



**Дагестанский
государственный
институт народного
хозяйства**

Раджабов Карахан Якубович

Учебное пособие
по дисциплине «Многоагентные системы»
(Направление «Бизнес-информатика»)

Махачкала – 2011

ББК32.973.26

УДК 681.3.017

Автор: Раджабов Карахан Якубович, к.э.н., доцент кафедры «Информационные технологии» Дагестанского государственного института народного хозяйства.

Внутренний рецензент: Гереева Тату Рашидовна, кандидат экономических наук, заведующая кафедрой «Математические методы в экономике» Дагестанского государственного института народного хозяйства.

Внешний рецензент: Ризаев Максим Касимович, кандидат физико-математических наук, доцент кафедры "Прикладная математика" математического факультета Дагестанского государственного университета.

Учебное пособие разработано с учетом требований п.41 Типового положения об образовательном учреждении высшего профессионального образования (высшем учебном заведении) РФ, Утвержденного постановлением Правительства РФ от 14.02.2008 №71.

Раджабов К.Я. Учебное пособие по дисциплине «Многоагентные системы» для студентов направления подготовки «Бизнес - информатика» (профили «Архитектура предприятия» и «Электронный бизнес») – Махачкала: Изд-во «ДГИНХ» – 118 с.

Рекомендовано к утверждению и к изданию Учебно-методическим советом ДГИНХ Проректор по учебной работе ДГИНХ, председатель Учебно-методического совета, доктор экономических наук, профессор Казватова Н.Ю. 23 апреля 2011 г.	Одобрено Советом факультета «Прикладная информатика (в экономике)» Председатель Совета, к.э.н., доцент Раджабов К.Я. 16 апреля 2011 г.
Одобрено кафедрой «Информационные технологии», протокол № 7 от 21 марта 2011 г. зав. кафедрой к.ф-м.н., Галяев В.С.	

Печатается по решению Учебно-методического совета Дагестанского государственного института народного хозяйства.

Содержание

	стр.
Введение	5
Лекция 1. Введение в Многоагентные системы (МАС), основные понятия и определения, задание 1	7
Лекция 2. Свойства агентов и принятая терминология в теории многоагентных систем, задание 2	12
Лекция 3. Основы теории агентов 3.1. Классификация агентов. 3.2. Архитектуры агентов, языки описания и реализации агентов, задание 3.	19
Лекция 4. Коллективное поведение агентов в многоагентных системах 4.1. Модели коллективного поведения. 4.2. Конфликты в многоагентных системах. 4.3. Протоколы и языки координации, задание 4	30
Лекция 5. Архитектура многоагентных систем 5.1. Архитектура взаимодействия системы агентов. 5.1.1. Одноуровневая архитектура взаимодействия агентов. 5.1.2. Иерархическая архитектура взаимодействия агентов, задание 5.	39
Лекция 6. Архитектура агента 6.1. Общая классификация архитектур. 6.2. Архитектуры агентов, основанные на знаниях. 6.3. Архитектура на основе планирования (реактивная архитектура). 6.4. Многоуровневость. 6.5. Примеры архитектур агентов, задание 6	46

Лекция 7. Языки программирования агентов 7.1. Требования к языкам программирования агентов. 7.2. Классификация языков программирования агентов. 7.3. Сравнительная характеристика языков, задание 7	68
Лекция 8. Программные интеллектуальные агенты 8.1. Самоорганизации и кооперация в компании. 8.2. Процесс самоорганизации в мультиагентной системе 8.3. Архитектура и интерфейс мультиагентной системы. 8.4. Примеры применения многоагентных систем, задание 8	88
9. Задания для текущего контроля усвоения материала	111
10. Задания для итогового контроля остаточных знаний по дисциплине	112
11. Задания для самостоятельной работы студентов	113
Список рекомендуемой литературы	114

Введение

В ходе освоения студентами направления подготовки «Бизнес-информатика» учебная дисциплина «Многоагентные системы» занимает важное место в учебном процессе, поскольку способствует профессиональному освоению теории и практики исследований в области искусственного интеллекта. Данная дисциплина входит в состав профессионального цикла (региональный раздел) учебного плана подготовки бакалавров. Ее освоение способствует формированию у студентов компетенций, направленных на разработку и внедрение многоагентных систем на предприятиях (организациях), позволяющих проектировать и создавать эффективные программные комплексы, удовлетворяющие требованиям современной, динамично развивающейся экономики РФ и нашего региона, в частности.

