табл. 2.

Таблица 2. **Сложность в сравнении детерминированности и недетерминированности**

| Недетерминированные системы | Сложные адаптивные системы | Детерминированные системы |
|---|---|---|
| Неопределенность = 1 (полная неопределенность – хаос, беспорядочность) | 0 < Неопределенность < 1 (частичная определенность – частичный порядок) | Неопределенность = 0 (полная определенность – жесткий порядок) |
| Элементы системы полностью автономны и никак не связаны между собой (каждый принимает решения сам по себе) | Элементы системы (агенты) частично автономны и частично связаны между собой (имеют свободу выбора) | Элементы системы не автономны и жестко связаны между собой (например, следуют приказам и не имеют свободы выбора) |
| Неорганизованность (хаотичность и спонтанность) | Самоорганизация и эволюция (частичный порядок и гибкий план, работа по ситуации) | Высокая организованность (полный порядок и четкий план, ситуация не влияет) |
| Совершенно непредсказуемое поведение | Согласованное эмерджентное поведение (взаимодействие для выработки и согласования решений) | Полностью предсказуемое поведение, жесткий алгоритм |

Термин «детерминированный» здесь подразумевает, что неопределенность системы равна нулю, в то время как термин «случайный» означает, что неопределенность системы равна единице.

Все сложные адаптивные системы имеют показатель неопределенности между нулем и единицей.

Табл. 2 показывает связь между сложностью и неопределенностью (недетерминизмом): неопределенность является следствием сложности, с нарастанием которой она увеличивается.

Системы невысокой сложности имеют неопределенность, стремящуюся к нулю, и их поведение мало отличается от поведения детерминированных систем.

Системы с повышенной сложностью, неопределенность которых стремится к единице, балансируют на грани хаоса. Поведение таких систем характеризуется необычными свойствами, обусловленными наличием большой нелинейности, например, даже при небольшом изменении на входе системы могут возникать большие изменения на ее выходе («эффект бабочки»), может наблюдаться неожиданная смена траектории в линии поведения (бифуркация), возникает эффект резонанса и многие другие.

Подобное поведение будет присуще любым интеллектуальным системам, построенным как сложные адаптивные системы.

## 1.5. Семь критериев сложности

В результате проведенных исследований были выделены семь ключевых свойств, которые отличают сложные адаптивные системы от детерминированных систем [6]: *связность, автономность, эмерджентность, неравновесность, нелинейность, самоорганизация и эволюция*.

Мировой рынок, все больше использующий Интернет-технологии, является, возможно, самым интересным примером сложной

адаптивной системы [6-8], который может быть наглядно использован в качестве иллюстрации указанных свойств.

### *1.5.1. Связность*

Сложная адаптивная система состоит из большого количества разнообразных элементов (агентов), которые связываются в ходе взаимодействия друг с другом. При этом связи между агентами могут различаться еще и по своей силе, отражающей тот факт, насколько агенты подходят друг другу.

Более высокая степень связности системы, отражающая насколько одни агенты могут связываться с другими агентами, в сочетании с более слабыми связями, которые могут быть легко разорваны и сформированы заново, подразумевают более высокую сложность всей системы. Сложная система больше походит на размытое облако, чем на устоявшуюся жесткую структуру: у нее нет постоянной жесткой конфигурации, как нет и четких границ между самой системой и ее средой.

Представьте себе мировой рынок, который состоит из большого количества поставщиков, потребителей, производителей, инвесторов, банкиров и других агентов, связанных друг с другом торговыми отношениями. Степень связности, а следом и сложность мирового рынка, многократно увеличились в связи с широким использованием Интернета, где каждый может легко найти каждого, кто ему интересен или полезен. Сила связей между участниками рынка является весьма изменчивой: некоторые являются почти постоянными, как долгосрочные отношения поставщиков и потребителей, но многие связи очень слабы и постоянно меняются, например, некоторые потребители постоянно меняют свои заказы и своих поставщиков.

Другие примеры: виртуальные организации, состоящие из совместителей, работающих в других организациях, дружба, в которой связи усиливаются или ослабевают с течением времени,

спортивные навыки, требующие высокой координации движений, теряемые с течением времени или восстанавливаемые путем тренировок и т.д.

### *1.5.2. Автономность*

Агенты управляются не централизованно; они имеют определенную степень автономности, хотя их поведение всегда подчинено определенным законам, правилам или нормам, принятым для системы.

Высокая автономность агентов и их большая свобода принимать разнообразные решения подразумевает более высокую сложность системы.

У мирового рынка нет централизованной системы управления, и все же «свободный рынок» не совсем свободен; участники рынка подчиняются национальным и международным законам, регламентам, установленным нормам поведения и прочим формам влияния на принятие решений. Степень автономности участников рынка влияет на его сложность, однако может быть скорректирована разными предписаниями, позволяющими увеличивать или снижать сложность, когда это необходимо.

Другой пример повышения уровня автономности агентов – компания, построенная как набор самостоятельных бизнес-центров, в которой руководители этих центров могут самостоятельно вести переговоры и заключать контракты на основе соглашений о разделе прибыли, принятых в компании.

### *1.5.3. Эмерджентность*

Общее поведение сложных систем формируется путем взаимодействия их агентов, что в свою очередь ограничивает поведение агентов.

Известный принцип Кауфмана гласит, что в самоорганизующихся системах локальные взаимодействия формируют глобальные структуры, которые влияют на порождающие их элементы путем отрицательных (ослабляющих) или положительных (усиливающих) обратных связей, что, как известно, может приводить как к «раскачке» системы и появлению самоускоряющихся (автокаталитических) процессов и даже резонансных явлений, так и к торможению всех процессов и возврату системы в состояние равновесия.

Эмерджентное поведение в системе формируется в ходе взаимодействий агентов и потому непредсказуемо, но оно является отнюдь не случайным. Оно обычно следует некоторым поведенческим шаблонам и ведет к установлению нового порядка в системе (более устойчивого равновесия), но всегда зависит от ситуации, как правило, чем более сложна система, тем более она чувствительна к ситуации, тем большая перестройка может осуществляться в ходе даже самых малых воздействий.

Эмерджентные свойства сложной системы не присутствуют в составляющих ее агентах и трудны для изучения. Простейший пример эмерджентного поведения связан с тем, как образуется и распространяется пробка на дороге – пробка движется против движения машин, влияя на принятие решений водителями. Чтобы понимать возникновение этого феномена нужно уметь анализировать сотни и тысячи решений, принимаемых водителями, и для управления этим феноменом нужно уметь влиять на принятие решений водителями.

На мировом рынке глобальное распределение поставок формируется в результате местных торговых операций. Такое распределение непредсказуемо, однако, и в нем можно обнаружить определенные шаблоны, например, экономические циклы с фазами подъема, спада, рецессии и т.д.

На примере компаний эмерджентное поведение часто демонстрируют междисциплинарные команды по проектам, члены которых оказываются способными высказывать ценные идеи, которые сотрудникам невозможно было бы получить по отдельности, изолированно работая в своих отделах.

### *1.5.4. Неравновесность*

Во внешней среде сложной системы или внутри нее часто происходят события, нарушающие состояние равновесия системы, например, приход нового важного заказа или выход из строя важного ресурса.

В реальной жизни у сложной системы может не быть времени вернуться в состояние равновесия в период времени между двумя событиями, и потому система вынуждена работать вдали от равновесия. То же обычно можно сказать и о каждом агенте, например, агент заказа на перевозку может найти грузовик, который лишь частично соответствует требованиям заказа (приезжает позже срока), но у агентов уже нет времени продолжать поиск новых вариантов, необходимо принимать окончательное решение и выезжать к заказчику.

Рынки всегда функционируют далеко от равновесия, ведь все время заключаются новые торговые сделки, и даже сделки, заключенные ранее, меняются с высокой скоростью. В результате у рынка не хватает времени на возвращение к равновесию между двумя событиями.

При определенных условиях в таких системах возникают колебания (осцилляции), связанные с циклическим переходом от одного состояния равновесия – к другому.

Этот феномен также наблюдается в жизни компаний, например, растет недовольство сотрудника, и он увольняется из компа-

нии, а на смену ему приходят более продуктивные люди, которые превращают убыточное направление в прибыльный бизнес, и тогда ушедший сотрудник возвращается.

Заметим, что даже в существующей связи удовлетворенность партнером с каждой стороны может быть совершенно различной, что означает неравновесие и в силе связи.

Понимание силы связей между элементами может помочь прогнозировать «узкие места» и в будущем управлять поведением сложной системы.

### *1.5.5. Нелинейность*

Нелинейность означает, что результат поведения сложной системы не сводится к результатам поведений ее отдельных частей (агентов).

Действительно, если агенты постоянно взаимодействуют, обращаясь друг к другу с предложениями или меняя что-то в общей для всех ситуации, то это немедленно приводит к изменению условий, влияющих на принятие решений другими агентами.

Как и в обычной жизни, все часто начинается с незначительных событий (возмущений), последствия которых могут накапливаться и в какой-то момент приводить к резкому преодолению порога, радикально меняющего все поведение системы.

Характерным примером являются автокаталитические реакции в самоорганизующихся системах, открытые И. Пригожиным в сложных растворах органических жидкостей. Для случая технических систем взаимодействия в небольшой группе агентов могут также приводить к ситуации, когда все больше агентов вынуждены менять решения – процесс изменения решений может развиваться с нарастающей силой, кардинально меняя поведение системы в целом.

Триггером для автокаталитической реакции могут быть самые малые события и даже совершенно случайные флуктуации, которые впоследствии перерастают в критические события («эффект бабочки»). Известно много примеров и в технике, когда наличие многих незначительных, но связанных изменений в течение долгого времени создает в итоге совершенно неожиданный критический сбой – катастрофу, имеющую далеко идущие последствия. Момент, когда незначительные изменения вдруг превращаются в критическое событие, влекущее полную перестройку связей внутри самоорганизующейся системы, называют переломным моментом.

Финансовый кризис 2008 года – показательный пример критического события. В случае увеличения сложности мирового рынка, использующего Интернет-технологии, критические события могут стать все более частыми и опасными.

Так и в жизни компаний – появление важного нового клиента и большого заказа могут вызвать лавинообразную цепную реакцию, ведущую к скачку в развитии компании.

### *1.5.6. Самоорганизация*

Сложные адаптивные системы *самоорганизуются*, т.е. сами изменяют свою структуру и поведение, чтобы устранить или сократить влияние событий, нарушающих их состояние, с большей или меньшей степенью адаптивности и устойчивости.

Кроме такой реактивной самоорганизации, у агентов, образующих систему, может быть способность к проактивной самоорганизации, например, через внутренние попытки повысить эффективность уже принятого решения.

Такая способность может позволить агентам совместными усилиями остановить процесс постепенной деградации системы и роста отказов, или даже проявить созидательные возможности

«эмерджентного разума», позволяющего сообща находить принципиально новые решения.

Важной возможностью для усиления самоорганизации должна со временем стать способность агентов получать новые знания из своего опыта.

Такие способности очень характерны для агентов мирового рынка, когда участники вносят поправки к законам о торговых операциях для повышения эффективности торговли – новые законы (глобальные структуры) начинают влиять на локальные взаимодействия, принятие решений и поведение тех, кто их порождает.

Аналогичным образом, в компаниях создаются инициируемые снизу рабочие группы для решения выявленных важных проблем, действия которых далее оказываются часто более эффективными для организации, чем приказы сверху.

### *1.5.7. Эволюция*

Если мы определяем окружающую среду системы как набор всех других систем, с которыми эта система взаимодействует, тогда мы можем утверждать, что сложные системы открыты: они адаптируются к своей среде, но и, в свою очередь, изменяют её.

Процесс такого взаимодействия обычно необратим, как оказывается, невозможно «откатить» назад любое решение, полученное в самоорганизующейся системе, поскольку исходные условия, при которых оно было получено, на практике уже нельзя восстановить.

Другими словами, сложные системы и их среды развиваются параллельно и асинхронно, оказывая влияние друг на друга, т.е. *коэволюционируют*.

Совместное развитие сложных систем мировой экономики, технологий и общества будет для примера показано в табл. 3 ниже.

## 1.6. Отрицательные и положительные аспекты сложности

Необходимо создавать модели, методы и средства управления сложными системами, которые позволят не только извлечь пользу из сложности, но и минимизировать или полностью нейтрализовать отрицательные особенности сложных адаптивных систем.

В большинстве случаев, самыми важными и трудными вызовами сложных систем является необходимость быстро реагировать на частые и плохо предсказуемые события, нарушающие планы (равновесие), а также нелинейность системы в реакции на эти события, затрудняющая какое-либо прогнозирование результатов.

Примеры возмущающих событий: постоянные изменения спроса и предложения, нарушения сроков поставок, изменение цен на услуги в результате повышения конкуренции, изменения в ранее заключенных соглашениях, заказах и обязательствах; задержки и отмены поставок товаров и услуг; нарушения правил безопасности, мошенничество и хакерство.

Часто возникающие и непредсказуемые события препятствуют целенаправленному планированию и успехам в реализации планов.

Частные компании и государственные организации встречаются с большими трудностями в управлении распределением ресурсов, так как спрос и предложение становятся все более индивидуальными, изменчивыми и трудно прогнозируемыми.

Еще больший вред наносят непредвиденные события, вызванные нелинейностью сложных систем, начиная с внезапной потери важного клиента и закачивая мировым финансовым кризисом. Постепенный отказ системы вследствие накопления множества незначительных ошибок в обслуживании и управлении приводит к катастрофам, что показывает важность понимания природы сложных систем и непригодности старых методов управления.

С другой стороны, самоорганизация и эволюция в последнее время все чаще считаются положительными сторонами сложных систем.

Еще одной важной полезной особенностью сложных систем является разнообразие агентов, которое гарантирует надежность и живучесть всей системы. На планете Земля живет приблизительно семь миллиардов человек, и каждый из нас уникален, отличен от других. Будем надеяться, что никакая атака, никакие эпидемии не смогут уничтожить нас всех разом – разнообразие обеспечит сохранение нашего вида.

Примером чрезвычайно сложной системы, способность которой к быстрой самоорганизации обеспечивает наше здоровье, а часто, и сохранение самой жизни в условиях постоянных внешних атак, является иммунная система человека.

Другим прекрасным примером является Интернет – выход из строя одного из узлов или даже целой группы не нарушает работоспособность системы в целом.

Конечно, понятия «положительный» и «отрицательный» в отношении систем с высокой самоорганизацией весьма относительны. Высокая дееспособность вирусов к распространению эпидемии или террористической сети к организации массовых взрывов является положительным аспектом для вспышки болезни и для террористов соответственно, но чрезвычайно отрицательным фактором для их жертв и общества в целом.

Итак, вывод состоит в том, что *сложность системы влечет за собой неопределенность, а неопределенность (недетерминизм) порождает новые возможности, на которые могут чутко реагировать другие сложные системы, создавая новые условия и планы для роста*.

# 2. РАЗВИТИЕ СЛОЖНЫХ СИСТЕМ

## 2.1. Эволюция сложных систем

Откуда же появляется сложность и почему в настоящее время так быстро растет число все более сложных проблем?

Существуют убедительные доказательства того, что одновременно с развитием человечества неуклонно растет сложность различных сфер нашей жизни – социальной, политической, культурной, экономической и других, которые все больше переплетаются и ко-эволюционируют между собой. Этот процесс *необратим* и проявляется в увеличении *разнообразия* и *взаимной зависимости* возникающих новых структур и видов деятельности, а также в увеличении *неопределенности* в процессах нашей деятельности, требующих новых знаний и творческого подхода.

*Естественный отбор, который управляет эволюцией, благоприятствует развитию способностей к адаптации и разнообразию нашей жизни (а значит, и ее сложности), как выигрышным свойствам сложных систем*.

Нужно использовать свойства сложных систем, такие как самоорганизация и эволюция, для повышения эффективности деятельности организаций, технических и программных продуктов и даже нас самих.

Мы неоднократно видели в своей практике, что лучшей стратегией при столкновении со сложными проблемами, является управление ситуацией за счет изменения степени автономности отдельных узлов или регулирования силы связей между ними, что

часто позволяет быстро приспособиться и устранить, или, по крайней мере, сократить последствия событий, совершенно неподвластных нам.

Представьте себе две организации, работающие на передовом рубеже инноваций. При этом одна из них ведет очень закрытую деятельность, а другая открыта и развивает многочисленные связи с университетами, партнерами, ассоциациями пользователей, ученым сообществом и т.д. У кого больше шансов выжить и иметь более интенсивное развитие? Ответ, казалось бы, во многом очевиден, но на самом деле зависит от ситуации. Если рынок хорошо предсказуем, то первая организация, не тратя время и силы на многочисленные связи, может быстрее завершить прорывной продукт и вывести его на рынок. Но если неопределенность на рынке продукции и труда высока, что обычно и бывает, то выиграет вторая организация, которая может победить за счет кооперации в поиске клиентов и подготовке кадров.

Другой пример – когда-то при разработке одной из первых наших платформ для создания мультиагентных систем мы были вынуждены запустить сразу два конкурирующих проекта разработки вместо одного, поскольку видели несколько путей создания будущих систем и еще не имели никаких аргументов в пользу единственного выбора любого из них, что позже дало нам гораздо большую эффективность в реализации проектов и позволило создать коллектив разработчиков, хорошо владевших разнообразными технологиями.

Оказывается, положительный эффект может быть связан не только с обострением внутренней конкуренции, но и синергетическим эффектом при переходе от конкуренции – к кооперации.

## 2.2. Совместное развитие технологии, экономики и общества

Общество *коэволюционирует* с технологиями создания материальных благ.

Аграрное общество, в котором ключевым ресурсом была земля, а все люди были полностью заняты в сельском хозяйстве, было вытеснено индустриальным, где ключевым ресурсом стал капитал, и большинство людей начали работать на промышленном производстве товаров

Теперь мы находимся на стадии перехода от промышленного общества – к информационному обществу, в котором ключевым ресурсом являются знания, и где большинство людей работает в сфере услуг, основанных на знаниях (то есть связанных с обработкой информации), а не на производстве материальных товаров.

Совместное развитие общества, экономики и технологий, иллюстрирующее наши рассуждения в более глобальном историческом масштабе, показано в табл. 3. Средства, направленные на улучшение качества жизни, изменяют экономическую деятельность, которая, в свою очередь, меняет и само общество.

Однако эти средства становятся доступными, только если общество решает вложить в них капитал и целенаправленно использовать их для своего развития.

Важно отметить, что с развитием экономической системы изменяются и ключевые факторы экономического успеха. Экономия на масштабе в гигантских фабриках, бесспорно, ключевой фактор успеха в промышленной экономике прошлого века, становится все менее важным фактором при растущей сложности экономики знаний.

Мы утверждаем, что новым ключевым фактором успеха является адаптивность как способность быстро и точно отвечать на непредсказуемые изменения на рынке.

Таблица 3. **Совместное развитие общества, экономики и технологии**

| Этапы социального развития | Ключевые ресурсы | Средства распределения | Масштаб | Факторы успеха |
|---|---|---|---|---|
| Аграрное общество<br>Аграрная экономика<br>Орудия обработки земли | Земля | Сельские дороги | Местный | Эффективность |
| Индустриальное общество<br>Индустриальная экономика<br>Технологии массового фабричного производства | Капитал | Автомагистрали и железные дороги | Региональный и национальный | Экономия на масштабе (тиражирование) |
| Информационное общество<br>Экономика Знаний[1]<br>Цифровые технологии | Знания | Цифровые сети | Мировой | Адаптивность |

## 2.3. Сложность и информационное общество

### *2.3.1. Цифровые технологии как причина роста социальной сложности*

Сложность социальных систем растет пошагово. Текущий переход от индустриального общества к информационному, который начался после окончания Второй мировой войны с изобретения компьютеров, известен

[1]Экономика знаний – высший этап развития постиндустриальной экономики и инновационной экономики.

резким увеличением сложности, вызванной широким распространением цифровых технологий, которые предоставляют беспрецедентные новые возможности для социального взаимодействия.

Во время перехода от аграрного общества к индустриальному увеличение сложности было небольшим, хотя, в некоторых аспектах, весьма существенным.

Поспешная миграция жителей сельской местности в города в поисках новых возможностей трудоустройства вызвала беспорядки и, в то же время, расширила возможности социального взаимодействия благодаря увеличению плотности населения в городах и, таким образом, в итоге увеличила сложность социальной жизни. Жесткий традиционный общественный строй, основанный на земельной собственности, через волну изменений был вытеснен новым общественным порядком, в котором основной ценностью стал капитал, чтобы через новую волну преобразований смениться новым информационным обществом, в котором главную ценность приобретают знания, которые постепенно становятся рычагом власти, заменяя собой насилие и деньги как действующие факторы предыдущих формаций.

Во время текущего периода цифровые технологии значительно повысили активность социального взаимодействия (социальную плотность), избавив население от необходимости перемещения. Теперь мы, находясь на удалении друг от друга, можем формировать общности в группах по интересам по всему миру, и территории и расстояния уже не имеют решающего значения. Благодаря цифровым технологиям все участники информационного общества взаимодействуют более активно, быстрее, чаще и с большим числом других людей, чем когда-либо прежде в истории, и, конечно, высокая степень взаимодействия и рост скорости связи обусловили новый высокий уровень сложности современного общества (рис. 2).
На рис. 3 кратко представлена эволюция основных информационных технологий, перспективы которых связываются с Интернетом вещей, в котором каждая из наших обычных вещей будет иметь непрерывную связь с производителем, воспринимать окружающую

среду, принимать решение и коммуницировать с себе подобными. По оценкам Института McKinsey Global Insitute, Интернет вещей обеспечит рост производительности труда до 5% в год уже на протяжении следующих 10 лет, что совокупно даст примерно 900 млрд. долларов США новых доходов и экономии лишь для производственного сектора.

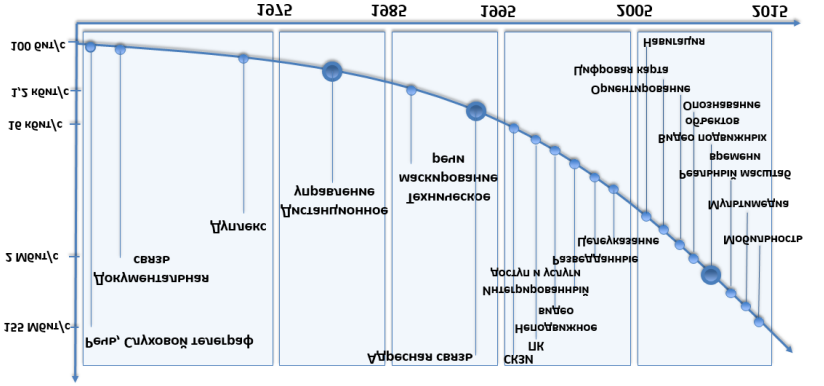


Рис. 2. Динамика роста возможностей
информационно-телекоммуникационных технологий

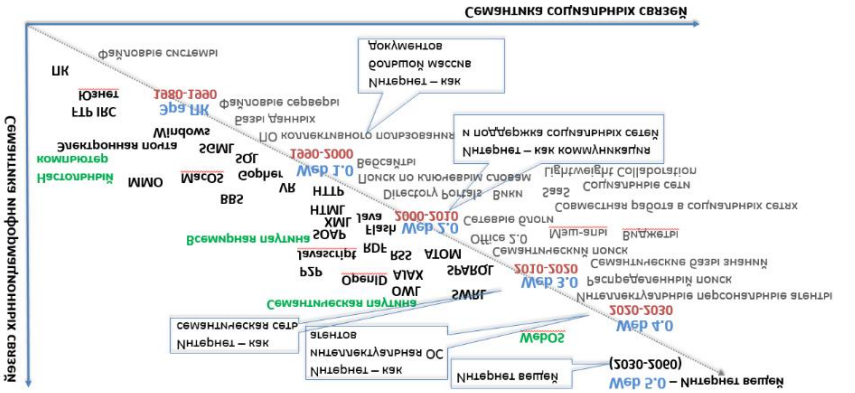


Рис. 3. Развитие ИТ технологий

Эксперты сходятся во мнении, что потеря темпа освоения новых информационно-коммуникационных технологий грозит человечеству резкой потерей продуктивности работы в командах людей, где бы они ни работали: от крупных государственных корпораций – до малого и среднего бизнеса (рис. 4).


Рис. 4. Будущее производительности

### *2.3.2. Переход от капитала к знаниям как ключевому ресурсу власти и успешной экономической деятельности*

В промышленной экономике за деньги можно было купить любые знания, необходимые для бизнеса.

В информационной экономике, наоборот, знания о том, как решать сложные проблемы, могут привлечь любые инвестиции, которые требуются для начала и развития успешной экономической деятельности.

Основоположники информационной экономики и основатели таких компаний, как Microsoft, Apple, Google, Amazon и Facebook, работа которых основана на применении знаний и информации, являются новой экономическая элитой.

Питер Друкер первым определил знания, как ключевой ресурс для бизнеса нового мира, возникшего после окончания Второй мировой войны, и первым ввел термин «работник умственного труда» («knowledge worker»).

В то время, как сложность нашей социальной среды увеличивается, мы нуждаемся в новых знаниях для решения новых сложных проблем – сильный стимул для вложений в образование и выращивание инноваторов. Для этого необходимы новые цифровые технологии, которые подходят для сбора, хранения, обработки и распределения знаний (ключевого ресурса информационного общества).

Технологии обработки знаний в настоящее время являются единственными технологиями, чье соотношение эффективности к стоимости непрерывно возрастает.

### *2.3.3. Переход от производства к услугам, основанным на знаниях*

Перемещение массового производства из развитых стран в развивающиеся является неотъемлемой частью глобализации.

Замена массового производства услугами, основанными на знаниях, в качестве основной деятельности по созданию благ, может быть произведена только в тех странах, которые располагают знанием и опытом в современных ИТ, а также большим количеством первоклассных работников умственного труда, среди которых исследователи, инженеры и руководители из самых разных областей, в том числе финансов, ИТ, консалтинга, дизайна, СМИ и т.д.

Услуги, основанные на знаниях, являются продуктами интеллектуального труда: проведение исследований, проектирование и разработка продукции и услуг, прототипирование, разработка производственных технологий, управление проектами, разработка программ и прочие ИТ-услуги, архитектурное проектирование и контроль строительства, маркетинг и торговля, PR и информационное

освещение, консалтинг, организация досуга, управление инвестициями, банковское дело, услуги здравоохранения, образовательные услуги, логистические услуги и т.д.

Большинство этих услуг хорошо известны и не новы, новшеством является применение инновационных технологий оцифровки, хранения, обработки и передачи информации, которая позволяет обмениваться знаниями между континентами.

Быстрое освоение рынка услуг, основанных на знаниях, предоставляет прекрасную возможность для экономического процветания государства в то время, как массовое производство толкает любую страну на столкновение с острой конкуренцией.

Apple, Google, Facebook и Microsoft помещают знание, созданное исследователями и разработчиками, не в материальные товары, а в программные системы. Вполне естественно, что компании, которые воплощают созданные знания в программных системах, печатных изданиях или в консалтинговых услугах, достигают существенно более высокой добавленной стоимости, чем производители, которые должны закупать дорогие материалы и/или компоненты извне.

### *2.3.4. Переход от крупных корпораций к сети малых цифровых предприятий*

Крупные монолитные корпорации являются продуктом индустриальной экономики, которая характеризовалась стабильными рынками, порождающими устойчивый спрос на идентичные товары массового производства. Такие корпорации, будучи устойчивыми и стабильными, процветали в эру, когда экономия на масштабе была ключевым фактором успеха.

Критическим фактором успеха бизнеса на новом сложном мировом рынке является адаптивность, и поэтому мы можем с уверенностью предположить, что будущее крупных корпораций будет не-

легким, за исключением тех, которые производят унифицированную продукцию, например, подгузники или медицинские принадлежности.

Однако у крупных корпораций есть замечательная способность к выживанию, и многие из них совершенствуют свои структуру и процессы, и продолжают работать в обновленном формате, переходя к сетевым организациям.

Организационная структура, наиболее подходящая для поставок на постоянно меняющиеся рынки, это сеть обособленных самостоятельных производственных подразделений, каждое из которых имеет свои собственные уникальные знания и опыт (ресурс знаний).

Часто такая структура строится как *виртуальное предприятие* [6].

Виртуальное предприятие является сетью связанных автономных подразделений (бизнес-центров, департаментов, филиалов), где каждое подразделение может выступать фактически как самостоятельная компания, покупая общие сервисы (бухгалтерию, кадры и др.) у других подразделений на внутреннем виртуальном рынке. Виртуальная организация может состоять из реальных организаций, а также небольших групп специалистов или даже одного человека, также способного действовать как целая организация. Связи между подразделениями могут динамично устанавливаться, разрываться и восстанавливаться в зависимости от нужд предприятия, в результате самоорганизации для приспособления к среде. Таким образом, виртуальные предприятия обеспечивают стратегический запас ресурсов на случай, если некоторые из партнеров будут заняты, когда возникнет необходимость в их услугах. Так как организации-партнеры виртуальных предприятий работают в тех же изменчивых условиях, в их интересах также взаимодействовать со многими потенциальными покупателями их услуг.

Поскольку условия конкуренции все время меняются, виртуальное предприятие может разрывать связи с некоторыми старыми подразделениями и устанавливать связи с новыми, обеспечивая свое развитие. Виртуальное предприятие обычно сосуществует, конкурирует и сотрудничает с другими виртуальными предприятиями и, так как все эти предприятия параллельно развивают свои эффективность и конкурентоспособность (коэволюция), они создают постоянно меняющуюся среду.

Мировая экономика функционирует и развивается как сложная система. У крупных корпораций существует реальная возможность реорганизовать себя в виртуальные предприятия и избежать судьбы организаций-монстров эпохи плановой экономики.

Популярным видом организаций, все больше вкладывающим в создание электронных моделей объектов и автоматизацию всех процессов, является *цифровое предприятие*.

### *2.3.5. Большие объемы данных и выявление скрытых знаний*

Поскольку места, доступного для хранения компьютерных данных, становится всё больше, а Интернет многократно упрощает обмен информацией, появилась тенденция к накоплению чрезвычайно большого количества данных – так называемых «Больших данных» (Big Data) – и их хранения в обширных серверных хранилищах, называемых «облаками» (Clouds).

Эти собранные данные эффективно используются для моделирования ситуаций, проверки гипотез и подкрепления предположений и суждений экспертов. Для этого все большее значение приобретает анализ данных («Data mining»), направленный на выявление скрытых полезных знаний, как ключевого ресурса информационного общества.

Анализ данных уже сегодня широко используется в маркетинге для выявления классов новых потребителей, а также в таких обла-

стях, как спорт (для тренерской деятельности), медицина (для постановки диагноза) или образование (в поиске индивидуальных методов обучения каждого отдельного ученика).

### *2.3.6. Перспективы семантической паутины*

Следующей технологической революцией будет переход от текущего управляемого данными Интернета к семантической паутине (Semantic Web), и этот момент, вероятно, станет поворотным для многих областей, так как новое поколение систем сможет понимать значение данных, например, содержание страниц Интернета.

Семантическая паутина является давней мечтой исследователей теории вычислительных систем, которая в настоящее время воплощается вместе с программами, основанными на онтологиях, а также мультиагентными системами, уже применяемыми на практике [7, 8].

Для полноценной реализации семантической паутины требуется функциональность, существенно превосходящая ту, что предлагают существующие технологии и системы. Так, разъяснение семантики предложений на естественном языке, требует совершенно новых методов и средств сложных адаптивных систем, а не привычного алгоритмического подхода.

### *2.3.7. Изменение менталитета*

Что же характеризует новый образ мышления и зарождающуюся новую методологию решения сложных проблем?

Вот некоторые базовые принципы:

1. Необходимо принять тот факт, что мир, в котором мы живем и работаем, является открытым и постоянно развивается, и что мы должны постоянно адаптироваться к изменениям в нашей среде.

2. Не стоит полагаться на то, что мы сможем точно определить причины наших успехов и неудач, но надо стараться хорошо делать свое дело.

3. Важно понимать, что даже внешне крепкие организации постепенно «отказывают» (распадаются), и что необходимо время от времени подпитывать их новаторскими идеями и методами, чтобы изменить направление и улучшить их эффективность и производительность.

4. Необходимо понимать, что каждым решением и действием мы активно способствуем строительству нашего будущего, и поэтому «короткие» пути (shortcuts) не приемлемы.

5. Нужно помнить, что время является драгоценным ресурсом, и что все решения и действия должны приниматься и выполняться в режиме реального времени.

6. Надо уметь обнаруживать новые потребности и предлагать новые возможности, которые скрывает от нас неопределенность.

7. И, наконец, нельзя останавливаться на достигнутом – даже самые выгодные сложившиеся условия могут уже завтра необратимо измениться.

Рассмотренные принципы легли в основу разработанного нового класса мультиагентных систем для управления ресурсами в реальном времени, которые будут подробно рассмотрены в следующих главах.

## 2.4. Выводы по Разделам № 1 и № 2

1. Рост сложности все больше проявляет себя как новый объективный закон окружающего нас мира.

2. В этих условиях необходимо искать новые пути изучения и использования феномена сложности для дальнейшего приумножения добра и блага нашего общества.

3. Сложная система предполагает в своем составе наличие множества автономных элементов (агентов), способных к взаимодействию и достижению согласованных решений в интересах общего целого.

4. Показывается, что эффективная сложная система всегда находится на границе порядка и хаоса.

5. Предложены 7 ключевых критериев, позволяющих отличать и характеризовать поведение сложных систем.

6. Показана важная связь между ростом сложности и развитием информационного общества.

7. Обсуждается новый образ мышления, помогающий достижению успеха в мире растущей сложности.

# 3. МЕТОДОЛОГИЯ УПРАВЛЕНИЯ СЛОЖНЫМИ АДАПТИВНЫМИ СИСТЕМАМИ

## 3.1. Принцип адаптивности

Традиционные подходы к управлению не могут быть эффективно применимы к сложным адаптивным системам, характеризуемым высокой степенью неопределенности, поскольку принципы самоорганизации напрямую противоречат стремлению тотально контролировать каждый шаг в решении проблемы.

Мы также не можем устранить неопределенность, упрощая сложные ситуации, так как любое изменение в связях изменит результирующее поведение системы, возможно, очень существенным и самым непредсказуемым образом. Если же мы пытаемся применить к сложным ситуациям некие жёсткие структуры, например, иерархическую организационную структуру или неадаптивный бизнес-процесс, то эти структуры, рано или поздно, будут разрушены или продемонстрируют свою ограниченность и крайне низкую эффективность.

Мы также не можем полагаться на сложные математические методы прогнозирования для предсказания будущего. Если будущее неопределенно, как мы можем его точно предсказать? Но так как события в сложных системах все же не случайны, мы можем найти некие шаблоны (образы), описывающие вероятностные модели поведения сложных адаптивных систем.

Мы не можем ожидать, что жесткое директивное планирование будет функционировать в условиях часто происходящих непредвиденных событий, нарушающих равновесие в системе – в таких усло-

виях планы очень быстро перестают соответствовать действительности.

Для эффективного управления сложными адаптивными системами и максимального достижения на практике ожидаемых результатов предлагается *принцип адаптивной перестройки решений и планов действий по событиям в реальном времени*.

При этом лучший подход к адаптивному управлению состоит в развитии способности к самоорганизации, которая обеспечивает наиболее гибкую перестройку и в каждой ситуации может помочь преодолеть или, по крайней мере, уменьшить последствия любых непредвиденных событий.

Этот общий подход применим как к программным системам, так и к любым организациям от малого бизнеса – до крупных корпораций.

Рассмотрим более подробно предлагаемый подход к адаптивному управлению сложными системами.

## 3.2. Параметры управления сложностью

### *3.2.1. Уровень автономности агентов*

*Автономность агентов* обозначает степень свободы в принятии решений, данную им в системе, чтобы решать, как поступать в той или иной ситуации.

На практике степень свободы агентов может определяться, в частности, тем, как широки и разнообразны возможности агента для построения, выбора, принятия и согласования вариантов решений и выработки встречных предложений.

Если у агента нет никакой свободы принятия решений, его автономность будем считать равной 0. Если агент свободен в принятии решений, его автономность равняется 1. В сложных адаптивных

системах автономность агента ($A$) всегда находится в диапазоне $0 < A < 1$.

*Чем выше автономность агентов, тем выше сложность системы.*

Отношение между связностью и автономностью агентов при постоянной силе связей показано на рис. 5.


Рис. 5. Зависимость сложности от автономности и связности агентов при постоянной прочности связей

Отношение между силой связей агентов и уровнем их автономности при постоянном уровне связности показано на рис. 6.

Эксперименты с моделями сложных адаптивных систем показывают следующее:

*Более высокие автономность* и *связность агентов при слабой силе связей, то есть более высокая сложность, способствуют продуктивности и творчеству (креативности) в системе, улучшают ее адаптивность и увеличивают скорость восстановления после критичных разрушающих событий, например, связанных с потерей ресурса.*


Рис. 6. Зависимость сложности от степени автономности агентов прочности их взаимосвязей при постоянной связности

С другой стороны, *ограниченная автономность агентов, их менее высокая связность и более сильные связи, то есть, менее высокая сложность, обеспечивают больший порядок и дисциплину, повышают уровень предсказуемости, уменьшают вероятность ошибок, вероятность критичных событий, но снижают продуктивность и креативность, адаптивность и устойчивость к сбоям.*

Рассмотрим подробнее параметры связности и силы связи агентов.

### *3.2.2. Степень связности агентов*

*Связность агентов* обозначает, насколько регламентированы взаимодействия агентов между собой в системе.

Если агент не может быть связан ни с каким другим агентом, его степень связности равна 0. Если же агент может быть связан со всеми агентами в системе, то его связность равняется 1. В сложных

адаптивных системах связность агента ($C$) всегда находится в диапазоне $0 < C < 1$.

*Чем выше степень связности агентов, тем выше сложность системы.*

Например, в активно изучаемых сейчас грид-системах с переменной структурой каждый узел может иметь связи, например, с 2, 4, 8 или любым другим числом «соседей», что может существенно влиять на скорость перераспределения задач по узлам и, в конечном счете, на качество и эффективность полученного решения.

Пришедшая в любой узел новая задача будет выполнена этим же узлом или может оказаться перераспределена другим «соседям», имеющим меньшую загрузку, которая постоянно выравнивается.

Аналогичным образом может строиться перераспределение заказов на перевозку грузов между грузовиками, начиная с того места, где появился груз, а далее по принципу расширяющейся окружности.

Иными словами, в целях управления степень связности может гибко меняться в ходе вычислений, первоначально ограничивая выбор для агентов ближайшим окружением, но постепенно расширяя локальную область взаимодействий, если хорошего решения не находится.

Фактически, это позволяет управлять качеством и скоростью решения задачи, усиливая или снижая возможности для самоорганизации в системе.

### *3.2.3. Сила связей агентов*

*Сила (прочность) связей между агентами* обозначает взаимную удовлетворенность агентов связью и способность агентов противостоять угрозе разрыва связей.

Иными словами, сила связей показывает, насколько подходят агенты друг другу или соответствуют их идеалам (и какова сила

притяжения между ними), как это можно наблюдать в любых проявлениях ключевых отношений между противоположными сущностями (подобно инь и янь) в процессе их конкуренции и кооперации.

Если связь отсутствует, будем считать, что ее сила равна 0. Если связь идеальна для агента, то ее сила для агента равна 1. В сложных адаптивных системах сила связей агентов ($S$) находится в диапазоне $0 < S < 1$. Более слабые связи легче разорвать и затем создать новые. Очевидно, что итоговая сила связи между двумя агентами может вычисляться как среднее от удовлетворенности связью каждым из них. При этом сохраняется неравновесность связей, в одну сторону связь может быть сильнее, чем в другую.

Снижение силы связей увеличивает сложность, и, следовательно, непредсказуемость глобального поведения системы, поскольку также ведет к повышению возможностей (разнообразия) в выборе решений.

*Чем легче менять связи между агентами, тем сложнее система.*

Слабые связи, которые могут быть разорваны первыми, когда система самоорганизуется, чтобы приспособиться к событию, являются неотъемлемым атрибутом сложности.

Сильные связи сопротивляются самоорганизации, а очень сильные связи могут и вовсе не допустить самоорганизацию, формируя жесткий «порядок» в системе.

Например, в мире грузовых перевозок агент заказа может быть сильно связан с конкретным грузовиком, который везет его с минимально возможной в системе ценой, в силу того, что тот движется по прямой и почти полностью заполнен другими грузами, с которыми разделяет стоимость доставки. Следовательно, агент заказа не будет искать других вариантов «на стороне». И наоборот, какой-то из грузовиков может желать избавиться от груза, из-за которого он

делает большой крюк и теряет прибыль, и тогда он будет постоянно искать возможность найти другой, более выгодный груз.

Следуя аналогии с термодинамикой можно заметить, что чем легче связи и, тем самым, гибче система, тем она ближе к состоянию хаоса и, наоборот, с ростом силы связи между агентами система приходит к состоянию более жесткого порядка.

### *3.2.4. Удовлетворенность агентов*

Еще один важный параметр управления сложностью связан с *удовлетворенностью агентов*, которая может рассматриваться как величина из диапазона [0,1] обратно пропорциональная разнице между текущим и идеальным значением по целевому критерию, заданному агенту (качество обслуживания, себестоимость услуги и т.д.).

Обобщая на случай действия нескольких критериев, можно принять эту величину нормированной сверткой по заданным в текущей ситуации критериям.

Пусть каждый $j$-й заказ имеет несколько частных критериев $x_i$, например, стоимость, прибыль, опоздание, качество, и предполагаемые идеальные значения этих критериев $x_{ij}^{id}$. У каждого агента заказа $j$ тогда можно подсчитать оценочную функцию (ценность) $f_{ij}(x_i - x_{ij}^{id})$ по компоненте $i$. Для каждого заказа определяется свертка оценочных функций с заданными весовыми коэффициентами $a_{ij} \geq 0$.

Надлежащим выбором знаков и вида функций можно свести задачу каждого агента к задаче максимизации ценности $y_j$ заказа $j$ (1):

$$y_j = \sum_i \alpha_{ij} \cdot f_{ij}(x_i - x_{ij}^{id}), \qquad (1)$$

где $\forall j$ весовые коэффициенты нормируются: $\sum_i \alpha_{ij} = 1$.

Для всей мультиагентной системы аналогично может быть сформулирована задача нахождения значений частных критериев

(состояний агентов заказов $j$), максимизирующих суммарную ценность заказов (2):

$$y = \sum_j \beta_j \cdot y_j = \sum_j \beta_j \sum_i \alpha_{ij} \cdot f_{ij}\left(x_i - x_{ij}^{id}\right), \tag{2}$$

$$y^* = \max_{x_i}(y),$$

где $\beta_j$ – вес заказа, позволяющий устанавливать и динамически менять приоритеты.

Аналогичное выражение можно получить для всех ресурсов системы и, в конечном итоге, для системы в целом. Суть этого выражения будет заключаться в том, что мультиагентная система призвана минимизировать разницы между идеальными и текущими значениями по критериям, заданным агентам, т.е. находить консенсус (баланс) интересов всех участников решения сложной задачи.

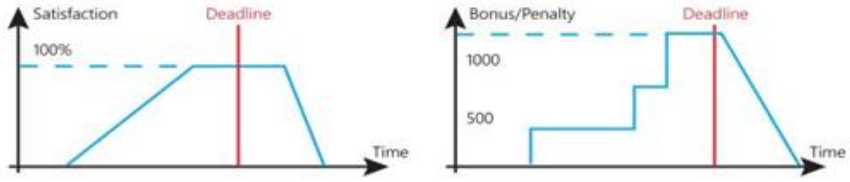
При этом чем больше удовлетворены агенты, тем меньше можно прогнозировать изменений в системе и наоборот, чем дальше агенты от идеала, тем больше усилий им придется приложить к достижению целей, что напрямую будет влиять на стабильность решения и прогноз поведения системы.

С вводом критериев и удовлетворенности появляется возможность управлять целями агентов, указывая направление, к которому должен стремиться агент, придавая по ситуации б*о*льшую важность одному или меньшую важность другому критерию, измеряя успех на каждом шаге работы агента и премируя или штрафуя агента за результат, а также точнее оценивать состояние агентов и системы в целом для прогнозирования ее поведения.

В этой связи наряду с удовлетворенностью вводятся *штрафные-бонусные функции для агентов*, показывающие насколько отклонение от заданного идеала или приближение к нему штрафуется и / или награждается премией.

Премии и штрафы выражаются в виртуальной валюте, которую агенты накапливают на своих счетах при работе на виртуальном рынке.

Например, несрочный заказ на грузовую перевозку может быть вполне «счастлив», если он доставляется в срок, чуть раньше или даже с небольшим опозданием, как показано на рис. 7,а. Однако, за успешное выполнение этой работы агент сможет претендовать на бонус в виртуальных деньгах, представленный функцией бонусов и штрафов, показанной на рис. 7,б.




а б

Рис. 7 . Функция удовлетворенности и функция бонусов/штрафов агента:
а – функция удовлетворенности агентов;
б – функция бонусов / штрафов агентов

Как следует из этого графика функции бонусов/штрафов, далеко не все ранние приезды или опоздания будут максимально премироваться.

С учетом этих функций каждый агент может зарабатывать виртуальные деньги, которые может тратить на улучшение своего состояния и преобразования расписания в нужном ему направлении.

Очевидно, что сложная система с неудовлетворенными агентами и большой запасенной энергией имеет все шансы даже при самом малом новом заказе перейти в совершенно новое состояние через долгий лавинообразный процесс полной перестройки расписания, и, наоборот, система с очень удовлетворенными аген-

тами и низким уровнем энергии вряд ли будет способна на радикальную перестройку.

С другой стороны, привнесение энергии в систему для некоторых агентов может помочь радикально улучшить ситуацию с тем или иным заказом или ресурсом причем точечно (индивидуально), что можно интерпретировать как внутреннюю инвестицию в улучшение выбранного фрагмента расписания.

В результате, каждый агент может быть охарактеризован в системе уровнем удовлетворенности и уровнем своего «энергетического потенциала», который он уже потратил или еще может использовать для перестройки расписания.

Так, некоторые агенты могут «выходить из игры» только потому, что очень удовлетворены своими результатами (даже при наличии денежных средств), а другие потому, что не удовлетворены, но не имеют никаких средств для получения квантов активности или установления связей.

### *3.2.5. Энергия агентов*

Условные денежные средства, получаемые и используемые агентами на своих счетах на виртуальном рынке, играют роль потенциальной энергии, позволяя перестраивать «окружающее пространство» решения, например, формируемого расписания использования ресурсов.

Такие денежные средства агенты могут получать при входе в систему (например, агент заказа может получать процент от своей цены), зарабатывать от предоставляемых услуг (агент грузовика получит за перевозку грузов), получать в виде компенсаций за освобождение места в расписании для более важного заказа, наконец, получать в виде временных дотаций от системы в попытке улучшить те или иные показатели.

Расходы денежных средств могут идти как на оплату услуг, так и на налоги, собираемые на виртуальном рынке, например, на существование в системе, получение управления для поиска новых вариантов, передачи сообщений и проведение переговоров, заключение договоров, установление или разрыв связи и другие. В итоге, чем большими деньгами располагает агент, тем больше возможностей он получает для перестройки расписания в свою пользу.

Это означает, что наибольшие изменения в системе следует ожидать в первую очередь, от агентов, обладающих высокой энергией и являющихся не удовлетворенными.

Анализируя распределение энергии по структуре расписания можно точнее прогнозировать возможное поведение системы и сокращать или увеличивать сложность принимаемых решений. Наблюдая и оценивая общую картину формирования порядка или хаоса в системе, можно предсказать, какие участки могут быть более подвержены изменениям в случае возникновения определенных событий, и, при определенных условиях, даже направленно и точечно инициировать нужные изменения.

*Таким образом, управление сложностью должно являться весьма тонким и деликатным процессом балансировки различных параметров сложности в целях достижения желаемого адаптивного поведения системы.*

Адаптивное поведение таких систем может оставаться в рамках определенных пределов равновесий (аттракторов) при условии, что заданные правила достаточно однозначны, чтобы предотвратить беспорядочное поведение, и все же достаточно гибки, чтобы позволить системе определенную свободу экспериментов при столкновении с новыми проблемами.

*Лучшей стратегией адаптивного управления сложными системами будет ввод разнообразных правил поведения агентов, применяемых по ситуации, например, более строгих в случае, если система работает в обычном режиме, и менее строгих, если*

*требуется срочное восстановление системы после потери критического ресурса или поиск возможности выполнения нового очень выгодного крупного заказа.*

Заметим, что любые правила сами по себе не могут предотвратить системную нелинейность, создающую непредсказуемые критические события. Для уменьшения значимости и снижения частоты критических событий мы должны регулировать автономность, связность агентов и силу связей между ними, а также ряд других параметров. И часто необходимо обеспечивать большую сложность одних частей системы в сравнении с другими. Чтобы достигать неравномерного распределения сложности, рекомендуется группировать агентов в области (регионы) с высокой внутренней связностью внутри каждой из них и низкой связностью между самими областями, как показано на рис. 8.

Низкая связность между регионами гарантирует, что возмущения не смогут распространяться между всеми областями системы, в то время как высокая связность внутри областей обеспечивает их особо высокую внутреннюю адаптивность. Усиление прочности межрегиональных связей и ослабление прочности внутренних связей в областях также может быть полезным.

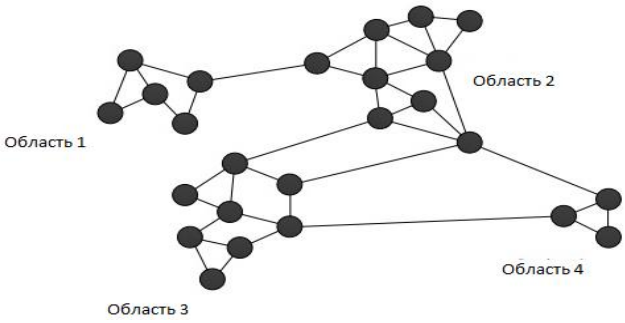



Рис. 8. Разделение большой сети на регионы
для предотвращения распространения последствий критических событий

Например, перепроектируя таким образом сети финансовых услуг, мы могли бы препятствовать распространению нежелательных возмущений из одной области в другие, например, международного мошенничества с кредитными картами.

Другой пример из области грузовых перевозок – более высокие платежи за возможность использовать грузовики из другого региона, что смогут себе позволить только самые выгодные заказы.

Существует мнение, что сложные адаптивные системы не могут быть спроектированы и могут возникать («вырастать») лишь эволюционно. Это мнение ошибочно, но проектирование сложных адаптивных систем существенно отличается от проектирования детерминированных систем. Ведь в итоге такого проектирования должна быть построена модель мира взаимодействующих «умных объектов» (программ) предметной области, способных принимать решения и взаимодействовать, отражающая ситуации и взаимодействия в реальном мире, а не один конкретный алгоритм или программа. Причем в идеале, в будущем, каждый такой мир должен быть открыт к вводу новых типов агентов без перепрограммирования системы.

Управление рассматриваемой внутрисистемной сложностью является важной частью проектирования сложных систем.

# 4. МОДЕЛИРОВАНИЕ СЛОЖНЫХ АДАПТИВНЫХ СИСТЕМ

## 4.1. Требования к моделям сложных адаптивных систем

Существует ряд трудностей в моделировании сложных адаптивных систем, так как они часто не имеют четких границ, чувствительны к малейшим событиям, характеризуются недетерминированным поведением, неравновесием связей и другими нелинейными отношениями между агентами, способны к непрерывной самоорганизации.

Проектирование и моделирование сложных адаптивных систем в настоящее время скорее можно назвать искусством, чем наукой.

Наш опыт в моделировании сложных систем для управления крупными предприятиями позволяет сформулировать два ключевых правила моделирования:

1. Модель должна обладать *необходимым уровнем детальности*. Например, если мы моделируем бизнес грузовых перевозок, будет достаточно создать агента на каждый заказ и каждый грузовик, но, если нам важны и решения по сервисному техобслуживанию грузовиков, то возможно будет необходимо создать агента на каждую станцию техобслуживания и каждый двигатель, шины и другие компоненты грузовика, что можно легко продолжить учетом смен работы водителей, необходимостью выбора мест заправок и т.д.

2. Модель должна обладать *необходимым уровнем адаптивности.* Модель системы должна быть такой сложной, чтобы обеспе-

чивать возможность адаптироваться к изменениям, которые переживает моделируемая система, и такая адаптация должна быть автономной (без ожидания инструкций пользователя), что возможно только в том случае, если сами модели достаточно сложны.

Действительно, если речь заходит об управлении флотилией грузовиков, каждый руководитель мечтал бы о наличии компьютерной модели, показывающей не только где находятся его грузовики в текущий момент (что уже легко видеть посредством датчиков GPS навигации), но и обозревать их планы и понимать, кто и где будет через несколько часов или дней, что собственно и позволяет прогнозировать в скользящем режиме все показатели бизнеса, включая прибыль по каждому грузовику и грузу, возможные точки потери эффективности и т.д.

Если же речь идет об управлении Международной космической станцией (МКС), необходимо наряду с летящей станцией, одновременно, видеть виртуальный полет ее земного «двойника», все состояния объектов и процессы в котором «зеркально» дублируются, что позволяет строить опережающие эксперименты с моделью, точнее прогнозировать и оценивать результаты и добиваться требуемого управления, что уже в настоящее время реализуется в рамках концепции кибер-физических систем.

Эти соображения приводят нас к заключению, что существующие моделирующие системы настоящего поколения (класса GPSS и другие) не могут использоваться для моделирования сложных адаптивных систем и ситуаций, так как они в принципе не являются сложными и адаптивными по своему устройству и не могут меняться самостоятельно, путем самоорганизации, что обычно требует привлечения разработчиков-программистов к самым малейшим изменениям в логике работы системы, это, в свою очередь, значительно удорожает эксплуатацию.

В настоящее время, как нам представляется, только мультиагентный подход может обеспечить действительную адаптивность и стать мощным инструментом моделирования сложных самоорганизующихся систем.

При этом диапазон применения таких систем является весьма широким.

Существует много примеров сложных адаптивных систем, компоненты которых могут быть смоделированы за счет построения достаточно сложной и детальной сетевой модели взаимодействий компонент более низкого уровня, к числу которых и относится большинство явлений из мира бизнеса, промышленности и транспорта, экономики и финансов, социальных, экологических проблем и проблем безопасности.

Итак, *для моделирования сложных систем требуется строить модели, реализуемые как миры предметных областей для взаимодействия агентов*. Как только построен такой мир, в той или иной степени адекватный реальному миру, с его помощью становится возможным моделировать сложные проблемные ситуации, имеющие место в реальном мире при различных состояниях окружающей среды. Например, воссоздать поведение сети поставок в условиях изменчивого состояния спроса и предложения на рынке.

## 4.2. Пример моделирования грузоперевозок для оценки эффективности перехода к принятию решений в реальном времени

### *4.2.1. Постановка задачи*

Покажем, как можно смоделировать адаптивный процесс распределения, планирования и оптимизации ресурсов в реальном времени.

Пусть имеется парк грузовиков из M машин, базирующихся в определенном городе в некоторой транспортной сети и имеющих GPS / ГЛОНАСС датчики на борту. В реальном времени поступают заказы и любые другие события (новые заказы, задержки, поломки и т.д.), которые необходимо планировать, учитывая текущие планы, индивидуальные предпочтения и ограничения заказов и ресурсов (рис. 9).

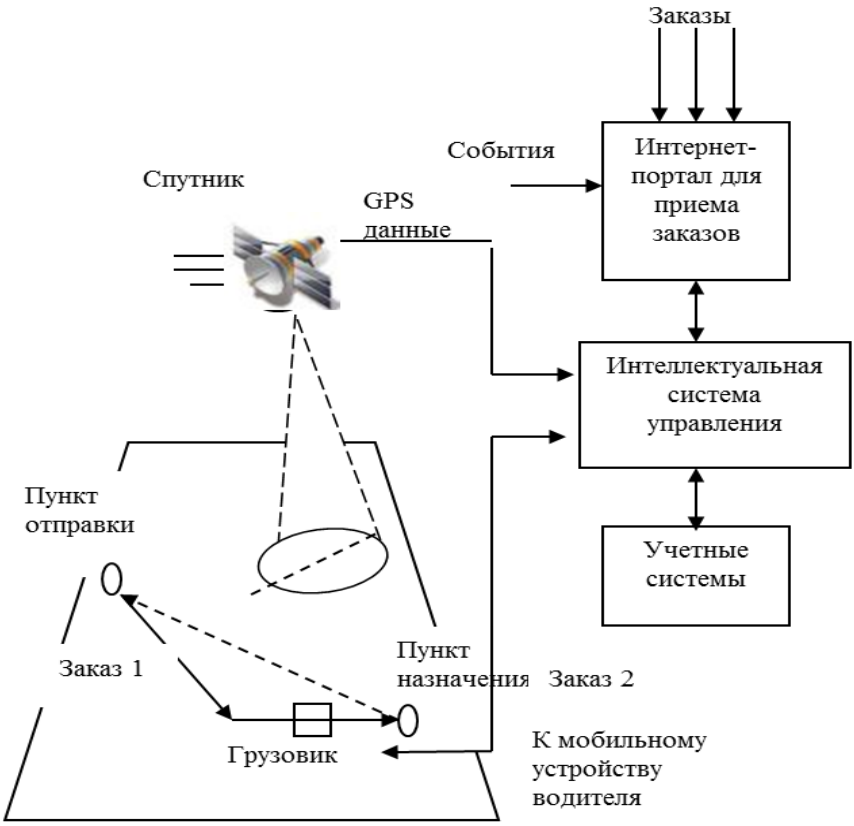



Рис. 9. Общая схема управления грузоперевозками

Изменения должны вноситься в планы ресурсов без останова и перезапуска проектируемой интеллектуальной системы, путем адаптивного изменения расписания «на лету» как с использованием свободных окон, так и подвижками / переброской ранее распределенных заказов.

Должен быть реализован полный цикл управления:

- реакция на события;
- динамическое планирование (перепланирование);
- согласование и пересмотр планов «на лету»;
- мониторинг и контроль исполнения планов.

Согласование планов должно осуществляться через сотовый телефон в ходе диалога с водителями. В случае расхождения плана и факта требуется автоматическое перепланирование и согласование с заказчиками, менеджерами и водителями.

Стоимость эксплуатации каждого грузовика известна и заранее задана.

Заказы характеризуются пунктом отправления, пунктом доставки, временем поступления, моментом начала (погрузки), моментом окончания (разгрузки), стоимостью и размером штрафа за опоздание.

Расстояния между всеми пунктами также известны и заданы матрицей расстояний.

Требуется адаптивно перестраивать план грузоперевозок по событиям в реальном времени и рассчитывать прибыль транспортной компании на любой момент времени. Планирование в реальном времени означает, что в каждый момент времени известны только те заказы, которые поступили до этого момента.

Необходимо исследовать зависимость прибыли от типовых стратегий планирования грузовых компаний, количества грузовиков и порядка поступлений событий.

Критерием оптимизации служит максимизация суммарной прибыли предприятия.

### *4.2.2. Модели организации грузовых перевозок*

Исследуем решение поставленной задачи для четырех типовых моделей организации процесса грузоперевозок.

Общая прибыль системы подсчитывается как сумма прибылей всех агентов грузовиков (3):

$$P = \sum_i P_i , \tag{3}$$

где $N$ – набор всех заказов, выполненных грузовиком $i$.

Прибыль отдельного агента грузовика (4):

$$P_i = \sum_{j \in N_i} \{(c_j - q_i)t_{ij} - q_i t_{ij}^{'}, \tag{4}$$

где суммирование выполняется по всем выполненным грузовиком i заказам j, cj – стоимость выполнения заказа j в единицу времени, qi – стоимость переезда грузовика в единицу времени, t ij – время выполнения заказа j грузовиком i, t ’ij – время переезда для выполнения заказа j.

**Модель 1.** Возвращение на базу. Для каждого заказа индивидуально бронируется свой грузовик, в расписании которого на момент поступления заказа есть подходящее «окно». Если погрузка осуществляется в другом городе, то грузовик должен прибыть туда «холостым ходом» ко времени погрузки. Отмена бронирования грузовика под заказ не допускается. После выполнения заказа грузовик возвращается в пункт базирования.

**Модель 2.** Челнок (Без возвращения на базу). После выполнения заказа грузовик не возвращается на базу и остается в пункте назначения, где ожидает следующего заказа. При необходимости, компания может выслать замену водителю другим транспортом к нужному месту, но пока данный процесс не рассматривается.

**Модель 3.** Опоздание со штрафами без перебронирования. Допускается планирование заказов с опозданиями, когда фактический момент начала выполнения заказа будет позже требуемого заказчиком. Из прибыли вычитается штраф, пропорциональный времени опоздания. Если штраф превышает возможную прибыль от выполнения заказа, такой заказ не планируется. После выполнения заказа грузовик остается в пункте назначения. Штрафы учитываются следующим образом (5):

$$P_i(t) = \sum_j \left\{ \left( c_j - q_i \right) t_{ij} - q_i t'_{ij} \right\} + \sum_k \left\{ \left( c_k - q_k \right) t_{ik} - q_i t'_{ik} - p_k t''_{ik} \right\}, \qquad (5)$$

где суммирование по индексу *j* идет по всем выполненным в срок грузовиком *i* заказам, суммирование по индексу *k* идет по всем выполненным с опозданием $t''_{ik}$ заказам, *pj* – штраф на единицу времени опоздания. Этот вариант вносит определенную большую свободу, но еще не является адаптивным, в котором планы меняются «на лету».

**Модель 4**. Адаптивное планирование со штрафами. Совпадает с предыдущей, однако допускается перебронирование грузовика под новый заказ в случае, если прибыль от нового заказа превышает прибыль от прежде запланированного. Таким образом, при поступлении заказа перераспределяются уже размещенные заказы, и ищется новое, более выгодное по прибыли, решение в группе грузовиков, затрагиваемых изменениями.

### *4.2.3. Краткое описание модели работы грузоперевозок*

Предположим, что с каждым грузовиком связан свой менеджер грузовика, а с каждым заказом – менеджер заказа, которые могут отправлять и получать сообщения и принимать решения согласно

своей логике и текущей ситуации, определяемой состоянием каждого участника. В целях наглядности результатов и унификации логики принят единый пространственно-временной масштаб: время отсчитывается от начала моделирования, т. е. от момента поступления первого заказа. Верхняя граница диапазона задается горизонтом планирования в днях. Расстояния приведены к временной шкале делением расстояния на среднюю скорость. Таким образом, может быть учтено состояние и пропускная способность дорог. Поэтому более длинный пространственный путь может быть по времени короче из-за большей скорости движения.

Текущие состояния заказов и ресурсов изменяются и фиксируются в моменты поступления заказов в систему, в моменты начала и окончания их выполнения, поэтому шкала отсчетов времени в общем случае $N$ заказов состоит из $3N$ точек.

При поступлении нового заказа рассылается запрос на его размещение всем менеджерам грузовиков, которые анализируют свое текущее состояние, наличие окон в будущем расписании, необходимость дополнительного переезда до пункта погрузки, оценивают свою возможную прибыль и отправляют ответ менеджеру заказа. Кандидаты на перепланирование (в случае возрастания прибыли) выстраиваются в специальной таблице по максимально возможной прибыли. Заказ получает тот грузовик, который дает максимальную прибыль. Прибыль учитывается как разница между стоимостью заказа и стоимостью транспортировки. Если для выполнения заказа требовался переезд, то его стоимость вычитается из стоимости заказа. Поэтому дорогие заказы, требующие длительных переездов до начала выполнения, могут быть вытеснены более дешевыми, но без дополнительных переездов. В случае стратегии с планированием заказов с опозданиями анализируется влияние штрафа на прибыль. Поскольку штраф пропорционален времени задержки, заказы с большим опозданием не будут планироваться.

Заказы, находящиеся в прошлом (по отношению к текущему времени), в переговорах (отправке и приёме сообщений) не участвуют.

Такой процесс продолжается с течением времени по событиям прихода, начала и окончания заказа, тем самым имитируется поступление заказов в реальном времени.

В результате моделирования по каждой из четырех моделей путем взаимодействия агентов происходит самоорганизация менеджеров «заказы – грузовики», организуется расписание для каждого грузовика и создается общий план, реализующийся участниками по мере течения времени.

Для оценки результата процесса самоорганизации вычисляется суммарная прибыль, получаемая от всех грузовиков.

### *4.2.4. Примеры моделирования*

Рассмотрим модельный пример расчета прибыли при адаптивном планировании перевозок в реальном времени одним грузовиком [9].

**Описание задачи.** Имеются 4 города (пункта), расстояния между которыми задано матрицей (табл. 4) в днях пути. Время в пути не обязательно соответствует расстоянию, поскольку качество дорог разное и скорости движения по ним могут быть разными.

Таблица 4. **Матрица расстояний между пунктами**

| | Пункт 1 | Пункт 2 | Пункт 3 | Пункт 4 |
|---|---|---|---|---|
| Пункт 1 | 0 | 1 | 1 | 2 |
| Пункт 2 | 1 | 0 | 2 | 1 |
| Пункт 3 | 1 | 2 | 0 | 1 |
| Пункт 4 | 2 | 1 | 1 | 0 |

В начальный момент времени грузовик расположен в пункте 1. В различные моменты времени случайным образом поступают заказы № 1-5 на перевозку грузов в разные пункты. Длительность выполнения заказов 1-2 дня. Горизонт планирования $t$ = 10 дней. Стоимость выполнения заказа по тарифу компании постоянна и равна $c$ = 3 у. е. / день, т. е. если длительность выполнения заказа 2 дня, то доход от его выполнения составит 6 у. е. Если грузовик простаивает, то каждый день простоя приводит к затратам в $qa$ = 0,3 у. е. Каждый день движения при перегоне пустого грузовика или при выполнении заказа приносит затраты $q$ = 1. Разрешается выполнять заказы с опозданием, за каждый день опоздания взимается штраф в размере $pp$ = 0,6 у. е. Некоторые заказы при этом сдвигаются вправо по оси времени.

Требуется планировать перевозки по мере поступления заказов (о заказах заранее ничего неизвестно) и определять прибыль.

Заказы описываются номером в порядке поступления, датой поступления (моментом времени $t$), моментами начала и окончания выполнения, длительностью (в днях), начальным и конечным пунктами (табл. 5).

На рис. 10 заказы представлены прямоугольниками, внутри каждого из них указаны номер заказа и через запятую момент его поступления, сверху прямоугольника указано «откуда – куда». Начало и конец каждого прямоугольника соответствуют моментам начала и окончания выполнения.

Таблица 5. **Характеристики заказов**

| **Характеристика** | **Номер заказа** | | | | |
|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 |
| **Время поступления** | 1 | 3 | 5 | 6 | 7 |
| **Время начала** | 3 | 4 | 7 | 8 | 9 |
| **Время окончания** | 5 | 5 | 9 | 9 | 10 |
| **Откуда** | 4 | 3 | 1 | 4 | 3 |
| **Куда** | 1 | 1 | 4 | 3 | 1 |

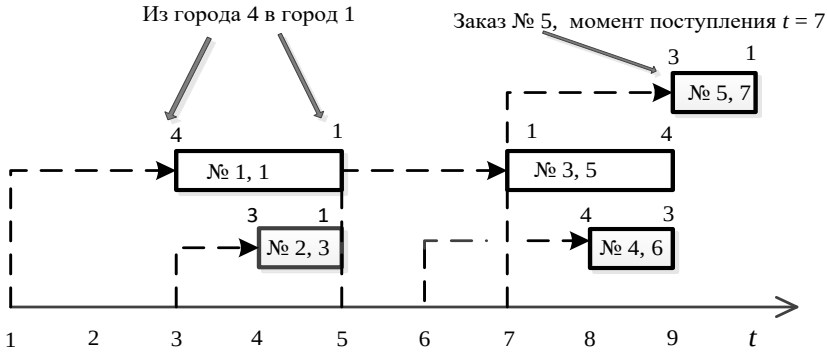



Рис. 10. Диаграмма поступления заказов

Подсчитаем прибыль грузовика 1 в модели 3 с учетом штрафов. Будем вычислять прибыль $P$ в моменты изменения состояния грузовика.

**Расчет прибыли.** *Шаг 1.* Выполнение заказа № 1 потребует выезда в момент $t = 1$ из пункта 1 в пункт 4 и займет 2 дня до момента $t = 3$. На момент $t = 3$ прибыль $P = -q * 2 = -2$. Будем показывать изменение прибыли $P$ в реальном времени.