Изучение данной дисциплины позволит студентам получить умения и практические навыки в области решения прикладных проблемно-ориентированных задач в области разработки интеллектуальных информационных систем, освоения специализированного программного обеспечения, специфичных языков программирования, например задач построения систем электронной коммерции и маркетинга, релевантных поисковых систем и задач распознавания текста, систем сбора разнородных данных и др.

Подготовленный лекционный материал в рамках данного учебного пособия соответствует образовательной программе по направлению 080500.62 «Бизнес - информатика», содержит в себе лекции, вопросы для закрепления материала, задания для самостоятельной работы. Вся тематика лекций последовательно изложена и взаимосвязана с разработанной ранее рабочей программой по этой дисциплине.

Учебное пособие предназначено для студентов 3 курса дневного и заочного отделений факультета информационных технологий направления

подготовки «Бизнес-информатика», профили «Архитектура предприятия» и «Электронный бизнес».

При составлении данного учебного пособия использованы литературные и URL-источники сети Интернет[1-22].

Лекция 1. Введение в проблематику, основные понятия и определения.

В области ИТ-технологий, и в частности, во множестве направлений искусственного интеллекта имеют место исследования, получившие название “Многоагентные системы”. Исследования по интеллектуальным агентам и многоагентным системам объединены в самостоятельный раздел искусственного интеллекта. Прикладной интерес к многоагентным системам обусловлен достижениями в области информационных технологий, искусственного интеллекта, распределенных информационных систем, компьютерных сетей и в компьютерной технике.

Многоагентные системы имеют реальную возможность интегрировать в себе самые передовые достижения перечисленных областей, демонстрируя принципиально новые качества.

Как отмечено в работе [13], первоначально идея интеллектуального посредника ("агента") возникла в связи с желанием упростить стиль общения конечного пользователя с компьютерными программами, поскольку в настоящее время имеет место стиль взаимодействия пользователя с компьютером, основанный на том, что пользователь запускает задачу явным образом и управляет ее решением.

Идея интеллектуального посредника возникла как попытка интеллектуализации пользовательского интерфейса, и развитие методов искусственного интеллекта позволило сделать новый шаг к изменению стиля взаимодействия пользователя с компьютером. Возникла идея создания так называемых "автономных агентов", которые породили уже новый стиль взаимодействия пользователя с программой. Вместо взаимодействия, инициируемого пользователем путем команд и прямых манипуляций, пользователь вовлекается в совместный процесс решения.

При этом, как пользователь, так и компьютерный посредник, оба принимают участие в запуске задачи, управлении событиями и решении задачи. Для такого стиля используется метафора "*персональный ассистент*" (ПА) [13,

15], который сотрудничает с пользователем в той же рабочей среде.

Главная особенность интерфейса, обеспечиваемого ПА, состоит в том, что этот интерфейс оказывается *персонафицированным*. Последнее достигается за счет того, что ПА наделяется способностью к обучению. В самом простом варианте, ПА получает информацию о привычках пользователя путем, как говорят, "наблюдая" за работой своего пользователя. Обучаясь интересам, привычкам и предпочтениям пользователя, а также окружающего его сообщества пользователей (это те, кто доступен персональному ассистенту через компьютерную сеть), ПА может стать весьма полезным, причем в различных аспектах: выполнять решение задач по поручению пользователя, тренировать его, управлять событиями и процедурами. Заметим, что по существу персонафикация пользовательского интерфейса - это новый резерв его интеллектуализации, который удачно дополняет "интеллектуальность интерфейса", которая традиционно ассоциируется только с экранными графическими средствами.

Исследования и экспериментальные программные разработки довольно быстро показали, что множество задач, в которых ПА с большой пользой может ассистировать пользователю, практически неограниченно: отбор информации, просмотр информации, поиск в Internet, управление электронной почтой, календарное планирование встреч, выбор книг, кино, музыки и т.д. Разработки в этой области поддерживались и поддерживаются такими известными фирмами, как Apple, Hewlett Packard, Digital, японскими фирмами. Термин "персональный ассистент" была заменен на "интеллектуальный посредник", или, как стали чаще говорить на русском языке - "*интеллектуальный агент*" (ИА).