*Шаг 2.* Движение с грузом из пункта 4 в пункт 1 займет 2 дня и к моменту $t = 5$ грузовик окажется в пункте 1 с прибылью $P = -2 + (c - q) * 2 = -2 + 2 * 2 = 2$.

Предположим, что агент грузовика оценивает варианты исполнения последующего плана по прибытии в пункт 1 в момент $t = 5$. Его прибыль к этому моменту $P = 2$. К этому времени в момент $t = 3$ поступил заказ № 2. Есть два варианта его исполнения:

- заказ № 2 выполняется с опозданием;
- не брать заказ № 2, оплачивая простой, а потом взять заказ № 3 из того же пункта 1; поскольку заказ № 2 можно выполнить с опозданием до начала выполнения заказа № 3, другие варианты рассматривать не требуется. Рассмотрим эти варианты подробнее.

*Шаг 3.* Нужно доехать из пункта 1 в пункт 3 (1 день) и выполнить заказ № 2 переездом из пункта 3 в пункт 1 (1 день). Искомый прирост прибыли $dP = -1 * q + (c - q) * 1 = -1 + 2 = 1$. Штраф за опоздание $-pp * 2 = -2 * 0{,}6 = -1{,}2$. Итого, к моменту $t = 7$ грузовик окажется в пункте 1 с прибылью $P = 2 + 1 - 1{,}2 = 1{,}8$. Казалось бы, выполнение этого заказа невыгодно, но нужно учесть, что при его невыполнении грузовик простаивает 2 дня и к моменту $t = 7$ прибыль $P = 2 - 2 * 0{,}3 = 1{,}4$.

*Шаг 4.* Поэтому грузовику выгодно выполнить с опозданием заказ № 2, выполнить заказ № 3, $t = 7…9$, (из пункта 1 в пункт 4) 2 дня, прибыль $P = 1{,}8 + 2 * (c - q) = 1{,}8 + 2 * 2 = 5{,}8$ и грузовик переезжает в пункт 4.

*Шаг 5.* К моменту $t = 9$ есть новый заказ № 5 в пункте 3 с началом выполнения $t = 9$, переезд до него равен 1 дню, что сдвигает заказ за горизонт 10. Поэтому грузовик отказывается от него. Имеется также просроченный заказ № 4 из пункта 4 в пункт 3, начало его выполнения должно быть в момент $t = 8$. Грузовик оценивает прибыль от возможного смещения выполнения на 1 день.

*Шаг 6.* Выполнение заказа № 4, переезд не требуется, $dP = (3 - -1) * 1 = 2$ – штраф 0,6 = 1,4. Если бы не было этого выполнения, то грузовик стоял бы 1 день до горизонта и тогда $dP = -1 * 0{,}3 = -0{,}3$. Поэтому грузовику выгодно согласиться.

Полная прибыль за 10 дней $P = 5{,}8 + 1{,}4 = 7{,}2$ (рис. 11).

*Итог*: выполняются заказы № 1 и 3 без опоздания, заказ № 2 с задержкой на 2 дня и заказ № 4 с задержкой 1 день. Заказ 5 не выполняется (рис. 11). На рис. 12 темным цветом показаны отсроченные заказы, которые выполняются со штрафами; светло-серым цветом выделены заказы, выполняющиеся без опоздания; сдвиги показаны широкими стрелками; сдвигаемые заказы показаны точками; белым цветом показан невыполненный заказ.

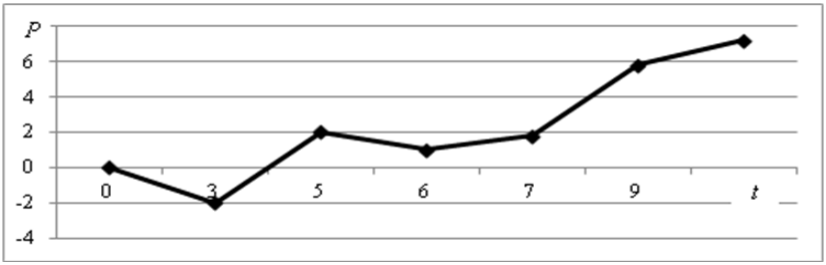



Рис. 11. Прибыль грузовика в зависимости от времени

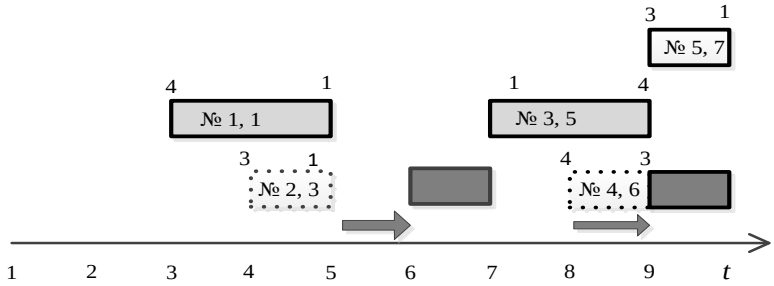



Рис. 12. Диаграмма выполнения адаптивного плана одним грузовиком

Итоговое движение грузовика показано на рис. 13. Белыми стрелками показаны переезды; светло-серыми с пунктирной границей – выполнение заказов с опозданием; темно-серыми – выполнение заказов в срок.

Сначала грузовик выезжает из города 1 в город 4. Затем выполняет заказ № 1 из города 4 в город 1 без опоздания. Потом перемещается в город 3 для выполнения заказа № 2. Затем выполняет с опозданием заказ № 2. После этого он из города 1 выполняет заказ № 3 в город 4 без опоздания. Затем с опозданием выполняет заказ № 4. Заказ № 5 остается невыполненным, поскольку выходит за горизонт ($t = 10$).

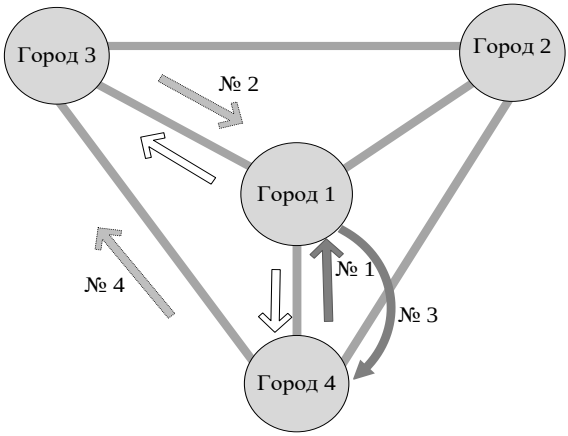



Рис. 13. Схема выполнения заказов грузовиком в модели 3

### 4.3. Развитие способности к адаптации

Адаптивность – ключевой фактор успеха для всех, кто действует на мировом рынке, который характеризуется частым возникновением непредсказуемых событий. Задачи управления можно свести к адаптивному распределению *возможностей* (*ресурсов*) по *потребностям* (*заказам*), поскольку это один из наиболее трудно решаемых вопросов в условиях растущей сложности.

Разделили ресурсы на следующие классы:

- человеческие или кадровые (водители и рабочие, пилоты авиалиний, инженеры по эксплуатации, страховые эксперты и др.);
- физические (машины, станки, транспорт, склады, заводы, здания, земля и др.);
- финансовые (оборотный капитал, инвестиции, кредиты и др.);
- информационные (понятия, кластеры данных, образы, обнаруженные на изображении, слова текста и др.).

Существующие автоматизированные системы планирования ресурсов предприятия (класса ERP – Enterprise Resource Planning), которые были разработаны для предприятий, функционирующих в условиях индустриальной экономики, характеризующейся стабильным рынком, как правило, работают в режиме пакетной обработки данных. Этот режим был вполне уместен, когда экономия на масштабе была ключевым критическим фактором успеха. Но такие планировщики и оптимизаторы оказываются не в состоянии справиться с динамикой современных предприятий, т.к. каждый раз при обнаружении событий изменений в ситуации должны начинать планирование с самого начала, при этом комбинаторный перебор требует продолжительного времени на поиск оптимального решения, которое может устареть прежде, чем будет найдено.

Такие системы становятся трудно применимыми на практике для управления в реальном времени, когда «на лету» меняются как заказы, так и ресурсы системы.

Кроме того, эти системы управляют человеческими, материальными и финансовыми ресурсами по текущим данным, без какого-либо использования знаний, которые могут составлять еще один класс ресурсов нового типа и могут быть эффективно использованы для принятия решений в случае различных непредвиденных ситуаций. Например для того, чтобы агент заказа мог найти другой цех для выполнения заказа, не по жесткой инструкции, а анализируя умения рабочих.

Вместе с тем, такие ERP системы все еще остаются важным компонентом программного обеспечения любого современного предприятия несмотря на то, что условия рынка изменились до неузнаваемости.

*Мы утверждаем, что для предприятий, работающих на современном мировом рынке, распределение, планирование и оптимизация ресурсов должны быть адаптивными и учитывать знания наравне с прочими ресурсами.*

Речь идет фактически о создании нового класса интеллектуальных систем адаптивного управления ресурсами предприятий на основе мультиагентных технологий.

Покажем требования к системам рассматриваемого нового класса.

## 4.4. Что нужно делать, чтобы быть адаптивным?

Основываясь на нашем опыте разработки адаптивных систем управления ресурсами для предприятий и организаций, выделим семь главных требований к адаптивным системам.

### *4.4.1. Принятие решений в реальном времени*

Принятие решений о том, какие действия следует предпринимать при наступлении очередного события, должно происходить в режиме реального времени, так как качество и эффективность решений во многом зависят от момента времени. Однако сложность такого рода решений состоит в том, что в условиях ограниченности ресурсов данный процесс может включать в себя пересмотр ранее согласованных и принятых решений по распределению возможностей по потребностям.

Иными словами, требуется выявление конфликтов и поиск согласованных решений (консенсуса) между агентами системы, имеющими многочисленные индивидуальные предпочтения и ограничения, причем в реальном времени.

Следует понимать, что при этом совершенно не требуется каждую секунду пересматривать принятые ранее решения и можно даже накапливать события в специальной очереди, но важно видеть, какие события приходят, и самые важные из них должны сразу запускаться в обработку и менять планы, поддерживая событийную

диспетчеризацию, либо использоваться для опережающего моделирования и прогнозирования последствий событий.

На каждом предприятии должен ежедневно сводиться баланс плана и факта выполненных работ, показывающий, насколько предприятие отстает от принятых планов работы или, наоборот, идет с опережением, какие ресурсы перегружены, а где есть значительный простой, какие есть узкие места на рассматриваемом горизонте планирования и т.д.

Вместе с тем, для многих существующих предприятий жизненный цикл планирования составляет около 1 месяца, что поддерживает низкую эффективность работы. Предприятия получат выигрыш от перехода к реальному времени.

### *4.4.2. Максимальная отсрочка в реализации решений*

После того как было принято решение о реакции на событие, реализация найденного решения (отправка нового расписания агентам) может быть задержана на максимально допустимое в сложившейся ситуации время для того, чтобы дать возможность прийти новым событиям и еще раз пересмотреть расписание для улучшения показателей бизнеса.

Например, можно заранее забронировать грузовик и держать его в гараже для важного заказа, чтобы не опоздать к точке погрузки ни при каких условиях, а можно продолжать использовать машины, пока они остаются в зоне досягаемости и могут быть в любую минуту быстро отозваны от других заказов. Очевидно, что выигрыш имеет вторая система, что не исключает и резервирования машины при определенных условиях, когда стоимость возможных штрафов от опоздания превзойдет возможную прибыль.

Это еще один пример динамически формируемого граничного условия, которое меняет поведение системы в некоторый момент времени.

### *4.4.3. Проактивная коммуникация с пользователями*

Мы привыкли, что все наши системы пассивны и лишь отрабатывают задания пользователя.

Рассматриваемые системы должны обладать способностью к проактивной работе по улучшению своего состояния в непрерывном взаимодействии со средой и другими системами.

Например, возможна проактивная коммуникации с пользователем для обращения за корректировкой заданных ограничений, когда ожидаемый результат невозможно достигнуть. Например, адаптивная система планирования рабочего времени в цеху машиностроительного предприятия должна просить разрешения вывести группу рабочих на сверхурочные работы, если не удается выполнить план в заданный срок. При этом стоимость сверхурочных работ должна быть, конечно, меньше того запаса прибыли, что приносит рассматриваемый заказ.

Другой пример: если грузовая машина забронирована под заказ клиента, а заказ до сих пор не поступил, следует выйти на менеджера, чтобы уточнить, не забыл ли тот направить заказ.

В этих случаях значительное расхождение плана и факта (в том числе, по моделируемому прогнозу) должно побуждать систему к действиям, направленным на разрешение проблемной ситуации.

### *4.4.4. Поддержка командной работы*

Находящиеся в системе агенты распределяют возможности по потребностям, например, назначая задачи менеджерам или рабочим, с помощью процесса переговоров, выявляющего и разрешающего конфликты и помогающего находить согласие между агентами.

Такой подход (в противопоставление управлению «сверху вниз») гарантирует, что требования, ограничения и предпочтения

всех участников учитываются, даже если они конфликтуют друг с другом. Финальное решение принимается агентами коллегиально при нахождении консенсуса и основано на контролируемых компромиссах для урегулирования конфликтов.

Степень эластичности по каждому критерию должна быть управляемой пользователями и в будущем зависеть от ситуации, для того чтобы позволить сложной системе саморегулироваться на основе достигаемых результатов.

Это является залогом успешной командной работы и для групп людей, эффективность взаимодействия которых может быть тем самым повышена.

### *4.4.5. Динамическое прогнозирование*

Будущее не может быть предсказано, но мы можем его прогнозировать, и точность прогноза обычно увеличивается при приближении к настоящему моменту времени. Использование динамического адаптивного прогнозирования событий на уровне каждого агента означает периодическую регулярность или событийность в обновлении прогнозов с учетом поступающих фактических событий.

Например, агент грузовика должен прогнозировать, какова вероятность подхвата груза в пункте назначения его текущей поездки и, возможно, при определенных условиях даже очень выгодный заказ не стоит брать в удаленный город, если обратно придется ехать пустым, поскольку вся прибыль будет затрачена на обратную дорогу.

Другой пример – агент товара в магазине должен менять прогноз своей продажи по каждой пробивке чека на кассе, что позволит ему оценить границы момента времени для очередной доставки и вовремя договориться с подходящим грузовиком.

Кроме того, рассматриваемые системы могут в отдельной ветви работы, например, на параллельно работающем сервере, копировать текущую ситуацию и запускать виртуальные события (возможный новый заказ, вероятная по статистике поломка станка и т.д.), которые опережающим образом «исследуют» пространство возможных решений, еще до момента наступления таких событий, подтверждая готовность или неготовность отразить возможные проблемные ситуации.

Такое прогнозирование может осуществляться постоянно в скользящем режиме, непрерывно формируя «радар рисков» для предприятия.

### *4.4.6. Экспериментирование с результатами*

В сложных системах агенты, принимающие решения, должны иметь возможность экспериментировать для достижения лучших результатов. Если сложившийся фрагмент расписания выглядит неудачным с точки зрения некоторых критериев, должна быть возможность направленно разрушить данный фрагмент и построить расписание заново, например, при других начальных условиях.

Важно понимать, что в рассматриваемых системах решение строится эволюционно и даже другой порядок прихода событий мог бы вызвать другой результат.

Существующее решение можно запомнить до проведения эксперимента и, если результат получится лучший результат, можно заменить решение на новый вариант. В результате любое формирующееся решение сложной задачи в системе (например, расписание) может подвергаться испытанию по методу «проб и ошибок» (или целевым «провокациям»), что позволит оценивать его качество и устойчивость.

Данный процесс может напрямую регулироваться с учетом резерва времени, отведенным системе на поиск решения.

### *4.4.7. Обучение из опыта*

Интеллектуальная система должна обладать способностью извлекать знания из своего опыта, то есть обучаться в ходе работы. При этом важно обнаруживать образы ситуаций (паттерны), дающие решения как с успешными, так и с отрицательными результатами, которых следует избегать.

Например, есть ли смысл вновь назначать рабочего на выполнение высокоточных операций, если он уже несколько раз допустил брак в подобных операциях? При этом следует обратиться к менеджеру и направить рабочего на переобучение или исключить работу из списка его компетенций.

## 4.5. Проектирование адаптивных решений

Проектирование начинается с создания базы знаний мира предметной области и бизнес-процессов для выработки, принятия и согласования решений.

### *4.5.1. Создание базы знаний*

Концептуальные знания лучше всего описываются при помощи онтологии, узлы сети которой представляют классы понятий, определенные набором свойств (атрибутов), а связи – это классы отношений между классами понятий. Онтология позволяет описать модель любой ситуации при помощи понятий и отношений между ними. Возможно также введение в онтологию и описания сценариев действий и рассуждений, которые позволят конструировать бизнес-процессы предприятия или организации.

Например, для заказа места в самолете и планирования полета необходимыми классами объектов будут: полет, пассажир, самолет,

пилот, цена места, маршрут, транспортная сеть и другие. Примеры возможных атрибутов для класса объектов «полет»: номер рейса, аэропорт вылета, аэропорт прибытия, время вылета, время регистрации и т.д. Процесс бронирования может быть описан как процесс поиска возможного места для потребности пассажира в полете, согласования с пассажиром цены места, проведения оплаты и выдачи подтверждения, каждая операция которого может выполняться специализированным агентом.

Сценарии действий и рассуждений, описывающие логику поведения агентов, определяют уровень их автономности. Если агенты будут вести себя согласно единственному возможному сценарию, другими словами, одному единственному алгоритму, их поведение будет строго детерминировано, и адаптивность бизнес-процессов будет самой низкой. При большом выборе сценариев и возможности агентов экспериментировать в условиях, когда предложенный сценарий не соответствует реальности, адаптивность бизнес-процессов увеличивается. Например, при возникновении проблемы с бронированием агент может переключаться на другую авиакомпанию или другой сервер и т.д.

Фактические знания, например, списки возможных к использованию ресурсов (авиакомпаний, серверов и т.п.) могут храниться и в традиционных базах данных.

Следующим шагом после описания концептуальных и фактических знаний должно быть построение онтологической модели проблемной ситуации предметной области, известной как *сцена*, состоящей из экземпляров классов объектов и их отношений, определенных в онтологии.

Для авиакомпании такая сцена может быть, например, выражена семантической сетью, в которой узлами будут Пассажир P1, Пассажир P2, …, Рейс F1, Рейс F2, …, Место S1, Место S2, …, Самолет A1, Самолет A2, … и т.д., а связями будут “S1 бронируется для P3”, “A1 назначен на F2” и т.д.

Сложные системы, такие как цепочки поставок крупных международных организаций, а также онтологии и сцены, которые их описывают, могут содержать сотни и тысячи объектов, атрибутов, правил и отношений. Эти условия применения онтологий для управления ресурсами во многом меняют требования к конструкторам онтологий, которые используются сегодня в основном для аннотации Интернет-страниц и других подобных применений.

### *4.5.2. Построение виртуального мира предметной области*

Виртуальный мир – это мир агентов, небольших программных объектов, которые, взаимодействуя через общие данные или обмениваясь сообщениями друг с другом, коллективно решают поставленную задачу.

Агент назначается каждому объекту сцены (узлу семантической сети), другими словами, каждой сущности. Например, в виртуальном мире системы планирования авиакомпании могут быть агенты рейса, самолета, пилота, аэропорта, места и т.д.

Виртуальные миры позволяют принимать решения в режиме реального времени путем переговоров между агентами, откладывать принятие решений на допустимые сроки, гарантировать, что изменяются только части связей системы, затронутые событием, контролировать происходящие события и быть всегда к ним готовыми.

Сложность виртуального мира может быть изменена через введение новых типов агентов и новых типов переговоров между ними, равно как и через усложнение конструкции и логики работы самих агентов. Наиболее развитые виртуальные миры должны содержать агентов, способных к самообучению опытным путем и приобретению знаний, чтобы более эффективно планировать ресурсы.

### *4.5.3. Управление реальным миром через виртуальный мир*

Реальный мир компании, например, по грузовым перевозкам, постоянно изменяется, что становится известным для сложной системы через получение событий.

Для авиакомпании такими событиями, вызывающими возмущения в системе, могут быть: бронирование мест, отправление рейса, задержка рейса, отмена рейса, закрытие воздушного пространства, отказ самолета и т.д.

Факт возникновения каждого реального события должен быть по возможности сразу отражен в виртуальном мире, где создается соответствующее виртуальное событие, заставляющее затронутую часть виртуального мира приспосабливаться к изменениям, порожденным в реальном мире.

Но в свою очередь, каждое решение по изменению (адаптации) виртуального мира должно быть сразу передано обратно в реальный мир для его реализации. Очевидно, что задержка в вводе всех этих событий, может самым негативным образом сказаться на принятии решений, например, самолет улетит без пассажира.

По приходу событий из реального мира агенты виртуального мира решают, как реальный мир должен приспособиться к новому событию (рис. 14).

Например, если в реальном самолете во время полета частично повреждается одна из частей, то его модель, представленная агентом самолета, тоже становится неисправной. Это событие может немедленно, с опережением, запустить процесс планирования специального обслуживания поврежденной подсистемы самолета, как только он приземлится в аэропорту.

При этом агент части самолета, ответственный за свой узел в общей семантической сети самолета, посылает сообщения агентам всех связанных с ним и вовлечённых в новую проблемную ситуацию уз-

лов для того, чтобы они знали, что эта часть повреждена, и могли спланировать действия для парирования проблемной ситуации.

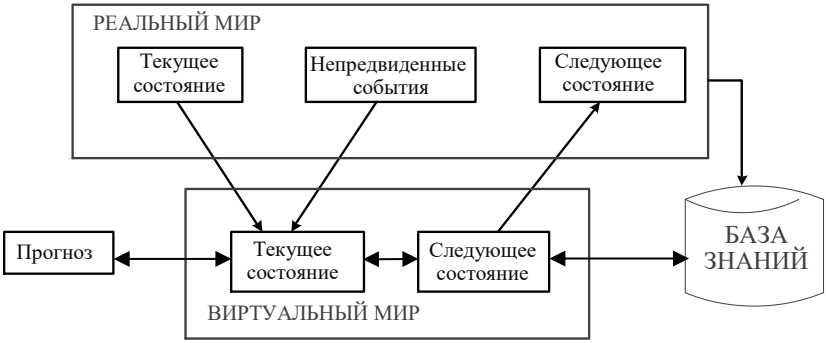



Рис. 14. Виртуальный мир управляется реальным миром и, в свою очередь, управляет реальным миром

В результате такое сообщение активирует перепланирование деятельности многих связанных участников, попадающих в общую «рабочую группу» пострадавших агентов, которые пытаются адаптироваться к новым условиям путем поиска замены части, проведения сервисного обслуживания и т.д.

Как только такое решение найдено, оно передается в реальный мир для реализации, обеспечивая совместное развитие двух миров: реального и виртуального (ко-эволюция).

Отметим еще раз, что, как только соответствующий виртуальный мир построен, он может быть использован не только для управления реальным миром, но также для моделирования возможного поведения реального мира в различных условиях, например, для изучения поведения цепочек поставок в меняющихся условиях рынка, заданных на горизонте времени.

Таких одновременно запущенных моделирующих систем может быть несколько. Они могут анализировать будущее системы

при различных условиях (событиях), непрерывно отвечая на вопросы «Что будет, если …».

Например, пусть при планировании работы аэропорта неожиданно появился прогноз погоды, сообщающий, что в 19.00 возможно будет сильная гроза, и необходимо узнать, какие рейсы скорее всего будут задержаны в этом аэропорту. В таком случае одновременно с основной реальной версией планировщика аэропорта может быть запущена моделирующая версия, куда данное событие может быть введено (иначе в реальной версии системы это событие вызовет изменения, которые на самом деле могут и не случиться). В итоге будет получено прогнозируемое состояние аэропорта в случае грозы, что вручную сделать может быть крайне сложно.

Более того, таких одновременно запущенных «траекторий» жизни мира аэропорта может быть несколько, чтобы оценить последствия самых разных виртуальных событий и решений, которые могут и не произойти в реальной жизни.

Все это может существенно повысить качество, гибкость и эффективность, надежность и обоснованность вырабатываемых и принимаемых решений для управления предприятиями реального мира.

### 4.6. Выводы по Разделам № 3 и № 4

1. Методология управления сложными системами предполагает высокую адаптивность в принятии решений элементами этих систем, включающую возможность изменения ранее принятых решений.

2. Для управления сложными системами важны параметры, понижающие или повышающие сложность, к числу которых можно отнести степень автономности и связности элементов системы, уровень их удовлетворенности и силу связей между ними.

3. На примере моделирования грузоперевозок показана возможность повышения эффективности использования ресурсов при переходе к адаптивному управлению в реальном времени.

4. Предложено 7 принципов, позволяющих повысить адаптивность в принятии решений, включая прогнозирование событий, поддержку командной работы с выработкой согласованных решений, проактивную коммуникацию и т.д.

5. Рассмотрены высокоуровневые компоненты архитектуры интеллектуальной системы для решения сложных задач управления ресурсами.

6. Показано, что для управления любой сложной системой должна быть создана ее адекватная компьютерная модель, которая может зеркально отражать процессы в реальной жизни и адаптивно перестраиваться по мере возникновения событий изменений в окружающей среде.

7. Высокая адаптивность может существенно повысить оперативность, качество и эффективность, гибкость и надежность принимаемых решений для управления предприятиями.

# 5. МУЛЬТИАГЕНТНЫЕ ТЕХНОЛОГИИ ДЛЯ АДАПТИВНОГО УПРАВЛЕНИЯ

## 5.1. Начальные сведения

Для адаптивного управления сложными системами при возникновении различных событий эффективным оказывается использование мультиагентных технологий (МАТ) [10-13].

Это новое направление в области информационных технологий начало формироваться в 70–80-ые годы прошлого века на стыке достижений в областях:

- объектно-ориентированного программирования (ООП);
- искусственного интеллекта;
- параллельных вычислений;
- телекоммуникаций (Интернет).

На сайте Ассоциации AgentLink [14], объединяющей разработчиков мультиагентных систем (МАС) Европейского союза, представлена дорожная карта развития этого направления до 2030 г., имеющая девиз «Вычисления как взаимодействия» («Computing as Interactions»).

Этот девиз выражает главный смысл данной технологии, позволяющей от централизованных, монолитных и последовательных программ с заранее фиксированной иерархической структурой перейти к открытым распределенным сетевым сообществам автономных программ, работающих асинхронно и параллельно, способных самостоятельно формировать требуемые структуры и взаимодействовать между собой для решения поставленных задач.

Мультиагентные технологии открывают принципиально новые возможности решения сложных проблем, например, планирования и оптимизации ресурсов, которые плохо решаются или не решаются вовсе классическими методами, путем создания интеллектуальных систем нового класса, использующих фундаментальные принципы самоорганизации и эволюции, характерные для живых систем, например, колонии муравьев или роя пчел [15].

В этом подходе, часто называемом «биологическим» (bio-inspired), решение любой сложной задачи (например, построения сложного расписания флота грузовиков или станков и рабочих производственного цеха) строится путем самоорганизации, взаимодействия и переговоров большого числа (десятков и сотен тысяч) очень простых агентов, непрерывно конкурирующих и кооперирующихся друг с другом. При этом сам процесс решения сложной задачи имеет эволюционный характер (решение непрерывно самоулучшается в ходе работы системы) и напоминает метод «проб и ошибок» с той лишь разницей, что решения принимаются не случайным образом, а путем направленных согласованных улучшений. По сравнению с трудно реализуемым комбинаторным перебором, согласованное взаимодействие агентов не гарантирует нахождения единственного оптимального решения, но позволяет достаточно быстро добиваться решения задач с приемлемым качеством и при этом решать сколь угодно сложные проблемы.

В результате рассматриваемые интеллектуальные системы оказываются способны демонстрировать феномены «интеллекта роя» («Swarm Intelligence») и более сложного «коллективного интеллекта» («Collective Intelligence»), проявляющие себя в высокой эффективности, гибкости, живучести и т.д.

Следует отметить, что самоорганизующийся «интеллект роя» – важная альтернатива принятому в области искусственного интеллекта (ИИ) классическому пониманию интеллектуальной системы.

В традиционном понимании «интеллект» должен быть «собран» (как при сборке автомобиля) из отдельных умных блоков, например, таких как центральный блок управления, память, сознание, разные виды рассуждений, включая дедукцию, индукцию, ассоциацию и т.д. Однако в «интеллекте роя» нет никакого главного блока управления, отвечающего за «интеллект», напротив, интеллектуальное поведение рождается в результате взаимодействия большого числа самых простых элементов. Действительно, интеллектуальные возможности одного муравья или пчелы малы, но вместе рой пчел или колония муравьев представляют собой мощный организм с высокой степенью адаптивного интеллекта, позволяющего защищать гнездо от нападений, осваивать новые территории, находить пропитание и решать другие очень сложные задачи в условиях изменяющейся окружающей среды.

Именно поэтому МАТ и рассматривается как технологическая основа построения компьютерных моделей сложных адаптивных систем, необходимых для управления сложностью в реальной жизни.

## 5.2. Базовые определения

*Агент – это автономный программный объект, способный достигать поставленных целей в условиях неопределенности путем выработки и анализа вариантов принятия решений и согласованного взаимодействия с другими агентами.*

В настоящее время выделяются следующие *базовые свойства программных агентов*:

- автономность (autonomy) – способность функционировать без вмешательства других систем или человека для достижения поставленных целей, контролируя свои действия и внутреннее состояние;

- проактивность (proactivity) – агент демонстрирует управляемое целями поведение, проявляя инициативу для улучшения своего текущего состояния;
- реактивность (reactivity) – агент воспринимает внешнюю среду и реагирует на события изменений, адаптируя свое поведение для достижения целей;
- социальное поведение (social ability) – агент взаимодействует с другими агентами среды для согласования решений.

*Сильное определение агента* подразумевает дополнение только что перечисленных характеристик еще рядом свойств. В частности, главным из них является наличие у агента хотя бы некоторого подмножества так называемых «ментальных свойств» (или интенсиональных понятий), к которым относятся следующие:

- знания (knowledge) – постоянная часть знаний агента о себе, среде и других агентах, которая не изменяется в процессе его функционирования;
- убеждения (beliefs) – переменные знания агента о себе и среде, в частности, о других агентах (те знания, которые могут изменяться во времени и становиться неверными, однако агент может не иметь об этом информации и продолжать оставаться в убеждении, что на них можно основывать свои выводы);
- желания (desires) – состояния или ситуации, достижение которых, но разным причинам, является для агента желательным, однако они могут быть противоречивыми и потому агент не ожидает, что все они будут достигнуты;
- намерения (intentions) – то, что агент должен сделать в силу своих обязательств по отношению к другим агентам или то, что вытекает из его желаний (т.е. непротиворечивое подмножество желаний, выбранное по тем или иным причинам, которое совместимо с принятыми на себя обязательствами);

- цели (goals) – конкретное множество конечных и промежуточных состояний, достижение которых агент принял в качестве текущей стратегии поведения;
- обязательства по отношению к другим агентам (commitments) – задачи, которые агент берет на себя по просьбе (поручению) других агентов в рамках кооперативных целей или целей отдельных агентов при их сотрудничестве.

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.

Агенты, реализующие только базовые свойств, называются легкими. Агенты, основанные на сильном определении, называются тяжелыми (BDI-агенты (Beliefs-Desires-Intentions)), они сложнее в реализации, которая требует использования логического вывода с помощью логики предикатов первого порядка. Современное понимание отличительных особенностей мультиагентных систем от традиционных представлено в табл. 6.

Таблица 6. **Сравнение парадигмы традиционных и мультиагентных систем**

| **Традиционные системы** | **Мультиагентные системы** |
|---|---|
| Иерархии больших программ | Большие сети малых агентов |
| Последовательные вычисления | Параллельное вычисления |
| Инструкции сверху вниз | Переговоры равных сторон |
| Централизованные решения | Распределённые решения |
| Управляются данными | Управляются знаниями |
| Предсказуемость и повторяемость | Самоорганизация и эволюция |
| Стабильность и детерминизм | Развитие и недетерминизм |
| Тенденция уменьшать сложность | Тенденция наращивать сложность |
| Тотальный контроль | Создание условий для развития |

Подчеркнем, что в отличие от программного объекта, агент – это автономный программный объект, который имеет собственную цель и к ней стремится, в связи с чем его нельзя «вызвать» как обычный «метод» (иначе он потеряет свое текущее состояние и обязательства перед другими агентами). Агента можно лишь только «попросить» что-то сделать; но он согласится или откажет полученному запросу в зависимости от того, насколько он продвигается к своей цели, а также с учетом того, каковы его обязательства перед другими агентами и насколько возможно изменение этих договоренностей в текущей ситуации, что реализуется через выявление конфликтов и проведение переговоров с другими агентами.

## 5.3. Основные особенности агентно-ориентированного подхода

*Агентно-ориентированный подход* представляет собой развитие известного подхода, основанного на понятии объекта, но в то же время имеет ряд принципиальных отличий. Понятие «объект» представляет собой единицу программных систем, задаваемую некоторой структурой и алгоритмом взаимодействия. Объект имеет единое имя и свои собственные данные и процедуры. Объект может состоять из нескольких объектов, определенных таким же образом, и, в свою очередь, быть частью более крупного объекта. Все действия в ООП выполняются через сообщения (однако сообщения указывают объекту, *что* делать, но *не как* делать).

В целом, понятие объекта определяется с помощью 4 признаков:

- инкапсуляция;
- отношение «класс-подкласс»;
- свойство наследования;

- прохождение сообщений, допускающее определение полиморфных процедур, т.е. процедур, код которых может различаться в зависимости от приемника сообщения.

Объекты не могут анализировать свое поведение, определять характер своих связей с другими объектами или природу адресованных им сообщений. Их алгоритм получения сообщений сводится к вызову процедуры. А главное, они не могут самостоятельно формировать цели.

*Основное отличие агента от объекта* заключается в его автономности: объект сам должен выполнять целеполагание, чтобы обеспечивать свое эффективное существование. Наличие у агента алгоритма целеобразования обеспечивает принципиально новый уровень автономии. Это значит, что он необязательно выполняет распоряжения какого-либо другого агента или пользователя, а просто зависит от условий среды, включая цели и намерения других агентов. В отличие от объекта, агент может принять на себя определенные обязательства или, наоборот, отказаться от выполнения некоторой работы, мотивируя это отсутствием компетентности, занятостью другой задачей и т.п. В то же время, агент может выполнять такие действия как порождение, подавление и замена других агентов, активизация функций (как своих, так и у других агентов), активизация сценария деятельности, запоминание текущего состояния других агентов и другое.

Для достижения поставленной цели агент должен уметь реагировать на события, решать различные задачи, принимать решения и коммуницировать с себе подобными для их согласования. Например, агент заказа на грузоперевозку должен уметь находить подходящие грузовики, бронировать самый подходящий грузовик, а потом следить за его перемещениями и контролировать выполнение заказа в срок, готовить отчет по поездке с расчетом себестоимости перевозки. Агент грузовика должен уметь работать с заказами,

строить маршруты, подбирать водителей, находить места заправки, прогнозировать появление заказов в городах и решать ряд других задач.

В условиях, когда вдруг случается незапланированное событие поломки грузовика, агент грузовика должен уметь найти все размещенные на нем заказы и информировать их о таком изменении. В ответ агенты заказов должны уметь заново подыскать себе другие грузовики, и, наоборот, при высвобождении нового, более выгодного грузовика, агент заказа, очевидно, должен иметь возможность забронировать этот грузовик, разорвав связь по бронированию с предыдущим грузовиком, если есть возможность улучшить результаты.

Такого рода взаимодействия агентов описываются специальными протоколами, которые регламентируют асинхронные или синхронные схемы развития «диалога» между агентами, включая информирование об изменениях, посылку различного рода предложений и реакцию на эти предложения и другие. При этом пересылка сообщения от первого агента ко второму агенту должна быть осуществлена в любой момент времени, вне зависимости от состояния второго агента.

Таким образом, агент – это не просто «алгоритм», это обычно набор сценариев поведения и взаимодействия, управляемых центральным блоком планирования действий и анализа результатов, иногда называемым «личностью» агента (personality). С некоторой степенью упрощения можно считать, что агент представляет собой «машину состояний» подобно конечному автомату или автомату с памятью, в которой любое событие на входе переводит агента из одного состояния – в другое. Но на практике агент более похож на небольшую операционную систему, обеспечивающую автономную работу в модельном или реальном времени.

Подведем первые итоги обсуждения понятия «агент», отметим важные свойства программных агентов, каждый из которых должен уметь:

- следовать поставленной цели и выбирать способы ее достижения;
- реагировать на события, изменения ситуации в среде путем мониторинга самой среды или получая сигналы (сообщения);
- обращаться к встроенным сценариям возможных действий или в базу знаний, чтобы определить, какую задачу необходимо выполнить, и как это сделать;
- отправлять сообщения другим агентам или пользователям, а также получать от них сообщения;
- решать задачи, необходимые для принятия решений;
- анализировать получаемую информацию, вырабатывать и сопоставлять варианты решений, принимать решения и согласовывать с другими агентами;
- устанавливать и разрывать связи с другими агентами;
- оценивать эффективность своих решений и работы в целом.

Агенты могут активироваться как при возникновении событий, так и быть постоянно или временно активными, переходить из пассивного состояния в активное и наоборот, а также быть проактивными, т.е. самостоятельно, без внешних побуждающих воздействий, искать возможности увеличения ценности своих решений.

*Ценность решения – совокупный показатель эффективности работы агента любого элемента сети предприятия (или предприятия в целом), который включает в себя такие показатели, как прибыль, качество или эффективность обслуживания, себестоимость услуги или товара, риски или любые другие.*

Определение ценности решения зависит от специфики работы предприятия и даже для двух похожих компаний грузовых перево-

зок подходы к принятию решений могут существенно отличаться, например, если одна старается всегда возвращать свои грузовики на базу в гараж, а вторая использует модель «челнока», который должен делать перевозки без остановки и возвратов на базу.

Интеллектуальные агенты могут использовать не только готовые «жесткие» сценарии, но и применять динамически загружаемые или создаваемые «на лету» новые сценарии действий на основе базы знаний о предметной области.

*Мультиагентная система (МАС) – система, состоящая из одной или более групп агентов, конкурирующих или сотрудничающих друг с другом с целью выполнения общей задачи таким образом, чтобы увеличить ценность принимаемых решений для всей системы (например, предприятия в целом).*

*Поведение МАС определяется не одним детерминированным алгоритмом, а формируется эволюционным путем из взаимодействия составляющих ее агентов*.

В МАС агенты всегда совместно работают в сообществах, часто называемых также мирами, роями, командами или группами. Мультиагентные системы могут включать или создавать одну или несколько групп агентов, каждой группе можно поручить выполнение отдельного задания, одно задание также может быть разделено на части и распределено по разным группам.

Мультиагентные системы обладают всеми характеристиками сложных адаптивных систем, в частности, самоорганизацией, которая в контексте мультиагентной технологии может быть определена следующим образом:

*Самоорганизация в МАС – способность группы агентов самостоятельно изменять существующие и / или устанавливать новые отношения между состоящими в ней агентами с целью решения новых задач, восстановления после сбоя и максимизации ценности решений для системы*.

На следующем этапе получит развитие *organization-based programming* – создание МАС в терминах *моделей организаций агентов* (пример таких организаций – ПВ-сети, п. 5.5). В этом подходе под моделью организации понимаются созданные заранее классы агентов и их роли, протоколы взаимодействий и т.д. Взаимодействие агентов строится на основе ролей. *Oragent model* не только позволяет представлять системы с высоким уровнем абстракции в терминах организаций агентов, правил, протоколов и ролей, но и обеспечивает реализацию концепций инкапсуляции, наследования, управления событиями для повышения динамичности и гибкости МАС. Пользователю будет предоставлена возможность купить виртуальный мир – модель работы магазина, фабрики и т.д.

Таким образом, одним из важных применений мультиагентных систем является адаптивное распределение ресурсов в условиях, когда число заказов (потребностей в ресурсах) и / или количество ресурсов, которые необходимо распределить, непредсказуемо меняется во время самого процесса распределения и исполнения заказов, что существенно отличает данную постановку от классической.

## 5.4. Холонический подход к созданию сложных систем

Создание МАС для адаптивного управления ресурсами становится возможным на основе холонического подхода, базирующихся на работах Артура Кестлера, который изучал феномен образования биоценозов в живой природе.

*Холон* (от греческого «holos» – весь, целый, с суффиксом «on», обозначающий часть, частицу) – это элемент (частица), соединяющий в себе свойства целого и части. Холоническая система состоит

из сети холонов, каждый из которых может быть элементарным или составным холоном.

Фактически, в своих работах А.Кестлер предсказал открытые многоуровневые сетевые самоподобные (рекурсивные) структуры из автономных элементов, построенные по аналогии с клетками живого организма, где сложное взаимодействие и взаимопроникновение может осуществляться на любом уровне.

В холоническом подходе выделяются 4 базовых типа агента:

- агент заказа;
- агент ресурса;
- агент продукта;
- штабной агент.

Например, агент ресурса крупного завода может быть представлен как вложенная сеть агентов цехов, каждый из которых, в свою очередь, может быть представлен сетью агентов участков и далее – до агента рабочего.

Более того, сам агент рабочего, на еще более низком уровне рассмотрения, может иметь собственные заказы на отдельные операции (как часть заказов цеха), свои ресурсы (доступные для планирования интервалы его рабочего времени), продукты (входные детали и материалы и результаты сборочных операций) и даже штабного агента, управляющего стратегиями планирования времени рабочего, прогнозирующего загрузку на следующую неделю, предлагающего освоить более востребованные в перспективе специальности и т.п.

Разработка холонических архитектур, связанных с выявлением и выделением отдельных холонов (миров) и организацией их взаимодействия, в последнее время становится одним из центральных вопросов проектирования мультиагентных систем, позволяя искать компромисс между гибкостью и производительностью системы.

Кратко охарактеризуем цели и задачи базовых классов агентов на примере решения проблем производственной логистики:

- *агент заказа* – должен обеспечить наилучший вариант выполнения своего заказа на имеющихся ресурсах при наилучшем качестве проведения работ, минимальной стоимости и сроках выполнения. Для этого он должен иметь доступ к знаниям о структуре заказа, технологии исполнения операций заказа, требования к рабочим и станкам по каждой операции, материалам и инструментам и т.д.;
- агент ресурса – должен обеспечить максимальную загрузку своего ресурса с учетом знаний о его специализации, производительности, плане регламентных работ и других особенностях;
- агент продукта – должен обеспечить минимальное время хранения продукта на складе с учетом знаний о структуре изделий, местах его производства и хранения, параметрах изделий;
- штабной агент – должен обеспечивать постоянный анализ ситуации в работе системы (предприятия) и вмешиваться в работу других агентов в случае возникновения критических ситуаций, например, изменяя стратегии поведения агентов нижнего уровня, или для обеспечения развития.

Перечислим примеры других задач, требующих адаптивного распределения ресурсов, в дополнение к производственной логистике:

- *транспортная логистика* – распределение грузовиков по заказам в пространстве и времени в условиях неопределенности, возникающей из-за часто происходящих непредсказуемых событий (изменения в заказах, задержки грузовика в пути, дорожно-транспортные происшествия, поломки транспорта, задержки в разгрузке-погрузке и др.);
- управление проектами – распределение исполнителей по задачам проектов с учетом их квалификации и личного опыта,

текущей загрузки подразделений, особенностей задач, требований задач к квалификациям и опыту сотрудников;

- электронная коммерция – распределение имеющихся товаров или услуг по запросам клиентов, при условии, что заказчик или поставщик могут неожиданно прервать процесс распределения и отказаться от заказа, потребовать замены товара, расширить заказ и т.д.

Каждая из указанных выше проблем может решаться с использованием рассмотренных классов агентов, например, в управлении проектами также имеются агенты заказов на решение отдельных задач проектов, агенты ресурсов исполнителей, знающие их квалификацию, агенты продуктов, создаваемых в ходе выполнения проектов, например, в виде документа или программной компоненты, и штабные агенты, действующие, например, от лица руководителя проекта или руководителя подразделения, сотрудники которого привлекаются к проектам и для которых важно обеспечить равномерную загрузку с учетом согласованного ранее плана отпусков.

Разработанную архитектуру мира агентов заданной предметной области, в которой специфицируются классы агентов и бизнес-процессы (протоколы) их переговоров, а также роли и ответственности, целевые установки агентов, по типу рассмотренных в этом разделе, можно рассматривать как *модель организации*.

Создание моделей таких «организаций» (organization-based programming) для различных предметных областей рассматривается как следующий шаг в развитии объектно-ориентированного (object-oriented programming) и мультиагентного программирования (multi-agent programming).

## 5.5. Сети потребностей и возможностей

Развитием холонического подхода являются сети потребностей и возможностей (ПВ-сети) [12], в которых вместе с агентами заказов, ресурсов, продуктов и штабного дополнительно используются *агенты потребностей и возможностей.*

Эти агенты могут выступать в качестве субагентов (зависимых агентов, помощников) основных агентов, позволяя указанным выше агентам параллельно и асинхронно постоянно искать друг друга на виртуальном рынке мультиагентной системы, выполняя непрерывный поиск соответствия между ними или *матчинг* (matching), как показано на рис. 15.

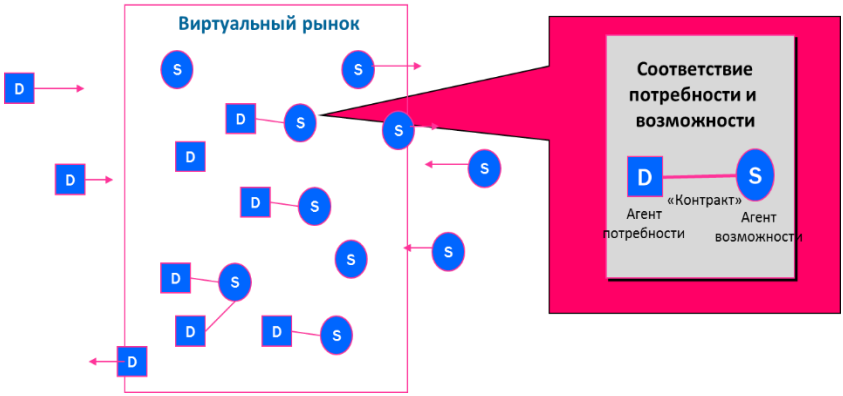



Рис. 15. Виртуальный рынок агентов ПВ-сети

Задача агентов потребностей D (Demand Agent) – найти возможности (ресурсы) для удовлетворения потребностей, а задача агентов возможностей S (Supply Agent) состоит в том, чтобы найти потребности для использования своих возможностей. В случае успешного поиска D и S агенты могут заключать или пересматривать контракты на требуемые услуги на виртуальном рынке си-

стемы с выплатой бонусов в случае успешного их выполнения или штрафов в случае разрыва связей, а также компенсаций за ухудшение позиций при уступках в ходе переговоров. Постоянный поиск соответствий между конкурирующими и кооперирующимися агентами потребностей и возможностей на виртуальном рынке системы позволяет строить решение любой сложной задачи как динамическую сеть связей между задачами (операциями), гибко изменяемую в реальном времени.

Так, на нижнем уровне задача (операция) может создавать потребность в интервале времени, а сам интервал времени ресурса (слот) – предлагать возможность исполнения для задачи, связка между ними и задает базовый элемент ПВ-сети для любого сколь угодно сложного расписания.

Задача штабного агента, действующего от лица предприятия в целом, заключается в том, чтобы гарантировать максимизацию ценности решений в ПВ-сети предприятия в процессе распределения возможностей по потребностям.

Работа ПВ-сети должна обеспечить реализацию полного цикла управления ресурсами, представленного на рис. 16:

- *Реакция на событие* – предполагает определенную политику обработки событий, часть которых может автоматически поступать на обработку, а часть – требует участия пользователя.
- *Распределение ресурсов* – решается задача, какие ресурсы следует использовать для отработки заказов, учитывая возможно разный размер грузовика, возможность использовать сверхурочных и т.д.
- *Планирование* – формирование расписания использования ресурсов.
- *Оптимизация* (пока есть время) – может включать моделирование и прогнозирование развития ситуаций.

- *Согласование решений* – предполагает взаимодействие с пользователем, который может утвердить предлагаемое системой решение, отменить или скорректировать решение, ввести собственное встречное предложение.
- *Перепланирование* в случае расхождения плана и факта – автоматически или по указанию пользователя запускает цепочку изменений расписания.
- *Обучение из опыта* – индуктивные обобщения, например, не назначать на важные заказы водителя, который всегда опаздывает.

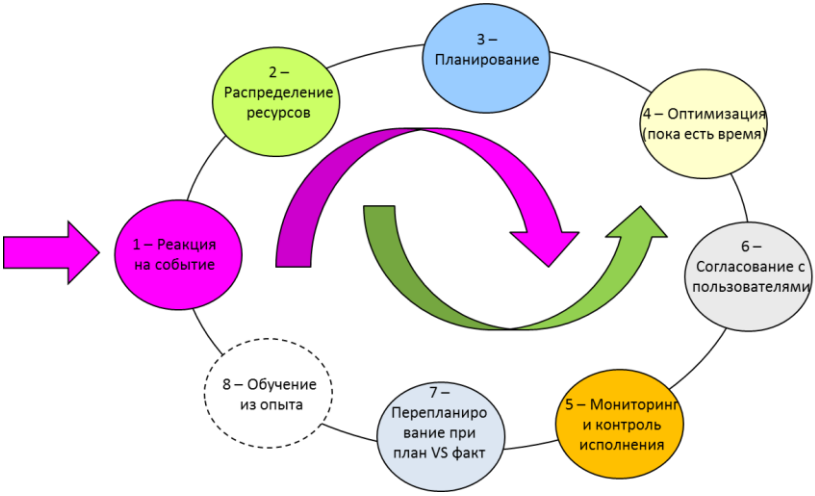



Рис. 16. Полный цикл управления ресурсами

Такой полный цикл управления ресурсами обеспечивает поддержку процесса автономного функционирования интеллектуальной системы управления ресурсами в любой предметной области.

## 5.6. Пример мира транспортной логистики

Рассмотрим подробнее особенности построения ПВ-сетей на примере мира транспортной логистики с консолидациями грузов (рис. 17).

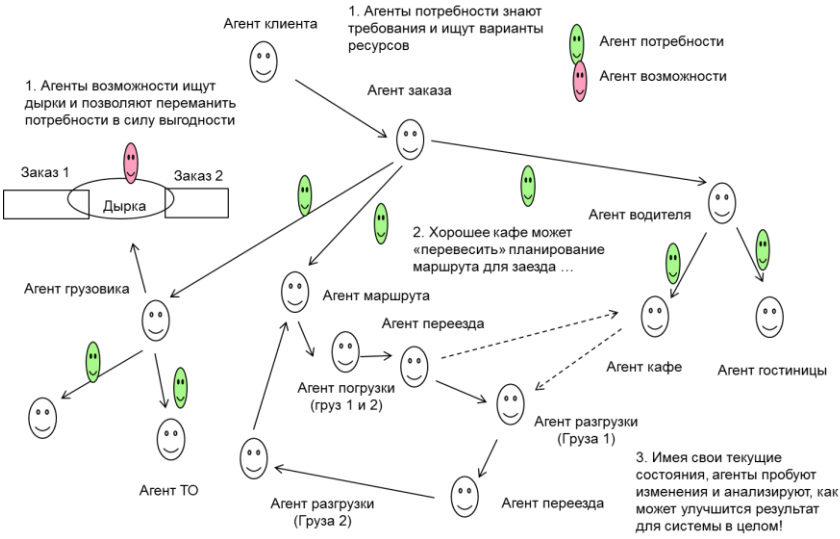



Рис. 17. Пример конструкции ПВ-сети мира транспортной логистики

В мире транспортной логистики можно выделить следующих агентов:

- агент клиента (договора) – заботится о выполнении всего объема заказов с заданным уровнем сервиса;
- агент заказа – пытается выполнить заказ в заданные сроки, задает исходные пункты для забора грузов и пункты назначения;
- агент грузовика – старается максимально использовать грузовик для выполнения заказов;

- агент груза – обеспечивает совместимость с другими грузами;
- агент маршрута – решает задачу поиска лучшего маршрута для движения;
- агент консолидаций – пытается создать динамические группы грузов одного маршрута движения для максимизации загрузок грузовиков;
- агент водителя – обеспечивает рабочий график водителя с учетом его квалификации, опыта и требований к режиму труда и отдыха;
- агент заправки – добивается наиболее рационального плана заправок по маршруту следования;
- агент кафе и гостиницы – предлагают наилучшие места остановок при движении по маршруту.

В рассматриваемом примере агенты потребностей могут создаваться агентом грузовика для одновременного, асинхронного и параллельного поиска маршрута движения, водителя, места и времени проведения технического обслуживания (ТО), места заправки, отдыха и т.д.

И, встречно, агент возможности может порождаться тем же агентом грузовика, когда, например, в маршруте движения грузовика обнаруживается значительный пустой пробег и нужно до срока выезда по пустому перегону найти заказ, который бы позволил грузовику перевезти груз в этот конкретный интервал, между двумя уже сложившимися поездками.

Такой агент возможности будет атаковать своими предложениями уже размещенные заказы на других грузовиках, поскольку цена его предложения будет наверняка ниже (в модели Shared costs – разделяемых затрат), т.е. будет высоко конкурентной, и стадия проактивности, скорее всего, принесет этому агенту хорошие результаты.

Фактически, агентам потребностей и возможностей передаются требования на поиск соответствующих агентов и определенным образом распределяются финансовые ресурсы, на основе которых эти агенты, действуя от лица и по поручению агента грузовика, ищут соответствующие ресурсы (водителя, заправку, станцию ТО и др.).

Пример модели конкретной ситуации, называемый далее *сценой мира* транспортной логистики, приведен на рис. 18.

Здесь показаны несколько конкретных заказов 1192 и 1205 от одного из клиентов, один составной заказ из Санкт-Петербурга через Москву в Самару и другой из Перми в Екатеринбург и Нижний Новгород. Для описания данной текущей ситуации потребуется спецификация таких объектов как «клиент», «заказ», «грузовик», «город», «дорога», «маршрут», а также связывающих их отношений «заказ-принадлежит-клиенту», «поездка-входит-в маршрут» и другие.

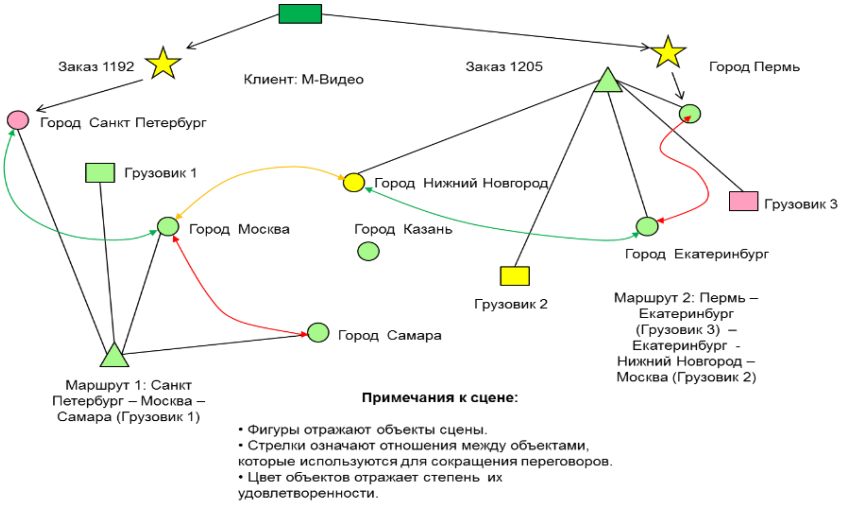



Рис. 18. Пример сцены ПВ-сети мира транспортной логистики

Аналогичные примеры можно легко привести для других сфер применения, например, в цехах на фабриках это продукция, инструменты, станки и рабочие, ТО станков и т.д.

Наличие сцены мира позволяет существенно снизить издержки на переговоры, поскольку каждый агент может прослеживать связи между элементами системы и использовать эту информацию в ходе выработки и принятия решений.

Разработанный подход к созданию ПВ-сетей был успешно применен для ряда промышленных приложений [12–15].

## 5.7. Модели микроэкономики агентов

Согласно предлагаемому подходу, любому реальному предприятию в соответствие может быть поставлена его действующая мультиагентная компьютерная модель (ПВ-сеть), организованная как мир виртуального рынка, в котором агенты имеют собственные счета и накопления, покупают и продают услуги, а также платят налоги по заключению контракта, на коммуникации и другие.

При этом уровень детализации основных структур предприятия может простираться не только до подразделений или отдельных людей, но делать активными любую мельчайшую физическую или абстрактную сущность предприятия. Так, в этой модели не только мастер участка или рабочий, но и каждый станок может заботиться о своей загрузке, не надеясь на человека. Каждая деталь может беспокоиться о своем длительном нахождении на складе и искать варианты скорее войти в следующий заказ, чтобы цена не стала слишком высокой. Аналогично, могут становиться активными и абстрактные сущности: заключенный договор напомнит о прошедшем сроке оплаты, денежные поступления запустят пересчет прибыли по проекту, а затраты найдут свою графу расходов.

В результате в предлагаемой модели предприятия вместо отдельных людей (менеджеров, диспетчеров, мастеров, рабочих, водителей и др.), которым всегда слишком трудно уследить за меняющейся ситуацией, начинает моделироваться и работать в реальном времени сообщество агентов на виртуальном рынке, в котором лицами, принимающими и согласовывающими решения, становятся любые физические или абстрактные сущности, включая станок, деталь и т.д. Все эти агенты на виртуальном рынке являются равными в правах и могут выполнять свои задачи только через проведение коммерческих переговоров и заключение сделок с другими агентами, но в отдельных случаях – и с пользователями.

Виртуальные деньги являются прямым аналогом энергии, питающей процессы самоорганизации в природных или социальных системах. Заказ или ресурс, обладающий большей энергией, может получить больше вариантов принятия решений, запустить более длинную цепочку изменений при формировании результата и т.д. Налоги, собираемые с агентов, в свою очередь, могут динамически меняться от ситуации и служить для управления процессами на виртуальном рынке или быть использованы системой для точечных инвестиций в «узкие места» результирующего решения для его улучшения.

Важной особенностью *микроэкономики виртуального рынка* является возможность адаптивного пересмотра ранее заключенных договоренностей между агентами с использованием *метода компенсаций*. Основной принцип при этом заключается в том, что, если для нового заказа невозможно найти подходящие свободные ресурсы, то его агент может сделать предложение агенту уже занятого ресурса, которое будет включать динамически формируемую компенсацию за отмену предыдущего назначения. Такое предложение может вызвать волну новых переговоров, включая переговоры об отмене установленного до этого назначения ресурса на заказ и по-

иск нового ресурса для освободившегося заказа. Волна этих процессов может распространиться дальше и на несколько ранее сделанных назначений. Виртуальные деньги, доступные для оплаты компенсаций в этой цепочке переговоров, снимаются со счетов агентов, которые предлагают провести переназначение, что существенно сокращает перебор вариантов и позволяет «волне» быстро затухать. В исключительных случаях, когда заказ прибывает от VIP-клиента, агент компании может не взимать компенсацию, чтобы гарантировать выполнение VIP-заказа, даже если это обернется затратами для предприятия, или наоборот, стимулировать длинные волны переговоров, если у системы есть большой запас времени на поиск лучших решений.

Ключевыми понятиями в *микроэкономике принятия решений* каждого агента являются:

- ценность решения, отражающая прирост удовлетворенности агента результатом (рост прибыли, охвата рынка, снижение затрат, увеличение качества обслуживания, удовлетворенности клиентов, благосостояния сотрудников предприятия и т.д.);
- стоимость решения, которую агент (предприятие) готов заплатить за улучшение ценности решения.

Для работы агентов в ПВ-сети предлагаются различные модели микроэкономики агентов, учитывающие удовлетворенность агентов, предлагающие разные механизмы расчета виртуальными деньгами, принимающие во внимание силу связей и т.д.

В настоящее время выделяются 2 основных типа микроэкономики агентов на виртуальном рынке:

- *микроэкономика 1-го рода* – физическая стоимость производства или транспортировки продукции, которая в смете расходов любого рассматриваемого предприятия обычно

моделируется прямыми расходами (например, амортизация, зарплата водителя и т.д.);

- *микроэкономика 2-го рода* – стоимость работы «управленческого офиса» агентов: диспетчеров, менеджеров, логистов и других специалистов по поиску решений (например, по построению расписания), которая обычно скрыта в накладных расходах предприятия.

Для пояснения этих типов микроэкономики рассмотрим пример мультиагентной системы для грузовых перевозок в реальном времени. Легко представить себе производственное здание такой транспортной компании размером в 2 этажа, в котором на первом этаже размещается сам гараж с машинами, а на втором – офис, куда звонят или куда приходят клиенты. Тогда микроэкономика 1 рода определит стоимость работы грузовиков из гаража компании (первый этаж), а микроэкономика 2 рода – работу ее офиса (второй этаж), принимающего заказы, ведущего переговоры с заказчиками, строящего маршруты и расписания, выдающего накладные и путевые листы водителям, рассчитывающим их зарплаты и т.д.

Заметим, что в некоторых случаях в себестоимости исполнения некоторого заказа на перевозку груза на короткую дистанцию стоимость его планирования может быть даже больше, чем стоимость физической перевозки.

При этом обычно рассматривается две модели учета расходов – на основании фиксированных расходов (fixed costs), когда стоимость ресурса для заказа рассчитывается по заданному тарифу, или с разделяемыми затратами (shared costs), когда стоимость динамически определяется от загрузки ресурса и может снижаться с ростом загрузки, т.к. все заказы делят между собой общую стоимость ресурса (например, грузовика при перевозке).

Для управления процессами самоорганизации агентов в системе, направленными на поиск решений, как уже отмечалось выше, могут

быть введены налоги, которые могут взиматься за бронирование ресурса или заключение контракта, нахождение элемента сети в расписании, переговоры (коммуникации) агентов, перестройку расписания, установление или разрыв связи и т.д. Кроме того, налоги могут быть динамическими, например, налог на перепланирование может возрастать при приближении к времени отправления грузовика, чтобы постепенно «замораживать» расписание.

Заметим, что при этом в системе всегда имеется и может быть в любой момент отображена полная индивидуальная информация по каждому заказу, продукту или ресурсу, например, для продукта можно увидеть, где он был произведен, сколько времени он провел на складе, сколько это стоило, как часто он перемещался и т.д.

Данная информация является ключевой для анализа продуктивности и эффективности любого предприятия и принятия управленческих решений в реальном времени.

## 5.8. Как достигать качества решений с ростом сложности и динамики?

Здесь мы подходим к ответу на главный вопрос – зачем следует применять мультиагентные технологии для управления ресурсами?

*Главный смысл применения мультиагентных технологий для решения задачи управления ресурсами состоит в том, чтобы обеспечить высокое качество решения, которое должно приближаться или превосходить качество решения данной проблемы человеком («To Be Better than Humans»), причем в особенности при принятии решений по событиям в реальном времени.*

В этих целях для построения плана (расписания) группы ресурсов в любой предметной области предлагается, прежде всего, создавать мультиагентную модель данного предметного мира, в которую

включать агентов, отражающих все многообразие интересов, влияющих на качество решения. Очевидно, что наилучшим (разумным) решением, отражающим глубокий здравый смысл, при этом будет считаться решение проблемы, в которой достигнут баланс интересов всех участников (консенсус) в пределах заданных ограничений. Подчеркнем, что в предлагаемых методах управления ресурсами каждый агент может стараться полностью «эгоистично» добиться результата, но при столкновении своих интересов с другими агентами должен путем переговоров и взаимных уступок решать возникающие конфликты в интересах системы (предприятия) в целом.

Одной из трудностей в принятии решений при наличии многих параметров в системе является определение того, какие из этих параметров особенно важны в данный момент для принятия решения, а какими можно пожертвовать, и до какой степени?

Еще одна трудность в принятии решений по управлению ресурсами состоит в том, что приоритеты или веса критериев принятия решений часто зависят от ситуации и могут динамически меняться в ходе принятия решений. Например, если план по прибыли уже выполнен, график движения может быть менее напряженным и можно дать водителям больше времени для отдыха.

В нашем сложном и динамичном мире следует всегда исходить из того, что сложность в будущем будет только нарастать и создав однажды мультиагентную систему управления своими ресурсами, заказчик обязательно пойдет дальше и захочет учитывать при планировании все большее число факторов. Ввод каждого нового фактора, и соответствующего ему агента, будет каждый раз еще более усложнять поиск баланса интересов, но и одновременно, все больше приближать качество решения к реальности, к качеству решений, достигаемых хорошо подготовленными и опытными диспетчерами.

*Важным преимуществом использования мультиагентных технологий является ситуационный поход к принятию решений.*

При этом становятся необходимы некоторые «счетчики» в логике работы агентов, аккумулирующие (накапливающие) информацию о поездках, которые затем используются в решающем пороговом правиле, результат которого каждый раз будет зависеть от истории и текущей ситуации.

В этих целях выигрыши по разным критериям должны быть сопоставимы в рамках единой валюты, которой и выступают виртуальные деньги. При этом также может быть использован вес критериев, который помогает управлять важностью (приоритетами). Таким образом, принятие во внимание интересов всех участников делает процесс принятия и согласования решений более сложным, но с другой стороны, открывает новые возможности по повышению качества и эффективности решений для предприятия в целом.

*Еще одним важным преимуществом использования мультиагентных технологий для управления ресурсами является возможность работы в реальном времени, когда качество и эффективность принимаемых решений напрямую зависит от момента времени.*

Рассмотрим простой пример: грузовик из Екатеринбурга разгружается в Казани и планирует идти за следующим заказом в Нижний Новгород. В этот момент приходит заказ из Казани в Екатеринбург. Система решает перебросить данный заказ на данную машину, а на заказ в Нижнем Новгороде передать другую скоро освобождающуюся машину из Самары. Очевидно, что через пару часов принимать такое решение будет поздно, ведь машины уйдут в очередные рейсы – надо принимать решение здесь и сейчас.

В результате, в рамках ПВ-сети динамически самоорганизуется многофакторный план распределения возможностей по потребностям, открытый для любых изменений. При возникновении нового

события может быть начата адаптивная перестройка связей агентов потребностей и возможностей в ПВ-сети, причем только тех, кто оказался зависим от события, которая заканчивается, когда найден новый консенсус между агентами, и ни один агент не может более улучшить ситуацию (*динамический останов*).

Тогда решение может быть выдано пользователям для окончательного принятия, либо для интерактивной или ручной доработки.

Данный подход позволяет создать действующую компьютерную модель предприятия как совершенно новую модель непрерывно самоорганизующегося социально-экономического «организма» из согласованно действующих агентов, устроенную подобно рою пчел или колонии муравьев, открытую к изменениям, гибкую и эффективную, производительную, надежную и живучую.

Предлагаемый подход становится все более востребован современными предприятиями для повышения эффективности использования ресурсов за счет перехода к управлению ресурсами в реальном времени.

# 6. МЕТОДЫ И СРЕДСТВА ПОСТРОЕНИЯ МУЛЬТИАГЕНТНЫХ СИСТЕМ

## 6.1. Метод компенсаций для адаптивного планирования ресурсов в ПВ-сетях

Дадим более формализованное описание рассмотренного выше метода, подробно описанного на примерах из реальной жизни грузовой компании.

Предположим, что перед мультиагентной системой стоит задача распределить число $n$ ресурсов по $m$ потребностям.

Каждый ресурс характеризуется:

- набором $\boldsymbol{f}$ атрибутов;
- стоимостью ресурса, выраженной в денежных единицах (д.е.).

Каждая потребность характеризуется:

- набором $\boldsymbol{g}$ атрибутов;
- покупательной способностью потребности, выраженной в денежных единицах (д.е.).

Как правило, потребности поступают в систему одна за другой, причем время их поступления и их характеристики являются непредсказуемыми.

Количество и характеристики ресурсов могут быть постоянными или меняющимися с течением времени.

В этих условиях процесс распределения будет выглядеть следующим образом:

1. Когда в систему поступает потребность, ей присваивается Агент, который отправляет сообщение всем агентам имеющихся ре-

сурсов о том, что ему требуется ресурс с определенными характеристиками, и он может заплатить за этот ресурс определенное количество денежных единиц.

2. Все агенты ресурсов, обладающие всеми необходимыми характеристиками (или частью из них, но имеющие стоимость, равную или меньшую указанного количества д. е.), предлагают эти ресурсы Агенту потребности.

3. Агент потребности выбирает наиболее подходящий вариант из предложенных ему ресурсов.

4. Если все подходящие ресурсы оказываются занятыми, Агент потребности пытается получить ресурс, который уже привязан к другой потребности, предлагая назначенному ей Агенту рассчитать требуемую компенсацию.

5. Агент потребности, который получает предложение о компенсации, рассматривает его, но принимает его, только если размер компенсации позволяет ему впоследствии приобрести другие подходящие ресурсы и одновременно увеличить ценность решения. Если он принимает предложение, то первый упомянутый Агент потребности реорганизует связи в системе: ранее установленная связь потребности с ресурсом разрушается, и вместо этого устанавливается новая связь другой потребности со свободным ресурсом, если данная операция увеличивает ценность решения. Это – пример самоорганизации с внутренним изменением связей.

6. Процесс повторяется до тех пор, пока все возможности не будут связаны с потребностями, и изменение существующих связей более не позволит увеличить ценность решения в пределах заданного диапазона изменений или до тех пор, пока не истечет интервал времени, отведенного на решение задачи.

7. Перед каждым действием агенты могут обращаться к базе знаний, чтобы получить информацию о том, какие действия они могут предпринимать.

Этот процесс определяет метод самоорганизации, позволяющий организовать решение сложных задач в любой открытой системе, от транспортной и производственной логистики – до кластеризации и понимания смысла фраз и предложений на естественном языке.

**Пример самоорганизации в транспортной логистике**

Рассмотрим пример из области грузовых авиаперевозок, где существуют: а) сеть грузовых терминалов; б) маршруты между этими терминалами; в) два грузовых воздушных судна, которые транспортируют груз по этим маршрутам (рис. 19).

Один из грузовых самолетов обладает большой грузоподъемностью (20 тонн), но небольшой скоростью (500 км в час); его эксплуатационные расходы довольно высоки – 7 денежных единиц на километр (д.е./км).

У другого грузового самолета грузоподъемность меньше (10 тонн), но скорость выше (750 км в час). Его эксплуатационные расходы составляют 3 д.е./км.

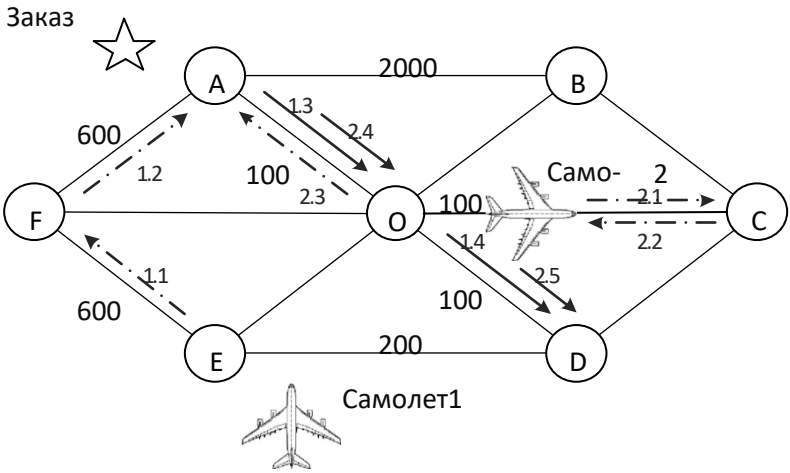



Рис. 19. Транспортная сеть грузовых авиаперевозок
(цифры отражают расстояния)

*Ситуация 1*. Пусть система получает заказ на транспортировку 5-тонного груза из точки A в точку D (расстояние в 2000 километров) за 30 000 д.е. Срок доставки – 1 день. Самолет 1 свободен и находится в терминале E, в то время как Самолет 2 транспортирует ранее принятый заказ к терминалу C.

Чтобы спланировать работу, создается Агент заказа 1. Агент отправляет обоим самолетам подробные данные по заказу на транспортировку. Рассмотрев эту информацию, Агент каждого самолета отправляет предложение Агенту Заказа 1.

Самолет 1 сразу готов начать транспортировку. Чтобы принять заказ, самолет должен вылететь от терминала E к терминалу A (1200 км). Стоимость этой операции: (1200 + 2000) *7 = 224000 д.е.; время, требуемое для ее выполнения: (1200+2000)/500=6.4 часа.

Самолет 2 должен будет сначала завершить текущую транспортировку, а затем отправиться к терминалу A, чтобы взять на себя выполнение нового заказа. Если Самолет 2 в настоящее время находится в точке O, ему нужно будет пройти маршрут O – C – O – A. Время, которое для этого потребуется: (1000+1000+1000+2000)/750= =6.67 часа, т.е., также укладывается в 1 день, и стоимость составит: (1000+1000+2000) * 3 = 12000 д.е. Прибыль Самолета 2 и Самолета 1 составит 18 000 д.е. и 7600 д.е. соответственно.

Учитывая, что остальные условия равны, Агент заказа 1 выберет самолет, который максимизирует прибыль предприятия, в нашем случае это – Самолет 2 несмотря на то, что путь до точки A займет у него больше времени. В результате этого решения в системе была создана первая связь между заказом (Заказ 1) и ресурсом (Самолет 2), как показано на рис. 20.

Связь 5 указывает на то, что для Заказа 1 был выбран Самолет 2.

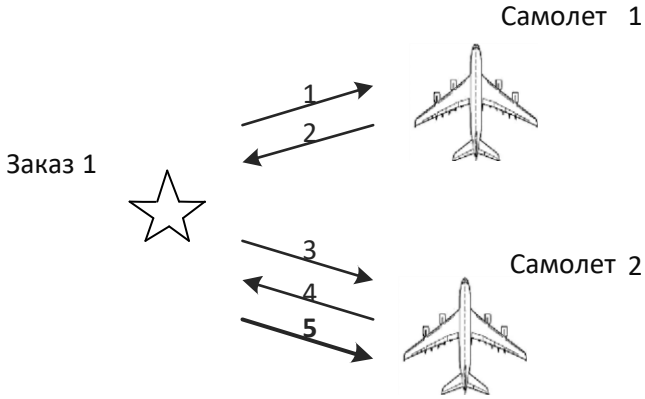



Рис. 20. Агент Заказа 1 находит два доступных самолета
и выбирает Самолет 2 для выполнения своего заказа

Установленные связи имеют следующий смысл:

- 1 – Агент Заказа 1 отправляет данные о заказе Агенту Самолета 1;
- 2 – Агент Самолета 1 отправляет предложение Агенту Заказа 1;
- 3 – Агент Заказа 1 отправляет данные по заказу Агенту Самолета 2;
- 4 – Агент Самолета 2 отправляет предложение Агенту Заказа 1;
- 5 – Самолет 2 назначается на Заказ 1.

Ситуация 2. Предположим, что несколько минут спустя системой был получен новый заказ на грузовую транспортировку: 2-х-тонный груз необходимо доставить из точки С в точку В (600 км) за 20 000 д.е. через 6 часов в этот же день.

Для обработки этой задачи создается Агент Заказа 2.

В данной ситуации очевидно, что только Самолет 2 может выполнить этот заказ: Самолет 1 не может выполнить работу в срок,

поскольку ему необходимо (2000 + 1200)/500 =6.4 часа. А Самолету 2 было бы достаточно (1000 + 1200) / 750 =2.9 часа.

Однако Самолет 2 был уже назначен на Заказ 1.

В этой ситуации запрос Агента Заказа 2 Агенту Самолета 2 создает следующую цепочку переговоров (рис. 21):

- Агент Самолета 2 отправляет сообщение Агенту Заказа 1, в котором предлагает отменить назначение на Заказ 1 для выполнения Заказа 2, и готов выплатить неустойку за срыв уже данных обязательств.
- Агент Заказа 1 проверяет возможные варианты. Он снова связывается с самолетами (6) и получает от Самолета 1 то же предложение (7) с выполнением работы в срок, которое было ранее отклонено т.к. оно было менее выгодным (7600 д.е. по сравнению с 18 000).
- Агент Заказа 1 устанавливает Агенту Самолета 2 размер неустойки (1600 д.е.) (8).
- Самолет 2 принимает информацию к сведению (9).
- Агент Заказа 2 рассматривает предложение о компенсации от Агента Самолета 2. Если он назначит Самолет 2 на Заказ 2, то его прибыль составит: 20 000 – (1000 + 1200) * 3 = 13 400 д.е. Поэтому компенсация, которую нужно заплатить Агенту Заказа 1, значительно меньше, чем прибыль, которую Агент Заказа 2 получит от сделки. Агент Заказа 2 выбирает Самолет 2 (10).
- Самолет 2 подтверждает Заказ 2 (11).

Данный пример показывает, как самоорганизация группы агентов позволяет легко адаптироваться к новым событиям.

Разработанный метод был впервые применен для решения сложных логистических задач, а далее был развит для понимания текстов и извлечения знаний из баз данных методом кластеризации.

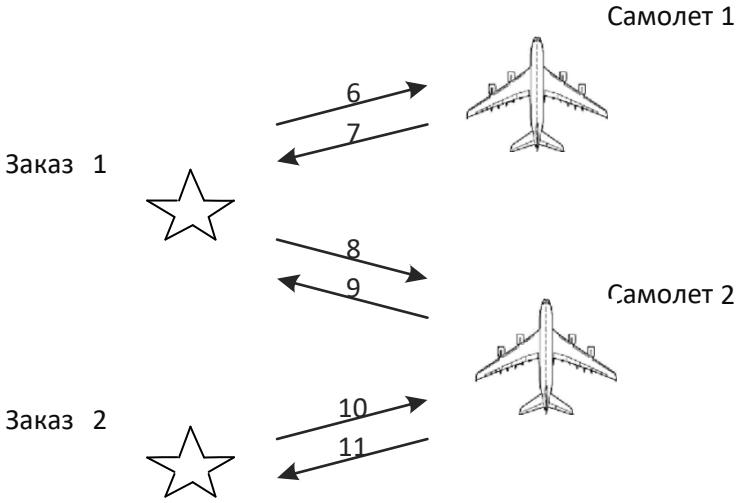



Рис. 21. – Агент Заказа 2 назначает Самолет 2
на выполнение своего заказа

### 6.2. База знаний для адаптивного планирования

В рассмотренных выше примерах знания о предметной области, например, транспортной логистики, играли ключевое значение для принятия решений агентами при управлении ресурсами.

Можно ли создать мультиагентную систему, которая была бы относительно универсальной и легко настраиваемой на особенности различных предметных областей?

Один из возможных подходов связан с созданием баз знаний на основе онтологий, представленных семантическими сетями понятий и отношений предметной области.

Понятие «онтология» стало активно развиваться в последнее время в связи с развитием направления семантического Интернета (Semantic Web), целью которого является представление информа-

ции о содержании Интернет страниц в виде, пригодном для компьютерной обработки, в настоящее время страницы «не знают», что именно в них содержится, это делает смысловой поиск ограниченным и, тем более не дает возможность программам рассуждать о содержимом или поддерживать диалог с пользователем.

В этом подходе каждая веб-страница и любой другой объект (видео и звук, текст и т.д.) могут иметь смысловую аннотацию содержания, т.е. страницы узнают, что в них находится.

В этих целях уже используется стандарт RDF, описывающий семантические сети (графы), в которых узлы и дуги имеют свои адреса и идентификаторы понятий и отношений. Утверждения, кодируемые с помощью RDF, в дальнейшем можно интерпретировать с помощью онтологий, созданных по стандартам RDF Schema и OWL, чтобы получать из них логические заключения. Как правило, это связки типа «объект 1-отношение-объект 2», посредством которых может быть описана любая семантическая сеть.

*База знаний* содержит знания о предметной области, классы понятий и отношений, а также факты, касающиеся экземпляров этих понятий и отношений, необходимые агентам для выработки, принятия и согласования решений [16].

*Онтология* – часть базы знаний, которая содержит понятия, представленные сетью из:

1. Классов объектов;
2. Отношений между классами объектов;
3. Атрибутов классов объектов;
4. Сценариев поведения класса объектов.

Классы объектов – это узлы, а отношения – связи между ними. Вместе с атрибутами и сценариями классы объектов и отношения содержат все знания, требуемые для работы мультиагентной системы.

Цели создания онтологий весьма разнообразны:

- нормативная – унификация понятий и отношений предметной области;
- формирование электронного «толкового словаря» предметной области;
- автоматические рассуждения на основе знаний предметной области;
- автоматический контроль правильности входных данных;
- поддержка деятельности по накоплению, разделению и повторному использованию знаний предметной области (предприятия) в процессах принятия решений;
- построение самообучающихся систем за счет того, что знания отделены от программного кода;
- интегрирование междисциплинарных знаний различных пользователей.

В рассматриваемом случае предлагается использовать онтологии с целью настройки МАС для управления ресурсами на специфику предметной области и далее каждого конкретного предприятия.

Например, в онтологии управления производством (рис. 22) могут быть подробно описаны типы исполняемых заказов, производимые продукты, типы продуктов, технологические процессы и нормы времени, необходимые для их производства, необходимые станки, материалы и инструменты, затраты на производство и хранение на складах и т.д.

Сценарии расчета скидок могут включать следующие стратегии: «*постоянный клиент*» (в зависимости от количества предыдущих покупок), «*оптовая торговля*» (в зависимости от объема и частоты покупок), «*покупка с задержанной доставкой*» (потребитель оплачивает товар заранее дешевле и в течение определенного времени ожидает доставку), «*цены конкурентов*» (установление цен, примерно равных ценам конкурентов). Существуют также другие правила принятия решений, например, правила оценки доходности

распределения ресурсов, критерии оценки, определенные потребителями, и т.д.

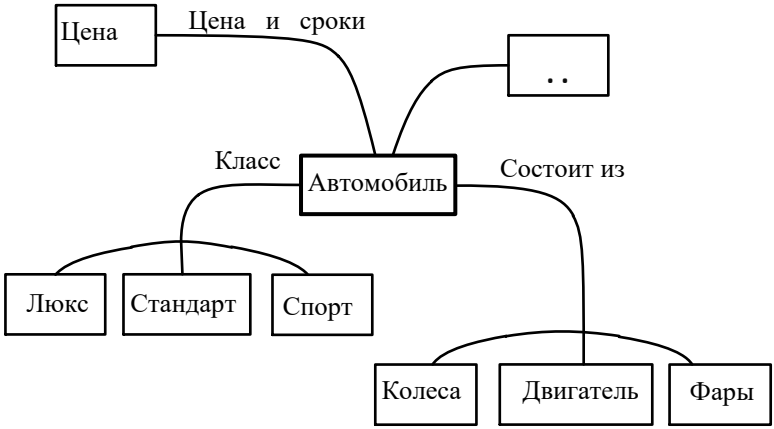



Рис. 22. Фрагмент онтологии производства автомобиля

Текущее состояние виртуального мира, зеркально отражающего состояние внешней среды в определенный момент времени, называют *сценой* (*ситуацией*). Сцены представляют модель текущей ситуации в ПВ-сети мира и могут храниться в традиционных базах данных.

Чем больше продуманных отношений «пронизывают» сцену, тем больше имеется возможностей для анализа ситуации и сокращения комбинаторного перебора в ходе адаптивного планирования и оптимизации ресурсов.

Например, если агент нового заказа знает город, «откуда» требуется везти груз, то по этому городу агент может быстро найти все грузовики, запланированные к выезду из него и проверить, какие грузовики выезжают отсюда в нужном ему направлении и попытаться встроиться в них прежде, чем пытаться строить новый маршрут и бронировать новый грузовик под свои нужды.

Тем самым сцена позволяет также резко сокращать полный перебор при поиске вариантов решений в транспортной логистике и других применениях.

### 6.3. Виртуальный мир

Виртуальный мир – это программная среда, в которой работают агенты и решается задача динамического распределения, планирования и оптимизации ресурсов.

Рассмотрим для примера самый простой виртуальный мир ПВ-сети, поддерживающий распределение ресурсов по потребностям вне времени и пространства.

Агенты в виртуальном мире могут быть свободными (то есть могут не иметь никаких отношений с другими агентами), тогда они стремятся установить с ними отношения, пытаясь соответствовать своим потенциальным партнерам. Свободные агенты в виртуальном мире постоянно ищут подходящие соответствия, рассматривают выгоду доступных связей и выдвигают предложения потенциальным партнерам. Связи между агентами устанавливаются только при согласии обеих сторон и при условии, что предложенное отношение является лучшим вариантом из всех доступных в настоящее время. Агент удовлетворен установленным отношением (что можно увидеть на пиктограмме, отражающей его состояние), если оно является более выгодным, чем в среднем по рынку или полностью соответствует его запросам. В противном случае отношение не устанавливается. Неудовлетворенность агента является причиной для пересмотра установленного отношения при первой новой возможности, о чем агент узнает из оповещений при подписке на определенные события, либо путем анализа сцены.

Если, исследуя рынок, активный агент не находит подходящего соответствия, он переходит в пассивное состояние и ожидает сооб-

щений от других агентов или события, способного изменить ситуацию, например, появление новой потребности в ресурсах, и, соответственно, создание нового агента потребности. В любом из перечисленных случаев агент становится активным. Он возвращается назад в пассивное состояние, когда цепочка переговоров, сгенерированная в результате того или иного события, заканчивается. Агенты, которые установили отношения с другими агентами (достигли соответствия), могут также переходить в пассивное состояние: они включают свой таймер и ожидают лишь периодической инициализации.

После активации агентов сообщением, процесс реализуется в следующей последовательности. Сначала активируются свободные агенты, затем неудовлетворенные агенты, и последними активируются удовлетворенные агенты, чтобы в случае необходимости пересмотреть их связи на основе метода компенсаций

Рассмотрим взаимодействие агентов, используя пример, изображенный на рис. 23. Существует семь агентов, которые стремятся установить друг с другом отношения. Агенты потребностей окрашены в белый цвет, агенты возможностей – в серый. Два агента слева уже установили отношение (сплошная стрелка между ними) и удовлетворены своей связью – их пиктограммы содержат улыбки. У трех агентов в центре также улыбающиеся лица: они находятся в процессе принятия решения и установления отношения (пунктирная стрелка). Группа справа находится в процессе принятия решения, но совершенно не довольна условиями (у них грустные лица), они, вероятно, не видят хороших возможностей соединения. Отметим, что активными могут быть как Агенты потребностей, так и Агенты возможностей (как показывают стрелки).

Как правило, агенты могут прекращать отношения только по взаимному согласию, но с некоторыми исключениями, которые будут обсуждены позже.

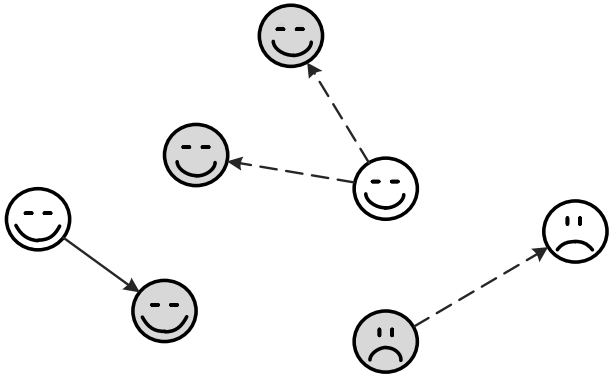


Рис. 23. Сцена: сопоставление Агентов потребностей (белого цвета) и Агентов возможностей (серого цвета), отмечена их степень удовлетворенности

Для решения проблем логистики необходимо рассматривать дополнительные факторы, начиная с пространства и времени.

Подразумевается, что необходимо построить карту грузовых терминалов и рекомендованных маршрутов (дорожных магистралей, воздушных и морских маршрутов, и т.д.) с соблюдением масштабов (упрощенный вариант такой карты изображен на рис. 24), и разместить на ней возможные локации заказчиков и пункты назначений, склады, конвейеры, транспортные ресурсы и т.д. Также необходимо учитывать, в какое время должно быть выполнено распределение ресурсов по потребностям.

На программном уровне каждая сцена в системе определяется группами агентов или связей, выстроенных обычно в виде двусвязных списков[2].

[2] Двусвязный список (двунаправленный связный список) – список, в котором каждый узел имеет ссылки, указывающие на предыдущий и на последующий узел в списке.

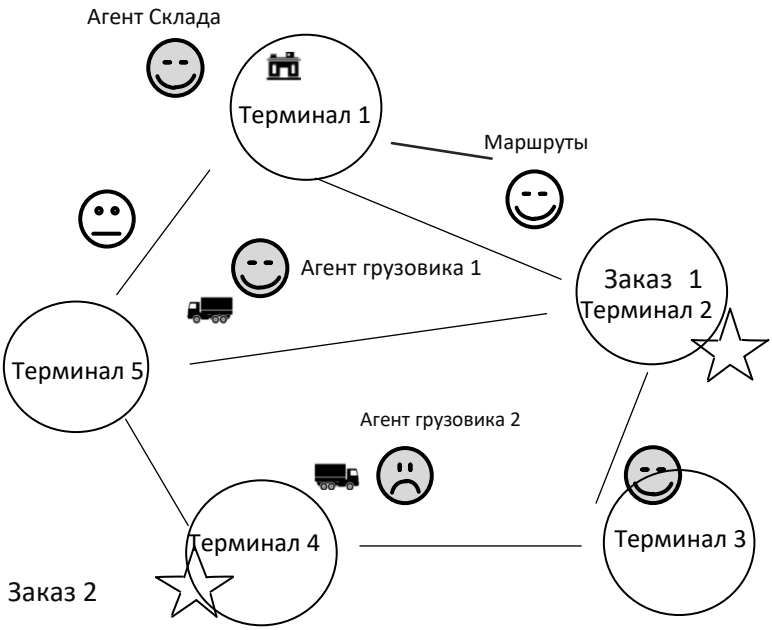



Рис. 24. Сцена виртуального мира логистики

Такое построение позволяет планировщику переходить от любого агента или связи к его «соседу» по списку. Благодаря тому, что элементы этих списков ссылаются друг на друга, агенты способны обнаруживать необходимые связи или других агентов, связи с которыми уже были установлены.

На практике эти агенты и связи разделены на группы активных и неактивных, а также удовлетворенных и неудовлетворенных агентов. Такое разделение позволяет планировщику быстрее делать предварительную выборку, и тем самым снижает время на обработку задач системой.

### 6.4. Машина принятия решений

Модуль принятия решений каждого агента включает в себя набор процедур и структур данных, которые типизируют и упрощают выполнение сценариев принятия решений. Эти процедуры могут быть написаны на специальном промежуточном языке, позволяющем обрабатывать структуру исходных данных и реализовывать такие действия, как организация обмена информацией среди агентов, сбор информации о динамике спроса и предложения, поиск возможных связей между агентами, выбор лучших из них, принятие решений и их пересмотр в случае изменения ситуации.

Каждый агент определяется рядом свойств и их значений следующим образом: <Свойство 1 = Значение 1>, <Свойство 2 = Значение 2>, … Чтобы узнать о свойствах агента и их значениях, другие агенты могут задавать ему соответствующие вопросы («У тебя есть такое свойство? Каково его значение?»).

Чтобы упростить логику сценария и сократить количество запросов, некоторые свойства могут скрываться от агентов. Принятие решений выполняется на основе полученной из сообщений информации, которая была определена заранее и содержится в модели мира и правилах поведения агента в различных ситуациях. Модель мира устанавливает, какие агенты и какого класса могут быть обнаружены в этом мире, какие отношения они могут иметь и какие действия могут быть с ними произведены.

Типичной структурой данных для принятия решений является таблица вариантов предложений с соответствующей информацией о параметрах каждого предложения. Пример такой таблицы для Агента потребностей в логистике приведен ниже. Для упрощения, табл. 7 отражает только три свойства: класс продукта, затраты и время доставки.

Таблица 7. **Пример таблицы принятия решений агентами**

| № | Предложения | Класс продукта | Затраты | Время доставки | Валовая прибыль агента | Валовая прибыль системы |
|---|---|---|---|---|---|---|
| | | | | | | |

Структура таблицы будет меняться в зависимости от того, сколько критериев и свойств определено для агентов потребностей и возможностей. Далее представлены основные операции, которые виртуальная машина принятия решений может выполнять при работе с таблицей: создание/удаление таблицы, заполнение или очистка полей, сортировка предложений по полю и поиск строки с использованием шаблонов.

Самый важный шаг в принятии решений – выбор наилучшего предложения. Эта процедура может быть весьма сложной. В самом простом случае существуют потребность и возможность, полностью подходящие друг другу (например, по цене и времени доставки). Более сложный случай – когда потребность и ресурс лишь близки друг другу по свойствам, то есть присутствует частичное соответствие.

Чтобы сопоставить свойства потребности и ресурса, Агент потребности может отправить запрос, при необходимости указав в нем требуемые важность и значения свойств, например, «Имеются ли ресурсы класса A со свойством A1, имеющим значение Z1 и важность B1, а также свойством A2, имеющим значение Z2 и важность B2?» и т.д. Агенты ресурсов проверяют свойства ресурсов и их значения в онтологии и дают ответ. Даже если ресурсы не полностью соответствуют указанным в запросе свойствам, они могут быть предложены как возможные варианты, т.к. их характеристики частично подходят.

Существует два способа сопоставления ресурсов и потребностей с учетом приоритетов их свойств:

1. *Последовательное сопоставление, начинающееся со свойств первостепенной важности*. Первый Агент потребностей отправляет сообщение Агентам ресурсов: «Я – Агент потребностей класса N. Если среди существующих присутствуют какие-либо Агенты возможностей класса N со свойством S1, укажите его значение». После получения их ответа, Агент потребностей создает таблицу, в которую вносит полученные свойства и их значения. Используя данные из таблицы, он выбирает одно или несколько подходящих предложений и отправляет соответствующим агентам вопрос: «У Вас есть свойство S2? Если да, то, какое значение оно имеет?» Данная стратегия позволяет агенту двигаться от более важных свойств к менее важным; применять стратегию частичного сопоставления (например, не уделяя внимания свойствам незначительной важности); сокращать количество операций при сравнении; оставлять значения скрытых свойств невидимыми, изменяя их в соответствии со стратегией проведения переговоров или ситуацией на рынке (например, согласно числу конкурентов).

2. *Параллельное сопоставление через посредника*. Агенты потребностей показывают свои открытые и скрытые свойства посреднику, который вычисляет коэффициенты схожести и отправляет их агентам. При этом способе агенты не знают значений свойств друг друга, но знают о степени их схожести или различия благодаря рассчитанным коэффициентам.

Еще одни важным типом структуры данных для принятия решений является таблица индикаторов рынка, которая показывает средние значения спроса и предложения (табл. 8).

Таблица 8. **Таблица индикаторов рынка**

| № | Продукт | Текущая средняя цена | Увеличение или уменьшение |
|---|---|---|---|
| | | | |

Наличие однотипных машин принятия решений у агентов позволяет унифицировать конструкцию агентов и упростить отладку процессов принятия решений.

## 6.5. Переговоры агентов

Переговоры агентов являются важнейшим способом формирования и согласования принимаемых решений.

Проводя переговоры, агенты координируют свои действия и договариваются о стоимости сделок (если данные действия разрешены). Например, если цена ресурса равна себестоимости + затраты на хранение + 10% прибыли, то агент, ответственный за этот ресурс, может уменьшить свою прибыль до, скажем, 5%, чтобы сделать цену продукта ниже, чем у конкурентов. В рассматриваемом случае метод компенсаций, подробно представленный выше, дает представление о структуре протоколов, используемых для переговоров агентов и принятия согласованных решений.

Важно отметить, что при приходе нового заказа проводимые переговоры могут приводить к тому, что старые заказы, ранее размещенные в системе, будут уступать свои лучшие позиции новому заказу, но это будет только в том случае, если новый заказ компенсирует ухудшение положения ранее размещенным заказам, т.е. для системы в целом это будет глобально выгодным, поскольку сумма всех улучшений будет больше суммы всех ухудшений.

В этих целях, когда приходит новый заказ, и его агент пытается «растолкать» на лучшем для себя ресурсе другие заказы и они, в свою очередь, «расталкивают» другие заказы на других ресурсах, базовая сцена не меняется, а меняется сцена, в которой динамически определяется разница по глобальному и локальному показателю агентов (без полного пересчета).

Таким образом, в ходе переговоров по методу компенсаций глобальный показатель системы пересчитывается адаптивно и лишь частично, без полного пересчета, что и позволяет оценить влияние события на глобальные показатели системы.

Примеры протоколов переговоров агентов для планирования задач по рабочим производственного цеха при начальном простом размещении задач в свободные слоты и последующей стадии проактивного улучшения показателей планов представлен на рис. 25 и рис. 26.

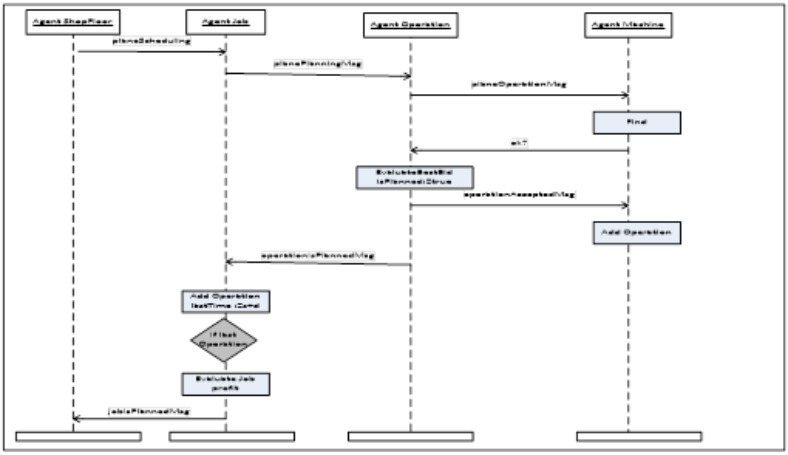



Рис. 25. Переговоры агентов задач и рабочих при начальном простом размещении операций в свободные слоты времени

На рис. 25 на стадии предварительного планирования агенты операций размещаются на ресурсах бесконфликтным способом, чтобы получить грубое приемлемое расписание для дальнейшего улучшения. Вертикальные линии диаграммы означают линию поведения соответствующего агента во времени.

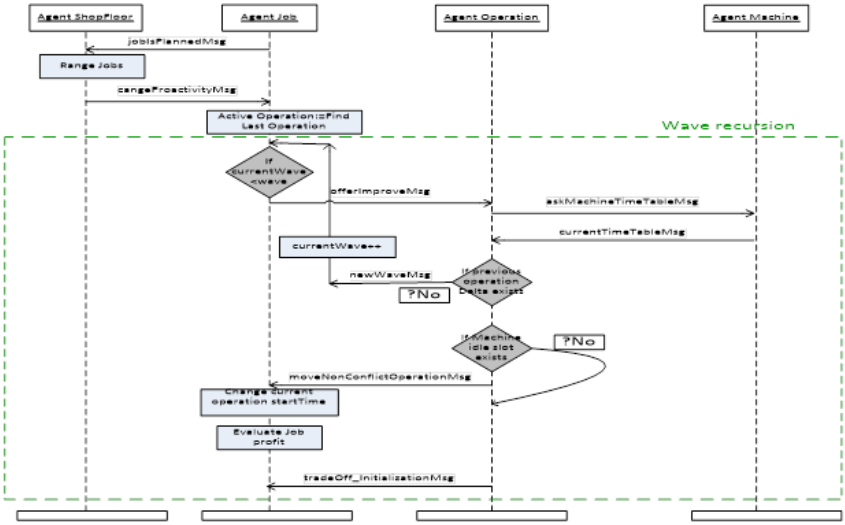



Рис. 26. Переговоры агентов задачи размещения рабочих при проактивном улучшении планов

На рис. 26 показаны примеры сообщений проактивной фазы, где выделен блок рекурсивного развертывания анализа затронутых изменений расписания.

Существуют различные способы организации ведения переговоров в логической архитектуре системы. Наиболее распространенный метод – *P2P-переговоры* (от *англ. peer-to-peer*, то есть «каждый с каждым» и «равный с равным»).

Другой способ состоит в том, чтобы организовать *виртуальный «круглый стол»* для агентов, и вести переговоры по принципу «многие со многими», который эффективен, когда необходимо договориться большому числу различных по типу агентов.

В этом случае агенты могут обозревать сцену виртуального круглого стола и поочередно вносить предложения, которые другие агенты проверяют на согласованность со своей системой значений.

Если некоторые ограничения не соблюдаются (например, превышен бюджет проекта), агенты возвращаются в переговорах на один или несколько шагов назад, чтобы выбрать направление сокращений и найти альтернативные варианты решений.

Наиболее популярным является *протокол Contract-net*, а также различные варианты *прямых и обратных аукционов*.

Ряд протоколов переговоров агентов стандартизирован международной ассоциацией по физическим агентам FIPA.

Подробное описание разработанных моделей, методов и алгоритмов можно найти в работах [17–19].

## 6.6. Архитектура МАС по управлению ресурсами в реальном времени

Общая архитектура предлагаемых мультиагентных систем представлена на рис. 27.

Основные модули МАС для управления ресурсами описаны ниже.

**Модуль распознавания образов ситуаций**

Модуль распознавания образов ситуаций позволяет выявлять скрытые знания в данных, которые могут использоваться для прогнозирования потребностей или возможностей. Например, если крупный и выгодный заказ поступает каждую пятницу, разумно предварительно заказать определенные ресурсы для реализации этого заказа. И если вдруг, вопреки ожиданиям, заказ не поступает, система уведомит клиента и освободит ресурсы для других заказов.

**Адаптивный планировщик**

Это основной модуль, который обрабатывает поступающий поток событий (новые заказы, отзыв уже запланированных, новый ресурс, выход из строя ресурса, задержки и другие) и в начале создает, а далее постоянно корректирует расписания в режиме реального

времени, предоставляя пользователю возможность дорабатывать их в интерактивном режиме.

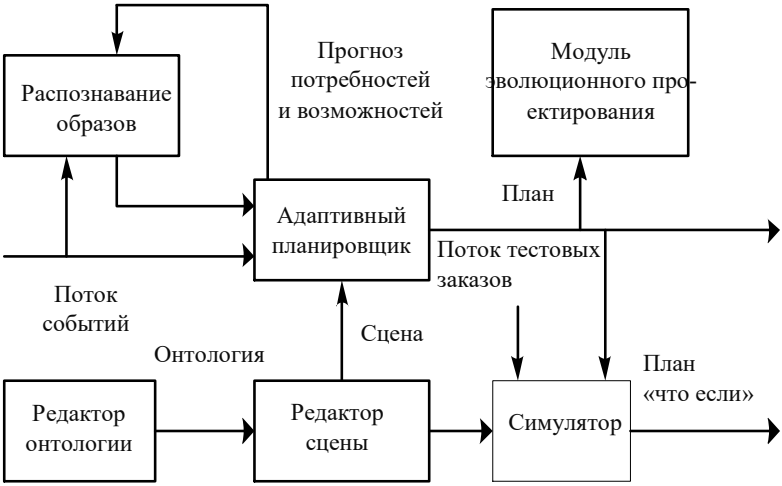



Рис. 27. Архитектура МАС для управления ресурсами

**Редактор онтологии**

Помогает создавать или редактировать онтологию производственной или транспортной сети, которая требуется для последующего построения сцен этой сети. Онтология содержит классы объектов (грузовик, завод, склад) и их отношения (например, «ресурс закреплен за потребностью»), атрибуты (стоимость заказа) и в идеале правила принятия решений, модели поведения, бизнес-процессы и протоколы взаимодействий.

**Редактор сцены**

Редактор сцены позволяет создавать конкретную ситуацию в модели производственной сети предприятия и вручную или автоматически описывать начальное состояние сети, извлекая данные из других источников (из баз данных, xls или xml файлов и т.д.). Для этого пользователь должен выбрать и загрузить соответствующую

онтологию, которая используется в качестве словаря понятий и отношений при описании конкретной пользовательской сети. Если понятий и отношений, требуемых для определения новой ситуации в сети, не достаточно, онтология предметной области должна быть расширена с помощью редактора онтологии, описанного выше.

**Симулятор**

Моделирующая система (симулятор) является удобным инструментом, который помогает понять, к какому результату приведет то или иное изменение в системе в будущем, без разрушения текущего расписания. В любой момент текущее состояние, скажем, транспортной сети или сети цепочек поставок, может быть загружено в симулятор для поиска ответов на различные вопросы, например, «Что произойдет, если неожиданно поступят новые очень крупные заказы»? Еще одна его функция заключается в том, чтобы анализировать возможности изменения конфигурации сети (изменяя расположение основных ресурсов или делая доступным новый тип оборудования) и «разыгрывать» различные сценарии оптимизации сети параллельно с непрерывным планированием поступающих заказов.

**Модуль эволюционного дизайна**

Модуль эволюционного дизайна автономно создает и развивает сети, создавая предположения о том, как адаптировать сеть к постоянно меняющимся спросу и предложению. Например, если при производственном планировании работ в цехе нарушения сроков все время связываются с отсутствием токарей, то данный блок выработает предложение менеджерам увеличить число токарей и даже оценит упущенную прибыль за прошедший период, или, например, рекомендует мастеру вызвать из отпуска рабочего нужной компетенции.

Архитектура центрального компонента MAC – адаптивного планировщика – показана на рис. 28.

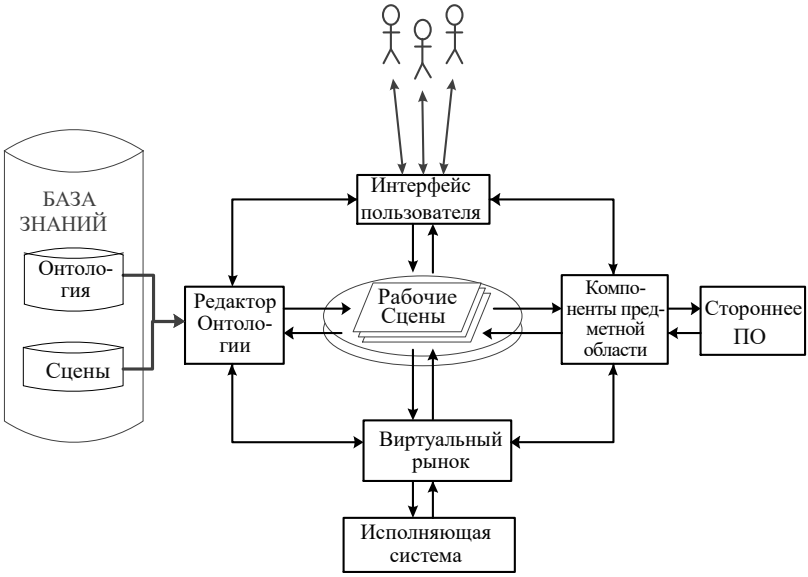



Рис. 28. Архитектура адаптивного планировщика

Кратко определим основные компоненты адаптивного планировщика.

**Онтология**

Онтологии представляет собой семантическую сеть знаний о том, как функционирует виртуальный мир предметной области, который в общем случае может содержать как декларативные (описательные), так и процедурные компоненты [16].

В виртуальном мире, описанном в онтологии, населенном физическими и абстрактными сущностями (объектами), можно создавать сценарии для выполнения действий над объектами и их реакциями в соответствии с правилами, взятыми из соответствующей предметной области знаний реального мира. Например, для агента заказа онтология логистики описывает, какие элементы необходимы, чтобы выполнить заказ; кто производит или поставляет эти элементы, и кто и где может заказать и получить их. Существует

также необходимость описания абстрактных миров, например, мира логистики (который содержит такие объекты, как заказ, расписание, срок и т.д.) или финансов (прибыль, доход, сделки и т.д.).

В процессе создания сценария существует возможность определить параметры объектов и агентов и установить их начальные состояния (например, для склада или транспортной системы) и ввести начальные заказы.

Затем виртуальный мир может быть запущен в работу.

**Виртуальный мир**

Виртуальный мир – это пространство, в котором работают агенты, т.е. исполняются автономные программы агентов.

Виртуальный мир является зеркальной моделью работы предприятия предметной области реального мира, в котором агенты выполняют свои задачи, сотрудничая или конкурируя с другими агентами.

**Виртуальный рынок**

Виртуальный рынок содержит библиотеки программных компонент, поддерживающих работу агентов на виртуальном рынке, включая различные модели определения удовлетворенности агентов, микроэкономики, схемы переговоров и т.д.

Возможности виртуального рынка позволяют управлять активностью агентов при решении сложной задачи, что напрямую влияет на качество и эффективность получения результатов.

**Исполняющая система**

Исполняющая система поддерживает асинхронную параллельную работу агентов и включает подсистему поддержки работы агентов, подсистему коммуникационной поддержки, подсистему управления переговорами и некоторые другие вспомогательные компоненты.

**Интерфейс пользователя**

Загрузив виртуальный мир, пользователь может задавать различные сценарии из потоков событий. Задание входных данных и

получение выходных данных, а также визуализация процессов формирования, согласования и принятия решений, осуществляется через интерфейсы пользователя. Например, в системе логистики, в виртуальном мире компании, пользователь может поместить на экран города и дороги, сборочные конвейеры и транспортные средства.

Основные окна пользовательского интерфейса адаптивного планировщика обычно позволяют импортировать заказы и размещать их в очереди событий, запускать планирование и просматривать получаемые расписания с помощью диаграммы Ганта и на карте, а также видеть ключевые показатели эффективности (KPI) системы.

Можно настроить и расположить различные части окна так, как будет удобнее конкретному пользователю. В любой момент пользователь может активировать диспетчер агентов и посмотреть, какие агенты в настоящее время работают в системе, просмотреть журнал переговоров и финансовых транзакций между агентами.

Чтобы проверить, как агенты принимают решения о распределении ресурсов, пользователи могут проанализировать журнал принятия решений агентов.

## 6.7. Мультиагентная платформа

Управление ресурсами в различных предметных областях, таких как управление проектами, транспортная или производственная логистика, железнодорожные перевозки или управление роем спутников, всегда связано с особенностями в постановках задач, где принимаются во внимание разнообразные факторы. При этом иногда даже в одной индустрии разные предприятия используют совершенно различные модели принятия решений, критические для бизнеса.

Чтобы не начинать программирование каждого нового МАС приложения «с нуля», рекомендуется определить компоненты мультиагентных систем, которые присутствуют в большинстве приложений, и воспроизвести их в общей библиотеке, которая называется *платформой для адаптивного планирования*.

Для разработки такой платформы, обеспечивающей поддержку промышленных приложений, необходимо иметь значительный опыт в разработке, реализации, внедрении и поддержке мультиагентных приложений.

Ниже будут описаны несколько рекомендуемых типовых компонент, которые могут являться составной частью платформы.

**Подсистема выполнения параллельных процессов**

Эта подсистема призвана обеспечить управление бо́льшим числом нитей агентов по сравнению со стандартным механизмом операционной системы.

Переключение с управления одним процессом на другой организовано обычно за счет включения в программу специальных точек прерывания по требованию системы, которые отнимают много времени, поэтому более эффективный перехват управления сразу приводит к большому выигрышу в производительности системы.

Большие новые возможности открываются перед мультиагентными системами с развитием суперкомпьютеров и созданием сетецентрических систем («систем систем»).

В последнее время появляется все больше сервисных платформ, поддерживающих параллельные «легкие» процессы, которые также могут эффективно использоваться при построении мультиагентных систем.

**Подсистема поддержки принятия решений**

Эта подсистема позволяет программисту расширять возможности исполняющей системы, добавляя к системе новые модули виртуального рынка (расширения) как для разработки функциональ-

ных компонентов системы, включая новые типы агентов или модели микроэкономики, так и для разработки интерфейса.

**Подсистема поддержки онтологии и сцен**

Основная задача подсистемы состоит в том, чтобы обеспечить доступ к онтологии и сценам, например, загрузки и сохранения соответствующих структур данных, поиск требуемых элементов, управления курсором в семантической сети и т.д.

Подсистема должна позволять прерывать любые процессы в любое время, а затем восстанавливать их, продолжая работу с любой сцены (состояния).

**Подсистема задания и выполнения сценариев**

Поддержка языков описания сценариев агентов – важная часть онтологии, которая позволяет пользователям создавать собственные правила и сценарии работы агентов, учитывая особенности работы своего предприятия.

**Подсистема коммуникации**

Одна из основных особенностей агентов – их возможность коммуницировать с другими агентами. В рассматриваемых МАС обычно требуется поддержка как синхронных, так и асинхронных режимов; необходимо накапливать очереди сообщений для каждого агента и процесса; обнаруживать сообщения, отправленные агентам, выведенным из строя, вести журнал сообщений для отладки взаимодействий между агентами и т.д.

**Подсистема визуализации**

Эта подсистема позволяет визуализировать исходные данные, процессы и результаты работы МАС.

## 6.8. Оценка МАС как сложных систем

Покажем, что разработанные мультиагентные системы удовлетворяют введенным выше семи критериям сложности.

**1. Связность**

Высокая связность агентов – ключевое свойство МАС для управления ресурсами. Обычно большое число разнообразных агентов взаимодействуют между собой, обмениваясь сообщениями. Но, как правило, Связность < 1, потому что не каждый агент будет взаимодействовать с каждым. Связи между агентами могут различаться по силе (прочности), и более слабые связи намного проще разрушить, для того, чтобы установить новые.

**2. Автономность**

Поведение агентов управляется целями, к которым агенты постоянно стремятся, используя сценарии (правила), хранящиеся в Базе знаний, или жестко заданные на уровне кода системы.

Сценарии работы агентов разработаны так, чтобы предоставить им определенную свободу самостоятельно выбирать и пересматривать варианты решений, и иногда даже идти на метод «проб и ошибок» в выборе способа, как действовать, если знание о том, как решить проблему, является неполным.

Агенты никогда не останавливаются в своей работе (могут лишь «засыпать» на время) и в любой момент пользователь может обратиться к агенту и запросить его показатели, историю принятия решений, состояние счетов и т.д.

**3. Эмерджентность**

Поведение МАС не регулируется централизованно, напротив, оно возникает из локальных взаимодействий программных агентов непредсказуемо (порядок срабатывания агентов может быть недетерминированным), но отнюдь не случайно.

Новое решение (переход из одного неустойчивого состояния в другое) может осуществляться как по мере прихода нового события (и даже малое событие может приводить к большим изменениям расписания), так и в результате внутренней проактивности системы, запускающей волну внутренних пересмотров решений.

Когда прибывают заказы и выполняется распределение ресурсов, сначала связи выстраиваются более прочными, но со временем, если используются виртуальные налоги, они слабеют и даже разрываются, а также регулярно подвергаются пересмотрам. Со временем, по мере прихода новых выгодных заказов, все большее число слабых связей разрывается и заменяется новыми более сильными связями, что позволяет всегда иметь в системе относительно стабильное расписание.

В экспериментах наблюдаются ситуации, когда решение всякий раз строится по-разному (например, в зависимости от того, какой из агентов первым начал переговоры), но конечный результат получается примерно один и тот же, если прямые и обратные связи при принятии решений уравновешивают друг друга.

**4. Неравновесие**

Возникновение события на входе системы побуждает пострадавших агентов приспособиться к нему, что запускает волну переговоров. Если такие события случаются часто, у агентов не будет времени, чтобы выполнить свои задачи и добиться равновесия как полного баланса интересов. Кроме того, сила связей между агентами может существенно отличаться для каждой пары агентов в разные моменты времени.

Следовательно, в этом случае система будет большую часть времени работать в состоянии, далеком от равновесия.

**5. Нелинейность**

При определенных условиях самое небольшое изменение во внешней среде (например, прибытие нового незначительного заказа) вызывает значительные изменения в расписании («эффект бабочки»).

Эффектом бабочки можно управлять за счёт перераспределения виртуальных денег по заказам на основе прогноза, реализуемого через специальные тестирующие «виртуальные заказы», кото-

рые могут играть роль «датчиков», предсказывающих будущие значительные перемены расписания.

При определенных условиях переговоры агентов могут перерасти в колебательное поведение, состояние, указывающее, что система находится на грани между двумя аттракторами и не может решить, какой больше подходит. Такие колебания могут усиливаться, если они распространяются от узла к узлу в агентной сети и могут привести к ее распаду, если не будут приняты меры, гасящие распространение опасных колебаний.

Примером таких действий может служить временное уменьшение связности агентов или введение локальных демпферов решений.

**6. Самоорганизация**

Когда в мультиагентной системе возникает событие, агент, назначенный системой на обработку события (например, агент сломанного грузовика), вместе с пострадавшими агентами заказов пытается пересмотреть распределение ресурсов; в результате, ранее установленные связи потребность-возможность могут быть разорваны, а новые сформированы самой системой без внешних инструкций.

Пострадавшие заказы просто переместятся на другие ресурсы, возможно, с ухудшением своего положения и положения некоторых других, которое затем может быть улучшено с приходом новых заказов или освобождением занятых грузовиков.

В результате, система постоянно самоорганизуется в направлении повышения ценности своих решений, даже с учетом негативных событий, нарушающих планы.

**7. Коэволюция (совместное развитие)**

Когда МАС связана с бизнес-процессом предприятия, любое изменение в этом процессе должно отражаться в МАС, и наоборот, в течение определенного периода времени изменения, накопленные в МАС, могут отражаться в бизнес-процессах предприятия.

Еще более наглядно это свойство проявляется при взаимодействии адаптивных планировщиков в рамках «системы систем», когда изменения в планах одного подразделения немедленно вызывают изменения в планах второго, и наоборот.

## 6.9. Выводы по Разделам № 5 и № 6

1. Мультиагентные технологии – перспективное новое направление в области интеллектуальных информационных технологий на стыке объектно-ориентированного программирования, параллельных вычислений, искусственного интеллекта и телекоммуникаций.

2. Мультиагентные технологии – это не только новая программная технология, но и подход к решению сложных задач, которые трудно решаются или не решаются вовсе в классической математике.

3. В этих целях мультиагентные технологии позволяют создавать интеллектуальные системы нового поколения, в отличие от традиционного «механистического» понимания ИИ, базирующиеся на фундаментальных принципах самоорганизации и эволюции, присущих живой природе, например, колонии муравьев или рою пчел.

4. В качестве концептуального базиса для создания моделей сложных адаптивных систем и методов адаптивного управления предлагается концепция сетей потребностей и возможностей (ПВ-сетей), развивающая холонический подход к созданию мультиагентных систем управления ресурсами в реальном времени.

5. Основным методом адаптивной перестройки ПВ-сети в разработанных системах является метод компенсаций, позволяющий пересматривать уже установленные ранее связи в ПВ-сети по событиям в реальном времени в консенсусе агентов, затронутых изменениями.

6. Для реализации разработанных моделей, методов и алгоритмов предлагается архитектура построения прикладных мультиагентных систем и инструментальная платформа поддержки основных сервисов, базы знаний и протоколов переговоров агентов.

7. Показывается, что мультиагентные системы удовлетворяют введенным ранее критериям сложных адаптивных систем и потому могут являться практическим средством для управления сложными системами в реальном времени.

# 7. ТИПЫ АРХИТЕКТУРЫ МУЛЬТИАГЕНТНЫХ СИСТЕМ. МЕТОДЫ ПРОЕКТИРОВАНИЯ МУЛЬТИАГЕНТНЫХ СИСТЕМ

## 7.1. Краткий обзор состояния мультиагентных систем

В своем развитии МАС прошли две фазы:

- 1977–1990-е годы – smart agents («смышленые» агенты). Работы этого времени были преимущественно теоретическими. В этих работах изучались вопросы о том, как агенты могут договариваться (negotiations), как координировать свои действия, как декомпозировать и распределять задачи и т.д.
- 1990-е годы – настоящее время – intelligent agents («интеллектуальные» агенты). Работы этого периода характеризуются практическими применениями. В центре внимания исследователей теперь находятся вопросы архитектуры, языки представления сценариев, средства проектирования агентов. Появились первые инвестиции в этой сфере, и сразу возник первый ряд экспериментальных систем.

Организация FIPA (Foundation for Intelligent Physical Agents, http://www.fipa.org), создана с целью исследования и разработки общих стандартов взаимодействия различных интеллектуальных агентов и программных приложений, основанных на агентских технологиях. Специализируется, в том числе, на создании спецификаций, делающих возможным взаимодействие между такого рода приложениями, созданными различными фирмами-разработчиками, выпустила толковый словарь по агентам.

Организация Agent Link (European Coordination Action for Agent-based Computing, http://www.agentlink.org) – Европейская

сеть производителей программных приложений на базе агентских технологий. Создана и функционирует под эгидой Европейской комиссии с целью координации исследований и разработок в области интеллектуальных агентов, осуществляя одновременно взаимосвязь между разработчиками и конечными пользователями. Agent Link объединяет ведущие европейские фирмы-разработчики интеллектуальных агентов, а также других мировых производителей как ассоциированных членов.

На сегодняшний день два языка описания взаимодействия агентов фактически выполняют функции стандартов:

1. ACL (Agent Communication Language);
2. KQML (Knowledge Query Manipulation Language).

## 7.2. Архитектура мультиагентных систем

В архитектуре мультиагентных систем можно выделить два основных варианта организации взаимодействия уровней: горизонтальное и вертикальное взаимодействие (рис. 29).

### *7.2.1. Композиционная архитектура мультиагентных систем*

Композиционная архитектура МАС известна под названием DESIRE (framework for Design and Specification of Interacting REasoning components). Она обеспечивает «прозрачное описание» сложного агента, а также позволяет интегрировать рассуждения и действия в единой (декларативной) логической среде. Предполагается, что агент в процессе работы выполняет действия следующего типа:

- активно воспринимает и фильтрует информацию из внешнего мира;
- строит заключение по этой информации;

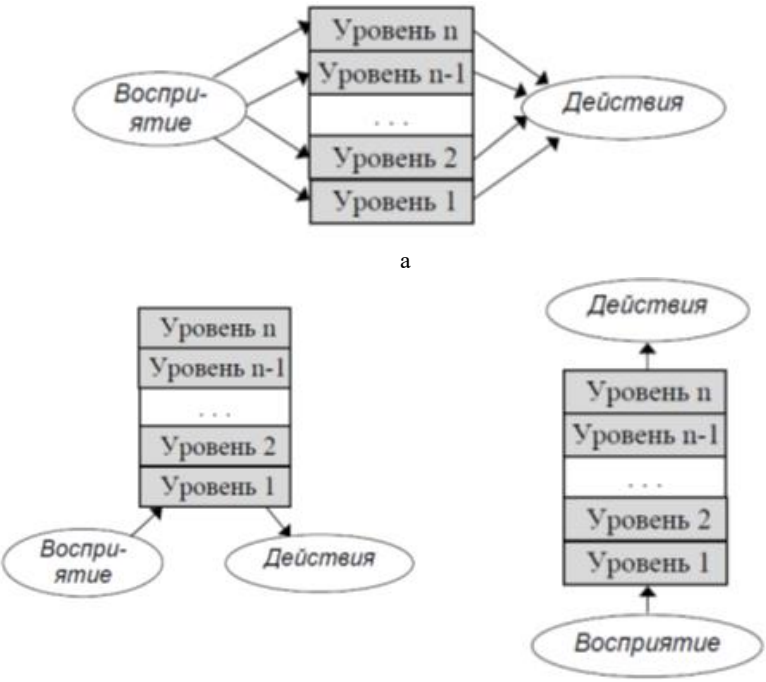



Рис. 29. Варианты организации уровней в МАС:
а – горизонтально организованная архитектура;
б – вертикально организованная архитектура

- вступает в коммуникацию с другими агентами с целью установления сотрудничества с ними;
- генерирует и обновляет свои убеждения, принимая или отклоняя дополнительные предположения;
- воздействуя на внешний мир, изменяет его.

Основу опыта агента составляют знания, которые в этой архитектуре классифицируются следующим образом:

- знания о материальном мире;
- знания о ментальном мире самого агента;

- знания о ментальных мирах других агентов;
- знания о взаимодействии с материальным миром;
- знания о коммуникациях с другими агентами (какие варианты общения возможны и полезны для получения дополнительной информации).

В структуре знаний различаются две части:

1. Часть, зависящая от времени («динамическое состояние информации» или база фактов);
2. Инвариантная часть, которая не изменяется во всех состояниях.

Главная идея композиционной архитектуры состоит в том, чтобы строить любого сложного агента как композицию компонент-примитивов, описывающих одну из подзадач, которая должна им выполняться. Компоненты соединяются друг с другом в соответствии с заранее определенной семантикой связей. Каждая из компонент должна иметь простое локальное описание и использовать свой набор знаний. Сложное поведение, которое охватывает и рассуждения, и действия, может обеспечиваться (динамической) компонентой взаимодействия агентов. Аналогичным образом, МАС в целом может составляться из отдельных агентов. Отдельные компоненты описываются в терминах многосортной логики предикатов.

Компоненты соответствуют функциональным свойствам. Можно отметить следующие достоинства этой архитектуры:

- интеграция различных типов рассуждений и действий в единых декларативных рамках;
- использование знаний о стратегиях для явного управления рассуждениями;
- гибкость в построении агентов различных типов;
- явные и управляемые акты наблюдения;
- явные и управляемые акты коммуникации.

### *7.2.2. Многоуровневая архитектура для автономного агента «TOURING MACHINE»*

Эта архитектура разработана для специального приложения автономного агента – подвижного робота. Агент имеет дело с непредвиденными событиями внешнего мира как в пространстве, так и во времени и действует совместно с другими агентами. При этом он должен адекватно реагировать на внешние события и принимать решения. Но внешний мир невозможно моделировать в деталях. По этой причине архитектура агента является гибридной. Она должна позволять ему работать в условиях неопределенности и неполноты информации, реагировать на непредвиденные события, пользуясь относительно простыми правилами.

Данная архитектура включает в себя три уровня, каждый из которых соответствует различным типам способностей агента:

- уровень реакции на события поддерживает способность агента быстро реагировать на события, выделяемые вышележащим уровнем, даже если они ранее не планировались;
- уровень планирования реагирует, исполняет и динамически реконструирует частичные планы, например, для выбора маршрута подвижного робота;
- уровень предсказания моделирует поведение объектов внешней среды и самого агента, что может использоваться для объяснения текущего наблюдаемого поведения и предсказания возможного поведения в будущем.

Данная архитектура интегрирует в себе ряд традиционных механизмов рассуждений на основе знаний с механизмами чисто поведенческого, «реактивного» характера.

### *7.2.3. IDS-архитектура*

Эта архитектура возникла в результате комбинирования двух направлений исследований. Первое из них опирается на исходное понятие «населенной (искусственными существами) динамической системы» (IDS – Inhabited Dynamic System) и включает логику рассуждений об их действиях и изменениях среды. Второе направление связано с построением эффективной реализации интеллектуальной системы.

Архитектура имеет трехуровневую структуру и является гибридной (рис. 30).


Рис.30. IDS-архитектура

Предполагается, что IDS находится в некотором мире (среде) и состоит из двух базовых подсистем: «Мыслящей подсистемы» («Ego») и «Машины» («тела»). Понятие «мыслящая подсистема» интерпретируется как интеллектуальная, основанная на знаниях

часть автономного агента, его «мозг», в то время как «машина» – тело агента, которое реагирует на воспринимаемые объекты и выполняет приказы, поступающие от «мыслящей подсистемы». На основе восприятия машина обрабатывает входную информацию и посылает выходной сигнал в интеллектуальную подсистему. Та, в свою очередь, посылает команды на исполнительные органы, где они отрабатываются без какого-либо дополнительного управления или изменения, вызывая соответствующие изменения во внешнем мире.

Разделение по уровням производится в соответствии с характером тех вычислений, которые на них выполняются. Первый уровень – это уровень процессов, на котором периодически выполняются с заданной частотой некоторые вычисления, а также осуществляется управление процессами восприятия и исполнения команд.

Второй уровень, называемый уровнем ответной реакции, вычисляет ответную реакцию на асинхронные события, которые либо воспринимаются уровнем процессов, либо им генерируются.

Уровень анализа проводит символические рассуждения, такие, как предсказание, планирование и перепланирование, а также включает компоненту обучения агента.

Данная архитектура является типичным представителем многоуровневой архитектуры.

### *7.2.4. WILL-архитектура*

В этой архитектуре активно используются метафоры и понятия, традиционные для описания интеллектуальной деятельности человека, что делает ее привлекательной и понятной. По мнению ее авторов Д. Моффата и Н. Фрижды, это наиболее простая архитектура автономного агента. Следует, однако, принимать во внимание, что Will-архитектура рассчитана на одного агента, который имеет

единственную цель. Его функционирование направляется собственными мотивами, которые названы «интересами» (concerns). Вопрос о методах кооперации и коммуникации агентов в исходной Will-архитектуре не рассматривается.

Для того чтобы агент рационально действовал в некотором мире, его надо снабдить рядом базовых функций, включая восприятие. Каждая из функций агента реализована как отдельный модуль. Предполагается, что агент имеет Сенсорный блок, Планировщик и Исполнительное устройство в качестве базовых модулей, которые должны быть интегрированы между собой.

Чтобы решить проблему согласования потоков информации, в архитектуре применяется схема «бродкастинга», когда все входы и все выходы модулей соединены между собой, так что любое сообщение, генерируемое тем или иным блоком, становится доступным любому другому блоку.

Все сообщения собираются в глобальном буфере, который называется Памятью. Все блоки, кроме Сенсоров, могут считывать информацию из Памяти, и также все они, за исключением Исполнительного устройства, могут записывать новую информацию в Память. Каждый модуль может считывать информацию из Памяти в любое время, когда ему это нужно.

Допускается, что характеристики системы могут меняться и генерироваться «изнутри», самим агентом, будучи обусловлены некими общими потребностями агента («интересами»). «Интересы» определяются как предпочтения агента находиться в каких-то состояниях, а каких-то состояний избегать. Когда агент получает информацию, которая в соответствии с его интересами отвечает предпочтительному состоянию, генерируется внутреннее представление агента о том, что это состояние среды является желательным и в будущем.

### *7.2.5. INTERRAP-архитектура*

Основная идея этой архитектуры заключается в том, чтобы представить агента как множество уровней, которые связаны через управляющую структуру и используют общую базу знаний. INTERRAP-архитектура представлена на рис. 31.

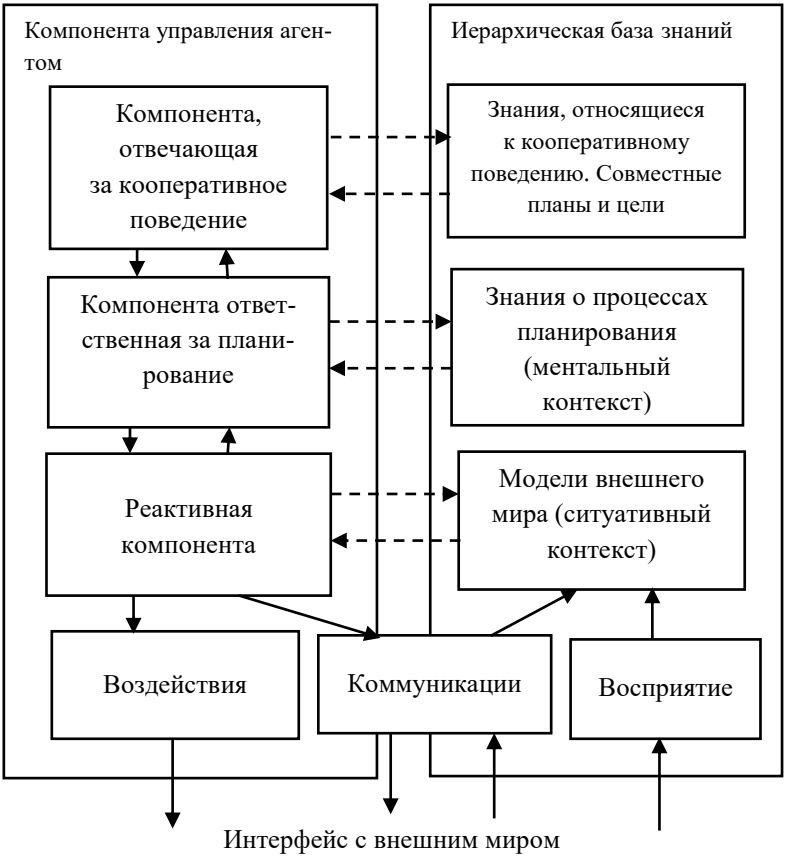



Рис. 31. InteRRaP-архитектура

Она состоит из пяти основных блоков:

1. Интерфейс с внешним миром;
2. Компонента, основанная на поведении;
3. Планирующая компонента;
4. Компонента, ответственная за кооперацию с другими агентами;
5. База знаний агента.

Интерфейс с внешним миром определяет возможности агента по восприятию событий внешнего мира, воздействия на него и средства коммуникации.

Компонента, ответственная за реактивное поведение, использует базовые возможности агента по реактивному поведению, а также частично использует знания агента процедурного характера.

Компонента, ответственная за планирование, содержит алгоритм планирования, позволяющий строить локальные планы агента, т.е. планы, не связанные с кооперативным поведением. Она же участвует и в планировании, связанном с кооперативным поведением агентов.

Компонента, ответственная за кооперацию агентов, участвует в конструировании планов совместного поведения агентов для достижения некоторых общих целей или выполнения своих обязательств перед другими агентами, а также выполнения соглашений.

База знаний агента имеет трехуровневую структуру. Уровни базы знаний фактически отвечают уровням абстракции знаний в соответствии со структурой управляющих компонент. Модель мира агента содержит убеждения агента в соответствии с уровнем, ориентированным на поведение. Второй уровень соответствует модели ментальных характеристик агента и знаниям о его текущем ментальном состоянии (намерения, цели, планы). Наконец, третий уровень содержит знания и убеждения агента о других агентах, информацию о совместных планах, целях я намерениях, т.е. то, что связано с «общественным контекстом».

## 7.3. Методы проектирования мультиагентных систем

### 7.3.1. Восходящий подход

Общая методика восходящего проектирования МАС включает следующие этапы:

1. Формулирование назначения (цели разработки) МАС;
2. Определение типа и основных свойств среды МАС;
3. Определение основных и вспомогательных функций агентов в МАС;
4. Уточнение состава агентов и распределение функций между агентами. Выбор архитектур агентов;
5. Выделение базовых взаимосвязей (отношений) между агентами в МАС;
6. Определение возможных действий (операций) агентов;
7. Построение базовой архитектуры (функционально-структурной единицы) МАС, анализ ее возможных состояний (нормальное, вырожденное, критическое и пр.);
8. Анализ реальных текущих или предполагаемых изменений внешней среды (условий функционирования) или внутренних противоречий МАС;
9. Определение соответствующих изменений функций агентов и формулирование стратегии эволюции МАС;
10. Построение общей архитектуры МАС, инвариантной к рассмотренной области изменений среды (или самовоспроизведение функционально-структурной единицы МАС).

Техническое задание (спецификация) на проектирование МАС должно наряду с ее назначением содержать описание типа среды, в которой будет функционировать МАС. Затем необходимо определить главные функции, возложенные на агентов. При этом организация, для которой выполняется проектирование МАС, может рассматриваться как система ролей, где каждая роль определяет

множество функций агента в МАС. Роли могут быть сосредоточены или, наоборот, широко распределены между агентами.

По ролевому признаку можно вычленить следующие основные виды агентов в МАС:

1. Агент-заказчик, рассылающий заявки на выполнение некоторого задания другим агентам;
2. Агент-координатор (посредник), который принимает заказы и распределяет их между другими (компетентными) агентами;
3. Агент-исполнитель;
4. Агент-субординатор (супервизор), который организует и контролирует работу вышеуказанных трех агентов и наделен правом экстренного вмешательства и перераспределения ресурсов в критических (нештатных) ситуациях.

Эти роли соответствуют минимальному набору функций, необходимых для функционирования организации как целенаправленной системы, т.е. достижения целей в условиях адаптации к изменениям среды. Роли не обязательно являются жестко фиксированными и могут изменяться с развитием организации. Например, исполнитель в результате эволюции организации может стать посредником или клиентом. Точно так же, агент-заказчик в одних случаях может рассматриваться в других случаях как поставщик. Таким образом, каждый агент играет ряд ролей, характеризующих различные услуги, которые он может оказать в данной организации.

Конечно, можно указать и другие варианты специализации агентов. Так, применительно к задачам децентрализованного управления производственными системами выделяются:

- агент-пользователь;
- канальный агент, обеспечивающий обмен информацией в системе;
- сервисный агент, принимающий заявки на транспортировку, выдачу или хранение ресурсов;

- интерфейсный агент, служащий для связи с внешней средой;
- координатор, ответственный за организацию взаимодействия вышеуказанных агентов и разрешение конфликтов.

В случае мигрирующих сетевых агентов и динамических организаций на базе Интернет можно также указать и другие роли сервисных агентов, например, агент-регистратор, агент-архиватор, агент-хранитель узла и т.п.

Понятие роли нередко определяется с помощью трёх атрибутов: ответственность, разрешение и протокол. Здесь ключевым атрибутом является ответственность, которая тесно связана с функциональными характеристиками. Реализация ответственности неотделима от множества разрешений, которые определяют «права», связанные с ролью, а, следовательно, и набор располагаемых ресурсов для реализации ответственности. Наконец, роль описывается с помощью набора протоколов, которые определяют способ взаимодействия с другими ролями в системе.

При проектировании организацию можно рассматривать как набор ролей, находящихся между собой в определенном отношении, и взаимодействующих друг с другом. Исходя из этого, организационная модель дальше распадается на две части: модель ролей и модель взаимодействий (рис. 32).

На стадии анализа происходит предварительная идентификация ролей, затем определяются и документируются соответствующие протоколы. В процессе проектирования создается модель агента, т.е. роли агрегируются в типы агентов, формируется иерархия типов, и документируются примеры каждого типа. Далее разрабатываются модель услуг и модель наиболее тесных контактов.

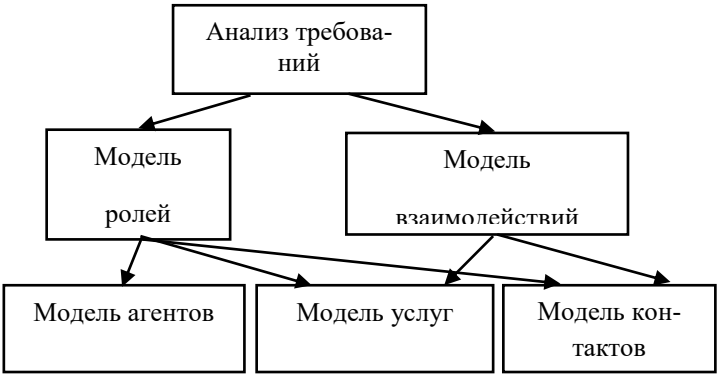



Рис. 32. Связь между моделями
при восходящем проектировании МАС

Итак, методология восходящего проектирования МАС требует предварительного задания исходных функций (ролей агентов), определения круга их обязательств по отношению друг к другу, формирования исходных и развивающихся структур на основе выделенных функций и исследования адекватности этих структур характеру решаемых задач в данных проблемных областях.

### *7.3.2. Нисходящий подход*

Главная идея нисходящего проектирования состоит в определении общих социальных характеристик МАС по некоторому набору критериев, построении базовых типов их организаций с последующим определением требований к архитектуре агентов.

В соответствии с концепцией нисходящего проектирования, индивидуальные свойства и поведение агентов в МАС определяются на основе типа организации и множества соответствующих взаимоотношений между агентами. Здесь проектирование предполагает движение от целого к частям, когда свойства агентов в МАС определяются свойствами сообщества МАС. Удачный выбор ис-

ходного набора организационных критериев (шкал критериев) и использование признака «централизация-децентрализация» позволяют адекватно структурировать МАС.

Общую концепцию нисходящего проектирования МАС можно описать следующим образом:

1. Выбор представительного набора критериев для определения типа организации. В качестве подобных критериев могут выступать: тип внешней среды, взаимодействия организации со средой, структура организации, взаимоотношения внутри организации, характер связей в организации (постоянные или гибкие переменные), стратегии организационного развития, стратегии адаптации к среде и т.д.

2. Конструирование шкал критериев и определение допустимых типов оценок по критериям.

3. Заполнение таблицы, состоящей из оценок по различным выбранным критериям, и определение типа организации.

4. Синтез структуры МАС, соответствующей установленному типу организации.

5. Подбор типов виртуальных агентов, соответствующих построенной организации, и синтез их архитектуры.

# 8. СТАНДАРТИЗАЦИЯ В ОБЛАСТИ ИНТЕЛЛЕКТУАЛЬНЫХ АГЕНТОВ. АГЕНТНЫЕ ПЛАТФОРМЫ. ЯЗЫКИ ПРОГРАММИРОВАНИЯ АГЕНТОВ

## 8.1. Агентные платформы

Средой разработки и существования МАС являются агентные платформы. Было разработано множество программных реализаций агентных платформ, каждая из которых имеет свои особенности, достоинства и недостатки. Вот лишь небольшой список из более чем ста доступных платформ, публикуемых на сайте организации AgentLink (European Coordination Action for Agent-based Computing [20]): JADE, FIPA-OS, AOS, ZEUS, KADOMA, NOMADS, ARA, AGLETS, GRASSHOPPER, TRACY, AJANTA, LEAP, JACK, SEMOA. Многие из них успешно существуют в виде коммерческих проектов (таких как JACK) или проектов, позиционируемых как проекты с открытым исходным кодом (JADE, ZEUS и др.).

В 90-х годах возникла необходимость создания единых стандартов на разработку агентных систем. В этот период были основаны две организации MASIF (Mobile Agent System Interoperability Facility) и FIPA (Foundation of Physical Intelligent Agents). В результате их работы появились стандарт MASIF и стандарт FIPA [21], дающие рекомендации по созданию систем мобильных агентов и систем интеллектуальных агентов.

### *8.1.1. Стандартизация в области интеллектуальных агентов*

Цель FIPA – поддержка продвижения коммерческих приложений технологии интеллектуальных агентов путем разработки открытых спецификаций, поддерживающих интероперабельность агентов и агентных сервисов.

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».

Основными спецификациями FIPA являются следующие:

- 00001 – Abstract Architecture Specification;
- 00007 – Content Languages Specification;
- 00023 – Agent Management Specification;
- 00024 – Agent Message Transport Specification;
- 00025 – Interaction Protocol Library Specification;
- 00061 – ACL Message Structure Specification;
- 00063 – Control Agent Specification;
- 00082 – Network Management and Provisioning Specification;
- 00084 – Agent Message Transport Protocol for HTTP Specification;
- 00086 – Ontology Service Specification;
- 00087 – Agent Management Support for Mobility Specification;
- 00094 – Quality of Service Specification.

На

Рис. 33–36 представлены категории спецификаций FIPA.


Рис. 33. Категории спецификаций FIPA

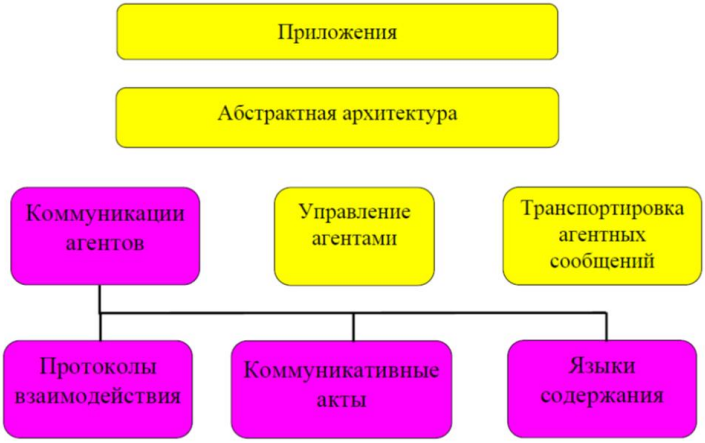



Рис. 34. FIPA: Коммуникация агентов

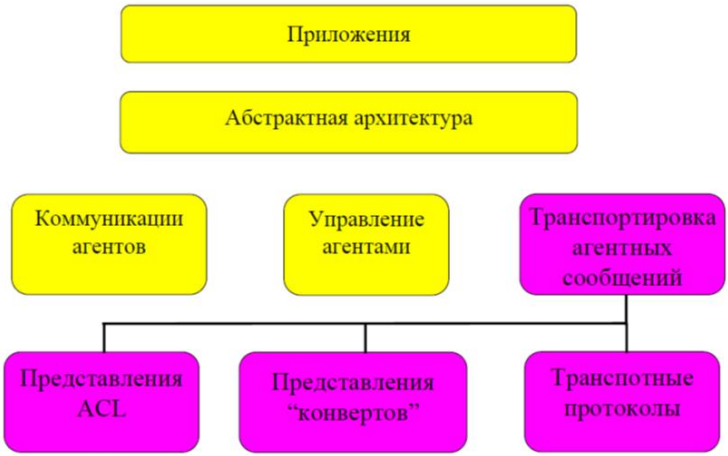



Рис. 35. FIPA: Транспортировка сообщений

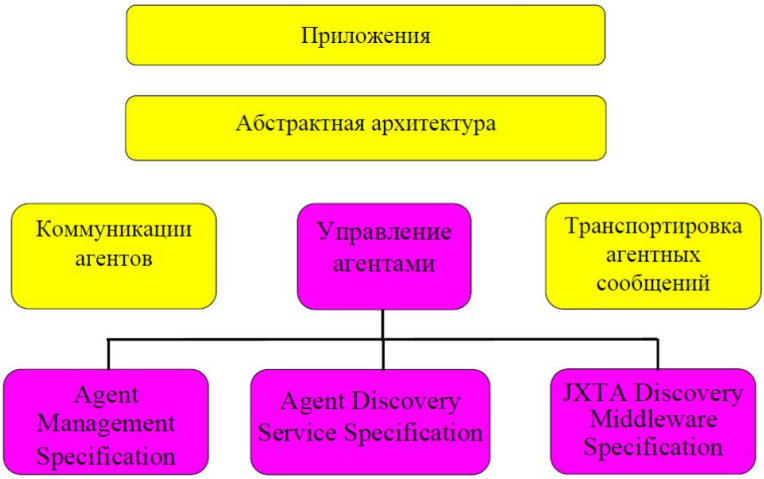



Рис. 36. FIPA: Управление агентами

### *8.1.2. Агентная платформа FIPA*

*Эталонная модель управления агентами* – множество логических компонентов, каждый из которых предоставляет множество возможностей, которые могут комбинироваться в физических реализациях агентных платформ.

*Агентная платформа* (АП) предоставляет физическую инфраструктуру, в которой могут быть размещены агенты. АП состоит из компьютеров, операционной системы, программного обеспечения, поддерживающего агентов, компонентов управления агентами по FIPA (DF, AMS и MTS) и агентов.

*Внутренняя конструкция* АП определяется разработчиками агентных систем и н является предметом стандартизации FIPA. FIPA рассматривает только способ коммуникации между «родными» для данной АП агентами и внешними или динамически регистрирующимися на АП агентами.

На рис. 37 представлена структура агентной платформы по FIPA.

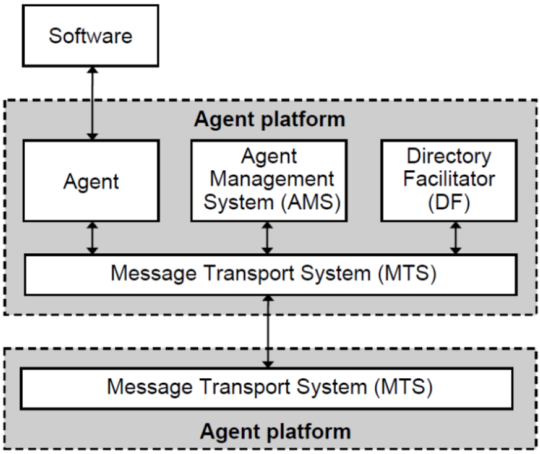



Рис. 37. Структура агентной платформы по FIPA

Рассмотрим компоненты агентной платформы.

**Agent Management System (AMS) – Система управления агентами**

AMS – обязательный компонент агентной платформы. Выполняет супервизорное управление доступом к агентной платформе и ее использованием. На одной агентной платформе может существовать только одна AMS. AMS поддерживает каталог идентификаторов агентов (AID), содержащих (кроме прочего) транспортные адреса агентов, зарегистрированных на данной АП. Предоставляет агентам сервис «белых страниц» для сохранения их текущего местонахождения. Каждый агент должен зарегистрироваться у AMS, чтобы получить корректный AID.

**Directory Facilitator (DF) – Служба Каталог**а

DF – обязательный компонент агентной платформы. Предоставляет агентам сервис «желтых страниц», с помощью которого агенты могут регистрировать свои сервисы и / или выполнять поиск в службе каталога с целью найти агента, предоставляющего заданные сервисы. На одной платформе может существовать несколько служб каталога, которые могут объединяться в федерации.

**Message Transport Service (MTS) – Служба транспортировки сообщений**

MTS поддерживает заданный по умолчанию метод коммуникации между агентами на различных АП.

**Software – стороннее (неагентное) программное обеспечение**

- Software – совокупность неагентного исполняемого кода, доступного через агента, не относится к агентной платформе. Агенты могут обращаться к стороннему программному обеспечению в следующих целях:
- добавление новых сервисов;
- приобретение новых коммуникационных протоколов;

- приобретение новых протоколов / алгоритмов безопасности;
- приобретение новых протоколов переговоров;
- обращение к инструментам, поддерживающим миграцию.

Именование агентов выполняется следующим образом. Идентификатор агента – расширяемая совокупность пар «параметр – значение», определяющих:

- имя агента (обязательный параметр);
- транспортные адрес(а) агента (необязательный параметр);
- адрес(а) сервиса разрешения имен (необязательный параметр).

Идентификатор агента описывается следующим образом:

(agent-identifier

:name <символическое имя агента>

:addresses <упорядоченная последовательность транспортных адресов, по которым можно контактировать с агентом>

:resolvers < упорядоченная последовательность AID, по которым можно контактировать с сервисами разрашения имен для агента>

)

Порядок перечисления адресов и AID в параметрах :addresses и :resolvers определяет предпочтение их использования.

Транспортный адрес обычно специфичен для Протокола Транспортировки Сообщений (Message Transport Protocol). Агент может поддерживать множество методов коммуникации, для этого в параметре :addresses указывается множество значений транспортных адресов. Сервис разрешения имен предоставляется AMS через

функцию поиска. Параметр :resolvers в AID содержит последовательность AID, по которым AID агента, в конечном счете, может быть разрешен в транспортный адрес или множество транспортных адресов. На рис. 38 представлен пример именования агента.

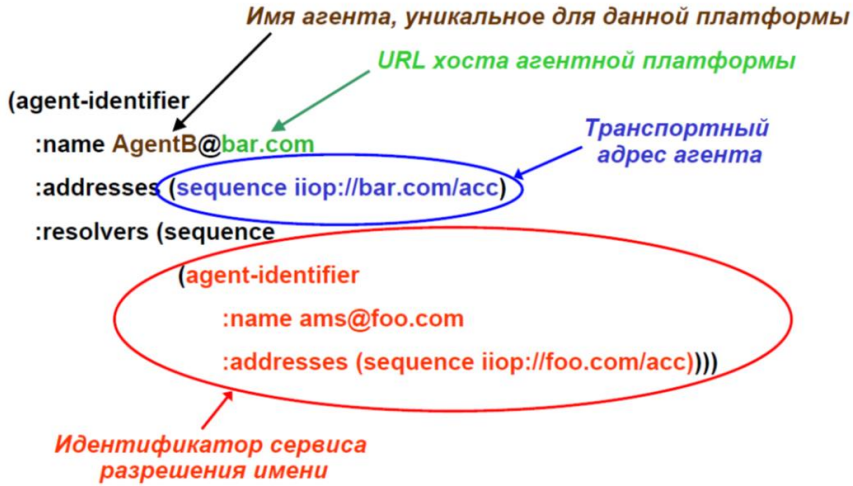



Рис. 38. Пример именования агента

Диаграмма жизненного цикла агента представлена на

Рис. 39 в виде графа, для которого определены состояния агента:

- Unknown – неопределенный;
- Initiated – инициализированный;
- Transit – перемещающийся;
- Active – активный;
- Waiting – ожидающий;
- Suspended – приостановленный;

и переходы из состояния в состояние:

- Create – создать;

- Initiate – инициализировать;
- Execute – выполнить;
- Move – переместить;
- Wait – ожидать;
- Wake-up – проснуться;
- Resume – продолжить;
- Suspend – приостановить;
- Destroy – разрушить;
- Quit – выйти.

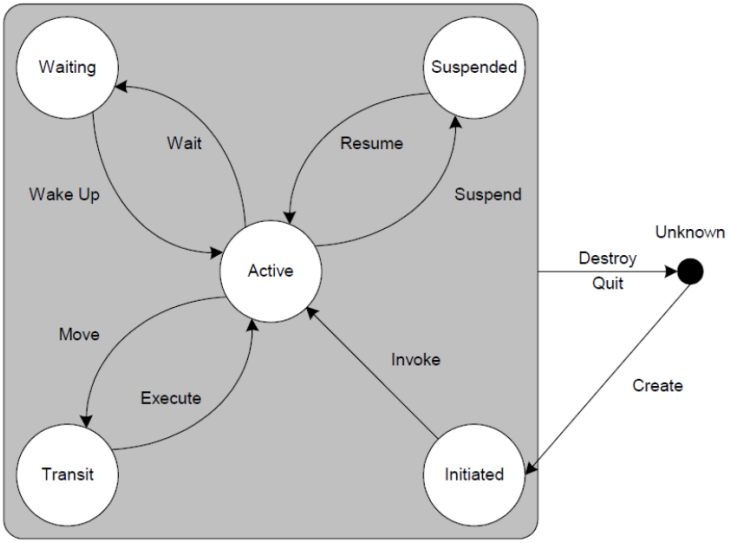



Рис. 39. Диаграмма жизненного цикла агента

Агенты взаимодействуют друг с другом посредством пересылки сообщений на языке FIPA-ACL. Формат сообщения на языке FIPA-ACL приведен на рис. 40.

```
( <Тип сообщения>
   :sender              // Отправитель сообщения
   :receiver            // Получатель(и) сообщения
   :content             // Содержание сообщения
   :reply-with          // Метка исходящего сообщения
   :in-reply-to         // Ссылка на входящее сообщение
   :replyBy             // Лимит времени на ответ
   :language            // Язык сообщения
   :ontology            // Онтология
   :protocol            // Используемый протокол общения
   :conversation-id     // Идентификатор разговора
)
```

Рис. 40. Формат сообщения на языке FIPA-ACL

Например, в сообщении ниже агент agent1 информирует агента agent5 о том, что цена на товар good200 составляет 150 д.е.

```
(inform
   :sender  agent1
   :receiveragent5
   :content (price good200 150)
   :language        s1
   :ontology        hp1-auction
)
```

В более сложном сообщении, приведенном на рис. 41, имена агентов, отправляющих и получающих сообщение, запрашиваются на АП, а сообщение представляет собой запрос на регистрацию студента Иванова из группы 8307 на курс по МАС преподавателя Пантелеева.

Пример протокола FIPA для обмена сообщениями между агентами приведен на рис. 42.

```
(REQUEST
   :sender (agent-identifier :name SendAg@host:1099/JADE)
   :receiver (set (agent-identifier :name RecvAg@host:1099/JADE))
   :content "((action
              (agent-identifier :name RecvAg@host:1099/JADE)
              (REGISTER :student (STUDENT :name Ivanov
                                          :groupnumber \"8307\")
                        :course (COURSE :name \"MAS Course\"
                        :instructor (INSTRUCTOR :name Panteleev
                                                :dept CS)))))"
   :language  fipa-sl0
   :ontology  eLearning
)
```

Рис. 41. Пример сообщения на языке FIPA-ACL

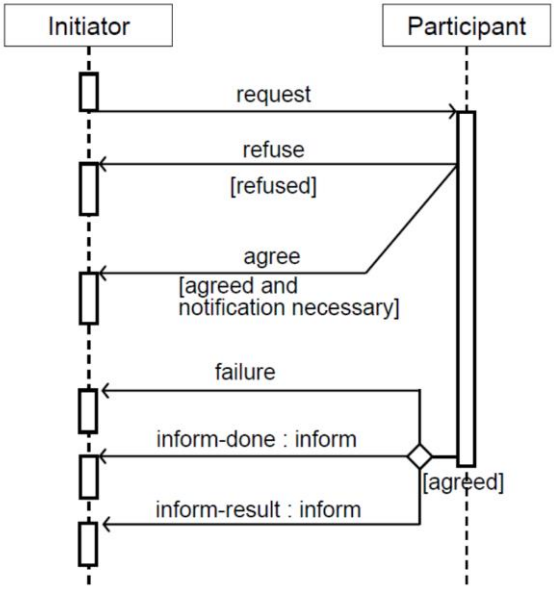



Рис. 42. FIPA-Request-Protocol

### *8.1.3. Агентная платформа JADE*

Одной из наиболее популярных агентных платформ в настоящее время является платформа JADE (Java Agent DEvelopment Framework). Проект JADE разрабатывается компанией Telecom Italia Lab с 2000 г. [22].

Агентная платформа JADE является типичным Middleware, т.е. программным обеспечением (ПО) среднего уровня (рис. 43), представляющим собой набор средств для создания и управления системой с множеством агентов [23].

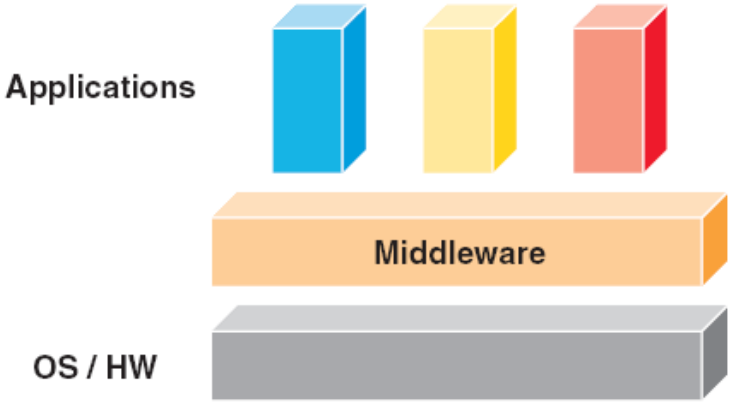



Рис. 43. Место Middleware в структуре ПО

Платформа разработки мультиагентных систем JADE включает в себя динамическую среду, где могут «жить» JADE агенты; библиотеку классов, которую программисты могут использовать для разработки собственных агентов; набор графических инструментов, позволяющих управлять активностью запущенных агентов.

JADE предоставляет программисту – разработчику агентных систем следующий набор средств [22]:

1. FIPA-compliant Agent Platform – агентную платформу, основанную на стандарте FIPA и включающую обязательные типы системных агентов, которые автоматически активируются при запуске платформы.

2. Distributed Agent Platform – распределенную агентную платформу, которая может использовать несколько компьютеров (узлов), причем на каждом узле запускается только одна Java Virtual Machine. Агенты исполняются как Java-потоки. Для доставки сообщений между агентами, в зависимости от их местонахождения, используется соответствующий транспортный механизм – Multiple Domains support – множество основанных на FIPA-спецификациях специализированных агентов, которые могут объединяться в федерацию, реализуя таким образом мультидоменную агентную среду.

3. Multithreaded execution environment with two-level scheduling. Каждый JADE-агент имеет собственный поток управления, но он также способен работать в многопотоковом режиме. Java Virtual Machine проводит планирование задач, исполняемых агентами или одним из них.

4. Object-oriented programming environment. Большинство концепций, свойственных FIPA-спецификации, представляются Java-классами, формирующими интерфейс пользователя.

5. Library of interaction protocols. Использование стандартных интерактивных протоколов fipa-request и fipa-contract-net. Для того чтобы создать агента, который мог бы действовать согласно таким протоколам, разработчикам прикладных программ нужно только имплементировать специфические доменные действия, в то время как вся независимая от прикладной программы протокольная логика будет осуществляться системой JADE.

6. Administration GUI. Простые операции управления платформой могут исполняться через графический интерфейс, отображающий активных агентов и контейнеры агентов. Используя GUI, ад-

министраторы платформы могут создавать, уничтожать, прерывать и возобновлять действия агентов, создавать иерархии доменов и мультиагентные федерации.

Платформа JADE написана на языке программирования Java с использованием Java RMI, Java CORBA IDL, Java Serialization и Java Reflection API. Она упрощает разработку мультиагентных систем благодаря использованию FIPA-спецификаций и инструментов (tools), которые поддерживают фазы исправления ошибок (debugging) и развертывания (deployment) системы. Эта агентная платформа может распространяться среди компьютеров с разными операционными системами, и ее можно конфигурировать через удаленный GUI-интерфейс. Процесс конфигурирования этой платформы достаточно гибкий. Единственным требованием такой системы является установка на компьютере Java Run Time требуемой версии.

**Архитектура агентной платформы JADE**

Платформа JADE является распределенной и представляет собой набор контейнеров (рис. 44).

Контейнером называется динамическая среда исполнения мультиагентных приложений, в которой находятся агенты. Каждый контейнер может содержать несколько агентов. Набор активных контейнеров называется платформой. Один из контейнеров всегда является главным (Main container), все остальные контейнеры связываются с ним и регистрируются в момент запуска. Поэтому первым контейнером при старте платформы должен быть главный, а все остальные контейнеры должны быть «обыкновенными» (т.е. неглавными) контейнерами и должны заранее «знать», как найти главный контейнер, на котором они будут регистрироваться, т.е. должны иметь данные о хосте и порте.

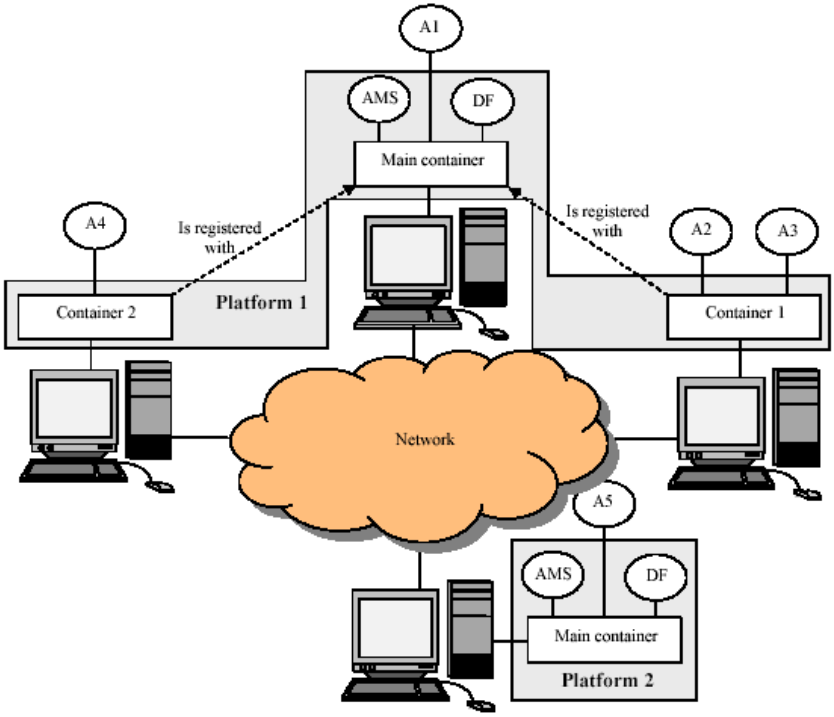



Рис. 44. Платформа JADE

Другой главный контейнер, запущенный где-либо в сети, представляет собой другую платформу, на которой могут зарегистрироваться новые обычные контейнеры. Рис. 44 иллюстрирует эту концепцию на основе примера, показывающего две JADE-платформы, состоящие из трех и одного контейнера соответственно. JADE-агенты определяются с уникальными именами. При условии, что они знают имена других агентов, они могут общаться, независимо от их фактического местонахождения: в общем контейнере (агенты A2 и A3), в разных контейнерах на одной платформе (агенты A1 и A2), или вообще на разных платформах

(A4 и A5). Пользователю не обязательно знать, как работает динамическая среда JADE, но необходимо запускать её перед началом выполнения своих агентов.

Кроме возможности приёма регистраций от других контейнеров, главный контейнер отличается от обычного контейнера тем, что содержит два специальных агента, автоматически запускаемых одновременно с контейнером:

1. AMS (Agent Management System – система управления агентами) обеспечивает службу управления агентами, которая позволяет создавать и удалять агентов, а также содержит в себе пространство имен агентов. Имя агента является уникальным и имеет следующий формат: <nickname>@<platform-name>. Зная имена друг друга, агенты могут обмениваться сообщениями как внутри контейнера и платформы, так и между различными платформами.

2. DF (Directory Facilitator – менеджер директорий) представляет собой службу «желтых страниц» (yellow pages), где агенты могут публиковать информацию о предоставляемых ими сервисах. С помощью DF агент может находить агентов, предоставляющих необходимые ему сервисы, и вступать с ними в переговоры. Внутри одной платформы может существовать несколько DF, предоставляющих информацию о различных группах сервисов или о сервисах различных групп агентов.

**Пример посылки сообщения**

На рис. 45 показан пример создания и отправки сообщения в платформе JADE. Отправка сообщения другому агенту реализуется простым заполнением полей объекта ACLMessage и последующим вызовом метода send() класса Agent. Посылающий агент информирует получателя с именем Peter, что «сегодня идет дождь».

```
ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
msg.addReceiver(new AID("Peter", AID.ISLOCALNAME));
msg.setLanguage("English");
msg.setOntology("Weather-forecast-ontology");
msg.setContent("Today it's raining");
send(msg);
```

Рис. 45. Пример посылки сообщения в платформе JADE

### *8.1.4. Агентная платформа JACK*

JACK разрабатывается AOS (Autonomous Decision-Making Software) Group. Приложения JACK® состоят из набора автономных агентов, которые получают входные данные из окружающей среды и общаются с другими агентами. Это обеспечивает системным интегратором с очень мощной формой инкапсуляции. Каждый агент определяется с точки зрения его целей, знаний и социального потенциала, а затем самостоятельно выполняет свою функцию в той среде, в которую он встроен [24].

Это очень эффективный способ создания приложений для динамичных и сложных сред – каждый агент преследует свои цели, реагирует на события и общается с другими агентами в системе. Нет необходимости явно программировать взаимодействия всего приложения, взаимодействия возникают как побочный продукт индивидуальных целей и возможностей субъектов-агентов.

JACK – развитая кросс-платформенная среда для создания, запуска и интеграции многоагентных систем коммерческого класса. JACK построен на логической основе: BDI (убеждения / желания / намерения). BDI является интуитивно понятной и мощной абстракцией, которая позволяет разработчикам управлять сложностью проблемы. В JACK агенты определены с точки зрения их убеждений (что они знают и знают, как это сделать), их желаний (цели, которых

они хотели бы добиться), и их намерений (целей, которые они в настоящее время стремятся достигнуть).

Полностью написанный на Java™, JACK портативная платформа, работающая на любых компьютерах – от ПК до мощных многопроцессорных серверов. JACK может работать с несколькими потоками между несколькими процессорами, имеет независимый от платформы графический интерфейс и легко интегрируется со сторонними библиотеками.

**Архитектура агентной платформы JACK**

Агент в JACK фактически является расширением Java-класса. Действия, выполняемыми агентом, описывается при помощи планов (примерный аналог поведений в JADE). Убеждения агента описываются при помощи наборов убеждений (beliefsets). Эти элементы будут рассмотрены далее. Таким образом, при декларации агента описываются следующие элементы:

- набор используемых планов,
- набор используемых убеждений (beliefsets),
- генерируемые события,
- обрабатываемые события,
- реализуемые возможности (capabilities).

Возможности являются средством для выделения в отдельные элементы близких по смыслу частей агента. Возможности могут быть использованы в нескольких классах агентов. Такой подход позволяет избежать дублирования кода и упростить структуру агента. Возможность содержит те же элементы и описывается практически так же, как и агент.

Кроме отдельных агентов, JACK позволяет создавать команды агентов и задавать командное поведение.

**Планы**

План является набором инструкций, выполняемых агентом. Планы могут вызываться при поступлении события либо вызы-

ваться из других планов. Каждый план может вызываться при поступлении только одного типа (класса) события. Агент может использовать несколько планов, соответствующих одному и тому же событию, их выбор зависит от типа события и будет описан ниже. Каждый план имеет два метода:

- relevant(EventType event) – проверяет релевантность плана конкретному экземпляру события. Например, при определенных параметрах сообщения этот план может быть уже неприменим к этому сообщению;
- applicable() – проверяет применимость плана в данных условиях, в соответствии с убеждениями агента и, возможно, значениями каких-либо внутренних переменных. Например, план Б требует, чтобы перед его выполнением был выполнен план А.

**Убеждения**

Для хранения убеждений агента используется реляционная модель, основным элементом в которой являются наборы убеждений (beliefsets). Набор убеждений можно сравнить с таблицей в реляционной базе данных. Наборы убеждений поддерживают запросы в виде логических выражений. Кроме того, автоматически поддерживается логическая целостность данных. При изменении данных в наборе убеждений может автоматически генерироваться соответствующее событие.

Наборы убеждений разделяются на убеждения открытого мира (Open world beliefsets) и убеждения закрытого мира (Closed world beliefsets). Они отличаются тем, что убеждения закрытого мира могут принимать только значения истина или ложь, в то время как убеждения открытого мира могут принимать третий тип значения «неопределено».

**Представления**

Представления (views) используются для представления данных, хранимых в наборах убеждений в заданном виде. В отличие от наборов убеждений, они не хранят сами данные, а только запросы к ним.

**События**

События используются в JACK для обмена сообщениями между агентами и между планами внутри агента, оповещении агентов об изменении наборов убеждений, оповещении о событиях таймера и т.д. Все события можно разделить на две группы, различающиеся способом их обработки агентом.

Обычные события являются аналогом сообщений в JADE. При их обработке агент действует по следующему алгоритму. Ищутся все планы, которые могут обрабатывать данный тип события. Из них выбираются релевантные и применимые в данных условиях, и выполняется первый план, удовлетворяющий обоим условиям. Если план выполнился успешно, событие считается выполненным успешно, и наоборот.

BDI события (Belief-Desire-Intention events) позволяют установить агенту цель (успешное выполнение события), которой он будет добиваться. Обработка таких событий происходит следующим образом. При поступлении BDI события вызывается некоторый специальный план, реализующий металогику агента, в котором выбирается один или составляется список планов, которые необходимо исполнить. После этого план(ы) выполняются. При успешном завершении плана событие считается завершенным успешно, в противном случае агент может составить альтернативный список планов и запустить их исполнение, и так далее. Если в конечном итоге альтернативных планов не осталось, или агент посчитал их выполнение нецелесообразным, событие считается завершенным неудачно.

**Инструменты для разработки мультиагентных приложений**

JACK предоставляет два инструмента для разработки мультиагентных приложений.

1. Design tool (рис. 46) – позволяет, аналогично Modeler в Living Systems, проектировать приложение в виде UML-подобных диаграмм и генерировать по этим диаграммам исходный код приложения.

2. Plan editing tool (рис. 47) – позволяет создавать планы в виде диаграмм. Планы компилируются непосредственно в исполняемый код. При отладке данная утилита позволяет следить за ходом исполнения плана, подсвечивая выполняемый элемент диаграммы.

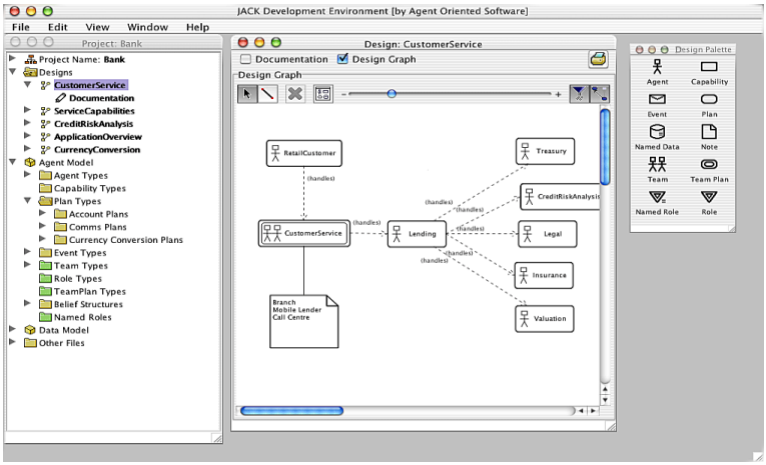



Рис. 46. Design tool

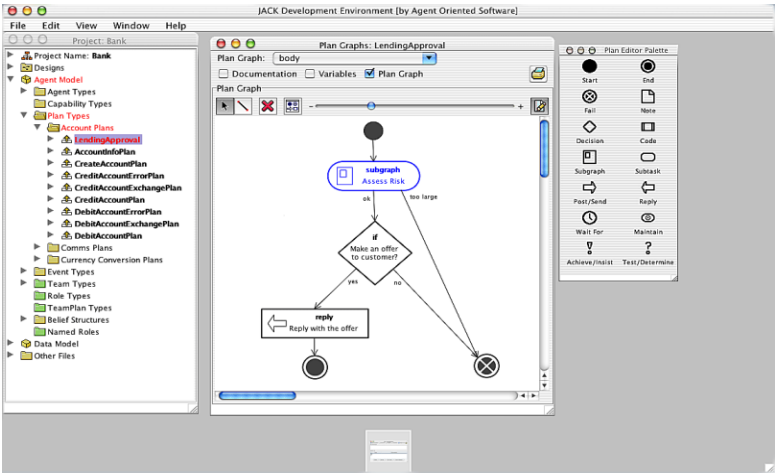



Рис. 47. Plan editing tool

### *8.1.5. Агентная платформа Living Systems – Living Systems Technology Suite*

Агентная платформа Living Systems Technology Suite (WhitesteiN Technologies) реализована на Java и является middleware [25] (рис. 48).

Living Systems Technology Suite состоит из двух частей:

1. Среда исполнения и библиотека классов для поддержки мультиагентных приложений;
2. Набор средств разработки.

Среда исполнения Living Systems поддерживает три режима исполнения мультиагентных приложений (

Рис. ):

1. Single – приложение и база данных находятся на одном компьютере;
2. Cluster – приложение распределено на нескольких хостах, которые связаны с единой базой данных;
3. Federation – несколько кластеров, объединенных в одно приложение.

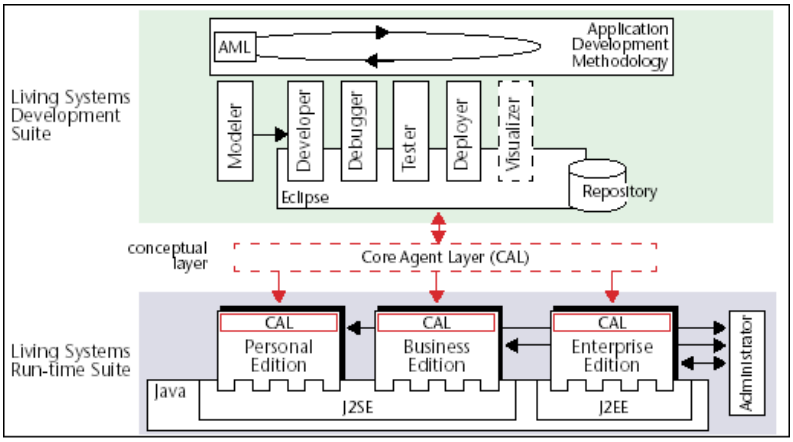



Рис. 48. Агентная платформа Living Systems

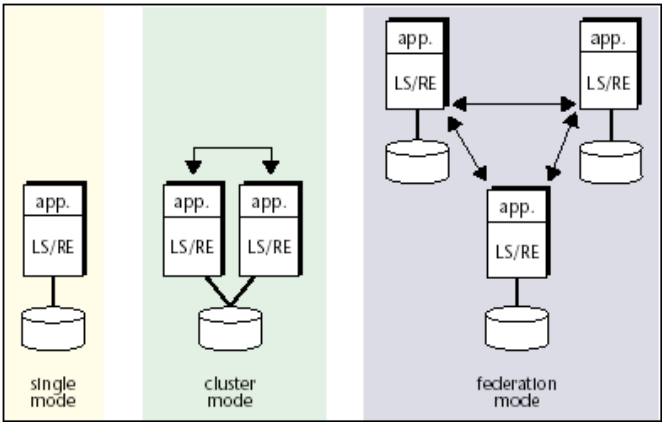



Рис. 49. Режимы исполнения

Для предоставления доступа к приложению конечному пользователю, а также для интеграции с другими системами поддерживаются технологии RMI, JMS, SOAP (рис. 50).

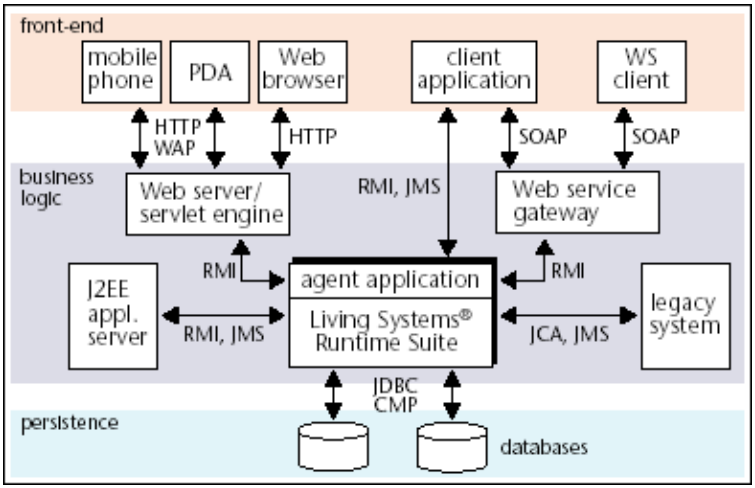



Рис. 50. Технологии для интеграции

Living Systems поддерживает три типа агентов:

1. Autonomous Agent (автономный агент) – проактивный, способен обмениваться сообщениями;

2. Data Agent (агент данных) – реактивный, предоставляет доступ к хранилищу данных;

3. Servant Agent (сервисный агент) – реактивный, предоставляет доступ к часто используемым функциям.

**Средства разработки**

Living systems предлагает очень богатый набор средств проектирования, разработки, отладки и тестирования мультиагентных приложений. Большинство средств реализовано в виде плагинов к open-source среде разработки Java-приложений Eclipse, что является достаточно удобным для разработчика.

**Modeler**

Modeler является средством проектирования МА-приложений на основе AML-диаграмм (рис. 51). AML является расширением языка UML, добавляющим поддержку мультиагентного подхода к разработке приложений. По AML-диаграмме поддерживается генерация кода приложения.

**Debugger**

Debugger является плагином стандартного отладчика Eclipse и поддерживает всю его функциональность, например, возможность удаленной отладки. Кроме того, Debugger позволяет устанавливать точки останова не только на весь класс агента, но и на конкретный экземпляр. Точки останова могут устанавливаться при выполнении следующих условий:

- агент получает определенное сообщение;
- агент отправляет определенное сообщение;
- агент пытается записать или прочитать определенную переменную в Agent Data Storage;
- вызывается определенный обработчик сообщений;
- вызывается определенный фрагмент.

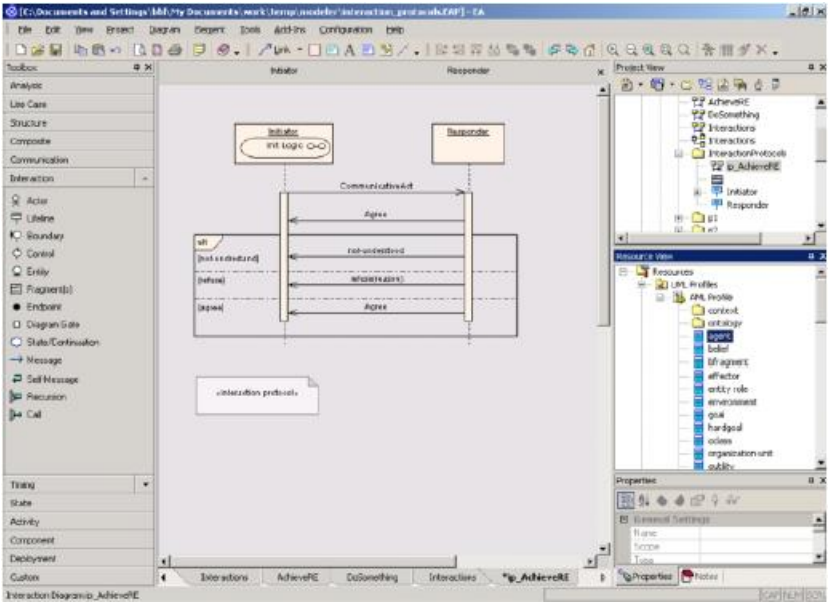


Рис.51. Modeler

**Tester**

Tester является утилитой для поддержки тестирования функциональности МА-приложения. Поддерживает JUnit tests и позволяет автоматически создавать шаблоны тестов для классов агентов и сообщений. Поддерживает тестирование переговоров внутри группы агентов.

**Deployer**

Deployer является утилитой, помогающей настроить параметры среды исполнения МА-приложения и запустить приложение в этой среде (рис. 52).

Заметим, что рассмотренные агентные платформы предоставляют развитую функциональность для реализации взаимодействия агентов. Однако, в большинстве из них реализованы делибератив-

ные агенты, принципы построения которых основаны на логических рассуждениях (логическом выводе). Использование подобных агентов для принятия решений в реальном времени затруднительно, т.к. рассуждения, основанные на логике предикатов первого порядка, могут потребовать слишком много времени.

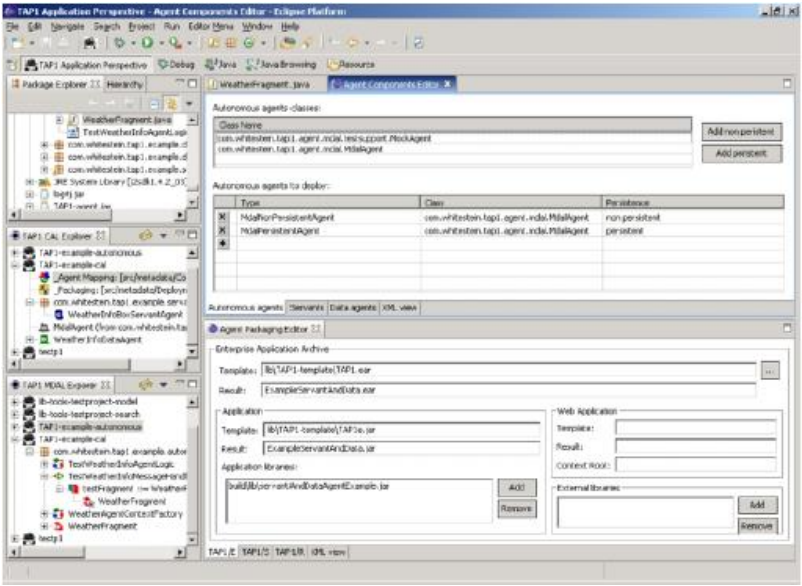


Рис. 52. Deployler

## 8.2. Языки программирования агентов

### *8.2.1. Требования к языкам программирования агентов*

Разрабатываемые в настоящее время МАС используют большой спектр различных базовых языков, но, к сожалению, пока ни один из них не может рассматриваться как истинно «агентно-ориентированный». Чаще всего, делаются попытки использовать тра-

диционные языки программирования, вводя в них соответствующие расширения.

Основные требования к языкам программирования агентов [16]:

1. *Обеспечение переносимости кода* на различные платформы. Чтобы обеспечить мобильность агента, язык должен поддерживать механизм посылки, передачи, получения и выполнения кодов, содержащих агентов. Имеются два различных подхода, которые решают проблему мобильности. Первый – это передача агента в текстовой форме, как специального сценария (script) с последующей интерпретацией этого сценария на принимающей машине. Второй – передача агента в форме машинно-независимого байт-кода. Этот байт-код генерируется транслятором на этапе создания агентской системы, посылается по сети и выполняется интерпретатором байт-кодов на принимающем компьютере.

2. *Поддержка сетевого взаимодействия*. Свойство агентов участвовать в переговорах и многие другие характеристики агентов опираются на доступ к удаленным ресурсам.

3. *Поддержка символьных вычислений*. В рамках современных взглядов агент должен активно использовать достижения и методы ИИ. Поэтому необходимо иметь поддержку символьных вычислений и, возможно, логического программирования, встроенную в язык (подобно PROLOG- или LISP-системам).

4. *Многопоточная обработка (Multithreading)*. В ряде случаев агент должен выполнять некоторые действия одновременно. Тем самым, язык программирования агентов должен включать поддержку параллельного выполнения различных функций агента (типа threads) и различных примитивов синхронизации (семафоры, мониторы, критические секции и т.д.).

5. *Безопасность*. Мобильные агенты, которые приходят из сети, могут таить в себе множество опасностей для принимающей

машины, так как они работают в ее адресном пространстве. Система безопасности должна предотвращать любые несанкционированные действия агента.

6. *Расширенная объектная ориентированность*. Язык программирования агентов должен иметь механизмы наследования, рассматривать вызовы процедур как сообщения, передаваемые от одних объектов к другим, включать возможности синхронного и асинхронного взаимодействия объектов, а также допускать параллельность внутри объекта.

7. *Языковая поддержка свойств агента*. Следует иметь поддержку специфических для агента свойств, встроенную в язык на уровне синтаксических конструкций, так, чтобы, например, свойства типа «beliefs-desires-intentions» (BDI), инструкции для переговоров и обеспечения мобильности, места встречи, и т.д. могли бы быть выражены с помощью соответствующих примитивов языка.

8. *Эффективность*, достаточная для потребностей прикладных программ.

Приведем классификацию языков, наиболее часто используемых в технологии интеллектуальных агентов [16]:

1. Универсальные языки программирования (Java);
2. Языки, «ориентированные на знания»:
   - языки представления знаний (KIF),
   - языки переговоров и обмена знаниями (KQML, AgentSpeak, April),
   - языки спецификаций агентов;
3. Специализированные языки программирования агентов (TeleScript);
4. Языки сценариев и scripting languages (Tcl/Tk);
5. Символьные языки и языки логического программирования (Oz).

Далее приведены краткие характеристики языков, наиболее часто используемых в технологии интеллектуальных агентов.

### *8.2.2. Язык Java*

Язык Java – один из наиболее популярных языков, используемых в последнее время для программирования агентов. Он представляет собой чисто объектно-ориентированный язык программирования с сильной типизацией. Отметим, что Java подобен C++ по синтаксису, но по идеологии более схож со Smalltalk и Objective C. Система программирования на Java включает в себя виртуальную машину Java и транслятор с Java в байт-код.

Язык Java предусматривает создание приложений, переносимых на различные платформы. Программа, написанная на Java, компилируется в специальный машинно-независимый байт-код. Затем этот код может быть исполнен с помощью интерпретатора Java на любом компьютере, где реализована Java Virtual Machine. Тем самым обеспечивается платформо-независимость Java приложений на уровне байт-кода, который может приходить откуда угодно, включая Web-страницу, в которой содержится ссылка на него. Java Virtual Machine функционирует в режиме мультизадачности и поддерживает threads-процессы (многопоточные). Средства создания и синхронизации таких процессов включены в Java на уровне языковых конструкций и классов. Средства многозадачности также призваны обеспечить реакцию системы в реальном времени для мультимедийных приложений, критичных ко времени.

Язык Java был спроектирован с учетом возможности создания приложений, работающих в распределенной среде.

### *8.2.3. Язык KQML*

В свою очередь, KQML – это язык и протокол для поддержки взаимодействия агентов в распределенных приложениях. Примеры возможных применений языка включают различные варианты

МАС, в том числе, системы с интеллектуальными агентами планирования, поддерживающими распределенную обработку. Типичные компоненты систем, основанных на знаниях:

- приложение, рассчитанное на неподготовленного конечного пользователя (End User Application);
- система, основанная на знаниях (Knowledge Based System);
- хранилище специальных знаний (Knowledge Base Repository), а именно, хранилище знаний о содержимом базы знаний, которое используется для отслеживания истории доступа, обеспечения целостности и других аспектов функционирования баз знаний;
- система управления базами данных (Database System);
- активные сенсоры (Active Sensors), отвечающие за обмен данными (знаниями) с «внешним миром».

Возможны различные способы взаимодействия перечисленных компонент.

Разработчики KQML выделяют 3 аспекта, особенно важных для коммуникации интеллектуальных агентов:

1. Связность (connectivity) – определение способа связывания агентов между собой;

2. Архитектура – акцентирует внимание на способе построения будущей системы (будет ли система статической, когда все компоненты известны уже на этапе проектирования/ реализации или динамической);

3. Собственно коммуникация, которая связана с рассмотрением синхронности/асинхронности обмена сообщениями между агентами.

При разработке языка были сделаны специальные предположения об основных моментах его использования для описания коммуникаций МАС.

**Архитектура**

Агенты – это отдельные процессы, работающие в одном адресном пространстве или на различных компьютерах, соединенных посредством Internet.

**Коммуникации**

Язык поддерживает следующие стратегии передачи информации: point-to-point, multicast, broadcast.

**Синтаксис**

Выражения между процессами передаются как потоки ASCII-символов.

**Протокол**

Для транспортной поддержки KQML был создан специальный протокол – SKTP (Simple Knowledge Transfer Protocol).

Язык KQML состоит из трех уровней:

- содержимого,
- сообщений,
- коммуникаций.

KQML-выражение можно рассматривать как выражение-содержимое, помещенное в сообщение-обертку, которое, в свою очередь, помещено в коммуникационную обертку.

На уровне содержимого находится представление знаний на некотором языке.

Уровень сообщений добавляет дополнительные атрибуты, такие как описание языка, на котором выражено содержимое, его онтологию и тип используемого метода переговоров.

Коммуникационный уровень добавляет информацию об отправителе и получателе сообщения, а также указывает, является сообщение синхронным или асинхронным.

Основу синтаксиса языка KQML составляют имена примитивных действий – сообщений, которые в английской транскрипции называются «performatives».

Например,

- Спроси всех (одного) ...
- Один хочет узнать у всех ...
- S хочет получить следующий ответ от R ...
- S хочет, чтобы R исполнил...
- Спроси, если ...

Тип сообщений определяет, что можно сделать с предложениями, содержащимися в сообщении.

Правила поведения агента описываются так:

**NAME** Rule name
**WHEN**
message condition(s)
**IF**
mental condition(s)
**THEN**
Private Action(s)
Mental change(s)
Message(s)

В заключение приведем пример на языке KQML поведения агента при зеленом цвете светофора:

**NAME** “Greenlight Move Forward”
**WHEN**
?KQMLMessage.Performative EQUALS TELL
?KQMLMessage.Sender EQUALS “stoplight-agent”
?KQMLMessage.Content EQUALS String
?KQMLMessage.Ontology EQUALS “Stoplight”
**IF**
?KQMLMessage.Content EQUALS “stoplight-green”

currentMotion.Status EQUALS stoppedAtRedLight
currentLocation EQUALS nextIntersection
NOT ( currentLocation EQUALS destination )
FOR_ALL (?BlockedIntersection,
NOT (?BlockedIntersection.Location EQUALS currentLocation ))

**THEN**

DO (Go (trafficSpeed))
DO ($nextIntersection = getNextIntersection currentLocation, currentMotion.Direction))
Assert (SET_VALUE_OF currentMotion.Status TO moving )
Assert (SET_VALUE_OF nextIntersection TO $nextIntersection )
SEND (performative = REPLY, receiver = “stoplight-agent”,
Content = “acknowledged”,
In-reply-to = ?KQMLMessage.Reply-with)

### *8.2.4. Язык KIF*

Язык KIF предназначен для обмена знаниями между различными программами, написанными различными людьми, на разных языках и в разное время. Он имеет декларативную семантику (значение выражений, представленных на этом языке, может быть понятно без обращения к специальному интерпретатору), логически полон (позволяет работать с произвольными предложениями логики предикатов первого порядка), обеспечивает представление знаний о представлении знаний, а также представление немонотонных правил вывода, определение объектов, функций и отношений.

KIF не претендует на статус языка для внутреннего представления знаний (хотя и может быть использован для этих целей).

Обычно программа получает внешние знания, выраженные на языке KIF, затем транслирует их во внутреннее представление (списки, фреймы и т.д.) и выполняет все вычисления, используя эти внутренние представления. Язык KIF используется при коммуникациях с внешними программами.

Основные свойства языка KIF:

1. Язык является декларативным, чем отличается от языков типа Prolog.
2. Язык является логически полным.
3. Язык является транслируемым, т.е. в нем имеются средства для трансляции декларативных баз знаний в/из типичных языков представления знаний.
4. Язык является хорошо читаемым, что упрощает его использование разработчиками баз знаний.
5. Язык может быть использован как язык представления знаний.

Применение KIF представляется следующим образом. Ядро агента, а именно, подсистемы управления памятью и планирования, база знаний и т.д. пишутся на C++ или на одном из языков типа Java, Telescript (если нас интересует способность агента к мигрированию по сетям), а KIF используется как язык для обмена знаниями/данными с другими агентами. Для этого агент должен иметь подсистему перетрансляции с языка внутреннего представления знаний на KIF. Язык KIF можно также использовать и как язык представления собственных знаний агента, в этом случае транслятор не нужен.

### *8.2.5. Язык AgentSpeak*

Аналогом класса в данном языке выступает семейство, а представителем (экземпляром) семейства является агент. Каждый агент обладает базой данных, множеством сервисов и множеством

планов – процедур, о которых известно лишь то, выполнены они или нет. Язык обеспечивает распределенное хранение информации в пространстве функционирования агентов.

Каждый агентский сервис может быть одного из следующих трех типов: achieve, query, told. Каждый агентский план определяется при помощи имени и описания абстрактной ситуации, в которой он может быть применен. Во время работы агента, если план применим в текущей ситуации, выполняются действия, связанные с данным планом.

Передача сообщений носит синхронно/асинхронный характер. Каждому агенту присваивается личный почтовый ящик, в который складываются все приходящие на его адрес сообщения. Кроме передачи сообщений типа агент-агент, поддерживаются коммуникации агент-семейство агентов. Последнее позволяет агентам использовать сервисы обработки сообщений на базисе свойств семейства. Сообщения маршрутизируются по пространствам сообщений, каждое из которых присоединено к конкретному семейству агентов. Сообщения, используемые в языке, могут относиться к одному из трех типов: информировать, запрашивать с ожиданием ответа, запрашивать без ожидания ответа.

Язык требует, чтобы все семейства агентов были определены до начала выполнения средой агентских заданий (во время компиляции), т.е. динамическое создание агентов не поддерживается.

Каждый агент может находиться в одном из трех состояний: активный, ожидающий, неработающий (idle).

По прибытии к агенту сообщения оно помещается в его почтовый ящик, далее агент начинает обрабатывать сообщение. Сначала по типу речевого сообщения (achieve, query, told) выбираются планы, которые могут быть потенциально применимы в данной ситуации (множество подходящих планов), далее выполняется просмотр абстрактных ситуаций множества выбранных на предыду-

щем шаге планов (множество применимых планов). Затем для последнего множества строятся реализации (ссылки). По умолчанию система выбирает один план из множества применимых и начинает выполнять действия, указанные в целевой части описания плана. Такой выбранный план называется намерением. В любой момент выполнения агент может обладать несколькими намерениями – потоками. Вследствие этого, агент является многопоточным процессом. Внешние и внутренние сервисы агента функционируют, конкурируя за ресурсы, поскольку агент может выполнять как внешние, так и внутренние сервисы.

### *8.2.6. Язык TeleScript*

Первая коммерческая реализация концепции мобильного агента была сделана в среде TeleScript-технологии фирмы General Magic. Эта технология основана на метафоре электронного рынка – общедоступной сети (public network), которая позволяет продавцам и потребителям товаров и услуг находить друг друга и заниматься совместным бизнесом.

Технология TeleScript оперирует следующими понятиями:

- места (places),
- агенты (agents),
- перемещения (travels),
- встречи (meetings),
- соединения (connections),
- полномочия (authority),
- разрешения (permits).

Основные понятия языка:

- *Места*. Технология TeleScript рассматривает компьютерную сеть как множество мест. Место – стационарный процесс на сервере, предлагающий услуги входящему агенту.

- *Агенты*. Коммуникационное приложение трактуется как набор агентов. Каждый агент занимает конкретное место. Однако, агент может перемещаться от места к месту и даже быть распределенным, занимая несколько различных мест в одно и то же время. Агентские процедуры выполняются параллельно.

- *Перемещение*. Агенту предоставляется возможность путешествовать с места на место. Перемещение – отличительный признак системы удаленного программирования. Оно позволяет агенту получить удаленную услугу и затем вернуться на место его старта. Система TeleScript позволяет коммуникационному пакету (computer package) – агенту (его процедурам и состоянию) перемещаться между компьютерами.

- *Встречи*. Встреча позволяет агентам вызывать процедуры друг друга. Встречи – это то, что «заставляет» агента перемещаться. Для встречи с расположенным рядом (co-located) агентом, данный агент выполняет инструкцию meet. Инструкция содержит требование (petition) – данные, определяющие агента, который «хочет» встретиться, и другие параметры встречи. Meet-инструкция позволяет покупателям и продавцам осуществлять транзакции.

- *Соединения*. Они позволяют агентам обмениваться информацией с разных мест. Для соединения агент выполняет connect-инструкцию. Данная инструкция содержит несколько параметров, таких как цель (target) соединения.

- *Полномочия*. Технология позволяет агенту или месту распознавать полномочия другого агента/места, причем агент или место не могут ни скрывать, ни фальсифицировать свои полномочия. Данная возможность позволяет защитить агентов и места от проникновения вирусов.

- *Разрешения*. Технология позволяет управлять назначением полномочий.

Язык программирования позволяет разработчику коммуникационного приложения определять алгоритмы функционирования агентов и данные, переносимые агентами во время перемещения по сети. Язык включает в себя возможности, предлагаемые C и C++. Приложение может быть написано целиком на языке TeleScript, но чаще агенты и оболочки мест пишутся с помощью TeleScript, а стационарные части приложения (интерфейсы с пользователем, базами данных и т.д.) – на C или C++.

# ЗАКЛЮЧЕНИЕ

Сложные системы обладают следующей функциональностью:

- Выдвигают цели, формируют стратегии и ставят задачи.
- Воспринимают и анализируют ситуации, планируют свои действия для решения задач и исполняют намеченные планы в изменяющейся среде.
- Моделируют окружающий мир и рассуждают на основе знаний.
- Обладают встроенной «моделью мира», моделью собственного «я» (сознанием) и инстинктами, позволяющими реализовывать их возможности без перепрограммирования.
- Коммуницируют с пользователем и другими системами на языке, близком естественному.
- Анализируют полученный результат.
- Обучаются из своего опыта и чужого, например, посредством чтения текстов, в диалоге с пользователем и на основе собственного опыта.
- Эволюционируют с течением времени, меняя внутреннюю организацию для повышения собственной эффективности.
- Постоянно активны и демонстрируют плохо предсказуемое, нестационарное и нелинейное поведение для решения сложных задач.

Для реализации подобной функциональности наиболее эффективными оказываются мультиагентные системы, основанные на мудьтиагентных технологиях.

Основные свойства МАС:

- Ведут обмен веществом, энергией и информацией с внешней средой.

- Динамически реконфигурируются (расширяются или сужаются) за счет внешних ресурсов в зависимости от ситуации в среде.
- Приобретают новые знания об окружающем мире.
- Реорганизуют внутренние связи на основе приходящих заказов (воздействий) или внутреннего анализа результатов (проактивность).
- Высвобождают запасенную энергию для полномасштабной перестройки внутренней структуры, ведущей к разрыву старых и установлению новых связей.
- Выход на пользователя или партнеров с встречными предложениями, продиктованными внутренними целями и задачами, предпочтениями и ограничениями.

В настоящее время МАС находят практическое применение в широком спектре предметных областей: от транспортной логистики и управления производством – до анализа текстов и извлечения знаний.

# СПИСОК ЛИТЕРАТУРЫ


1. Prigogine, I. The End of Certainty: Time, Chaos and the new Laws of Nature / I. Prigogine, I. Stengers. – Simon and Schuster, 1997. – 228 p.
2. Prigogine, I. Is Future Given? / I. Prigogine. – World Scientific, 2003. – 145 p.
3. Kaufman, S. At Home In the Universe: The Search for the Laws of Self-Organization and Complexity / S. Kaufman. – Oxford: Oxford University Press, 1995. – 336 p.
4. Holland, J. Hidden Order: How Adaptation Builds Complexity / J. Holland. – N.Y.: Addison Wesley, 1995. – 185 p.
5. Holland, J. Emergence: from Chaos to Order / J. Holland. – Oxford: Oxford University Press, 2000. – 274 p.
6. Investigating Current Social, Economic and Educational Issues using Framework and Tools of Complexity Science / G. Rzevski // Journal of the World University Forum. – 2008. – Volume 1. – Number 2. – P. 75-84.
7. Мультиагентные технологии в промышленных применениях: к 20-летию основания Самарской научной школы мультиагентных систем / П.О. Скобелев // Мехатроника, автоматизация, управление. – 2011. – № 12. – С. 33-46.
8. Интеллектуальные системы управления ресурсами в реальном времени: принципы разработки, опыт промышленных внедрений и перспективы развития / П.О. Скобелев // Приложение к теоретическому и прикладному научно-техническому журналу «Информационные технологии». – 2013. – №1. – С. 1–32.

9. Исследование моделей организации грузовых перевозок с применением мультиагентной системы для адаптивного планирования мобильных ресурсов в реальном времени / Н.О. Амелина, А.Н. Лада, И.В. Майоров [и др.] // Проблемы управления. – 2011. – № 6. – С. 31–37.
10. Wooldridge, M. An Introduction to Multi-Agent Systems / M. Wooldridge. – Wiley, 2002. – 340 p.
11. Открытые мультиагентные системы для оперативной обработки информации в процессах принятия решений / П.О. Скобелев // Автометрия. – 2002. – Т. 38. – № 6. – С. 45–61.
12. Мультиагентные модели взаимодействия для построения сетей потребностей и возможностей в открытых системах / В.А. Виттих, П.О. Скобелев // Автоматика и Телемеханика. – 2003. – № 1. – С. 162–169.
13. Метод сопряженных взаимодействий для управления распределением ресурсов в реальном масштабе времени / В.А. Виттих, П.О. Скобелев // Автометрия. – 2009. – Т.45. – № 2. – С. 78–87.
14. Agent Technology: Computing as Interaction. A Roadmap for Agent Based Computing // Agentlink: [сайт]. – 2022. – URL: http://www.agentlink.org/roadmap/index.html (дата обращения: 22.01.2022).
15. Bonabeau E., Theraulaz G. Swarm Smarts. What computers are learning from them? / E. Bonabeau, G. Theraulaz // Scientific American. – 2000. – Volume 282. – Number 3. – P. 54–61.
16. Гаврилова, Т.А. Базы знаний интеллектуальных систем: учебник для вузов / Т.А. Гаврилова, В.Ф. Хорошевский. – Санкт-Петербург: Питер, 2001. – 384 с.
17. Принятие решений на основе консенсуса с применением мультиагентных технологий / В.А. Виттих, Т.В. Моисева, П.О. Скобелев // Онтология проектирования. – 2013. – № 2 (8). – С. 20–25.

18. Ситуационное управление и мультиагентные технологии: коллективный поиск согласованных решений в диалоге / П.О. Скобелев // Онтология проектирования. – 2013. – №2(8). – С. 26–48.
19. Методы и средства построения интеллектуальных систем для решения сложных задач адаптивного управления ресурсами в реальном времени / С.П. Грачев, А.А. Жиляев, В.Б. Ларюхин [и др.] // Автоматика и телемеханика, 2021. – № 11. – С. 30–67.
20. The AgentLink RoadMap. European Coordination Action for Agent-based Computing // Agentlink: [сайт]. – 2022. – URL: http://www.agentlink.org (дата обращения: 22.01.2022).
21. Specifying Protocols for Multi-Agent Systems Interaction. The Foundation of Physical Intelligent Agents // FIPA: [сайт]. – 2022. – URL: http://www.fipa.org (дата обращения: 23.01.2022).
22. Java Agent DEvelopment Framework // JADE.Tilab: [сайт]. – 2022. – URL: http://jade.tilab.com/ (дата обращения: 23.01.2022).
23. JADE. A White Paper // JADE.Tilab: [сайт]. – 2022. – URL: http://jade.cselt.it/papers/2003/WhitePaperJADEEXP.pdf (дата обращения: 23.01.2022).
24. JACK Autonomous Software // AOS Group: [сайт]. – 2022. – URL: http://www.aosgrp.com/products/jack/ (дата обращения: 24.01.2022).
25. Living Systems Technology Suite // Wwhitestein: [сайт]. –2022. – URL: http://www.whitestein.com/products/ (дата обращения: 25.01.2022).

Учебное издание

*Симонова Елена Витальевна*

**МОДЕЛИРОВАНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ.**
**Часть I. Адаптивное управление сложными системами на основе мультиагентных технологий**

*Учебное пособие*

Редакционно-издательская обработка Л.Р. Дмитриенко
Компьютерная верстка Л.Р. Дмитриенко

Подписано в печать 28.06.2022. Формат 60x84 1/16.
Бумага офсетная. Печ. л. 12,75.
Тираж 25 экз. Заказ . Арт. – 20(Р1У)/2022.

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА»
(САМАРСКИЙ УНИВЕРСИТЕТ)
443086, Самара, Московское шоссе, 34.

---

Издательство Самарского университета.
443086, Самара, Московское шоссе, 34.