В процессе эволюции данного понятия, приведенная выше идея вышла за рамки интеллектуального пользовательского интерфейса, она все более и более ориентировалась на идеи и методы искусственного интеллекта, на активное использование тех преимуществ, которые дают современные локальные и глобальные компьютерные сети, распределенные базы данных и распределенные вычисления.

Активное развитие методов и технологий распределенного искусственного интеллекта, достижения в области аппаратных и программных средств поддержки концепции распределенности и открытости привели к осознанию того важного факта, что агенты могут интегрироваться в системы, совместно решающие сложные задачи.

Это означало появление нового направления развития распределенных систем искусственного интеллекта и такие системы получили название ***много-агентных систем***.

Определение: Многоагентная система - это множество интеллектуальных агентов, распределенных по сети, мигрирующих по ней в поисках релевантных данных, знаний и процедур и кооперирующихся в процессе выработки решений [32].

В результате возникла новая концепция сообщества "*программных роботов*", цель которых - удовлетворение различных информационных и вычислительных потребностей конечных пользователей.

Структура исследований в области многоагентных систем в настоящее время очень широка и сравнима с широтой исследований в области искусственного интеллекта.

Исследования в области многоагентных систем можно разделить на основные направления:

- теория агентов*, в которой рассматриваются формализмы и математические методы для описания рассуждений об агентах и для выражения желаемых свойств агентов;

- методы кооперации агентов* (организации кооперативного поведения) в процессе совместного решения задач или при каких-либо других вариантах взаимодействия;

- архитектура агентов и многоагентных систем* (область исследований, в которой изучается, как построить компьютерную систему, которая удовлетворяет тем или иным свойствам, которые выражены средствами теории агентов);

-языки программирования агентов;
-методы, языки и средства коммуникации агентов;
-методы и программные средства поддержки мобильности агентов
(миграции агентов по сети).

Особое место среди этих направлений занимают исследования, связанные с разработкой *приложений многоагентных систем* и инструментальных средств поддержки технологии их разработки. Можно еще упомянуть проблемы, связанные с *аутентификацией* (авторизацией) агентов, обеспечением *безопасности* и ряд других.

В рамках изучения нашей дисциплины нами будут рассмотрены теория агентов -> методы кооперации агентов -> архитектура агентов и многоагентных систем -> языки программирования агентов.

Что касается *приложений многоагентных систем*, то, в предлагаемом учебном пособии будет приведена отдельная лекция по приложениям данного научного направления.

Задание 1.

Перечень контрольных вопросов по теме:

1. Что изучает дисциплина «Многоагентные системы», и как она связана с другими направлениями исследований в области искусственного интеллекта.
2. Ознакомьтесь с 8 определениями искусственного интеллекта, приведенными в книге «Стюарт Рассел, Питер Норvig. Искусственный интеллект. Современный подход (Второе издание)». (Табл.1.стр.35.)
3. Обсудите в группе тест Тьюринга и определите, какой компьютер мы можем считать интеллектуальным?
4. Что вы понимаете под законом правильного мышления? Приведите примеры.
5. Определитесь с понятиями агент, рациональный агент.

6. Какими преимуществами обладает подход к исследованию искусственного интеллекта как области проектирования рациональных агентов, дайте им характеристику.
7. Подготовьте реферат по теме «Предыстория искусственного интеллекта».
8. Приведите примеры практического применения теории рациональных агентов и области их приложений.
9. Выберите одно или несколько упражнений из главы 1 книги «Стюарт Рассел, Питер Норвиг. Искусственный интеллект. Современный подход (Второе издание)», стр.73-74; выполните его и предоставьте для ознакомления и оценки преподавателю (вынесите на обсуждение в учебной группе).
10. Охарактеризуйте современное состояние разработок в этой области (область экономики, ИТ-технологий, в частности, Интернет).

Лекция 2. Свойства агентов и принятая терминология в теории многоагентных систем

В настоящее время вопрос о том, какую компьютерную программу следует квалифицировать как агента и/или многоагентную систему, находится в стадии интенсивного обсуждения. Причина такого интереса к этому вопросу объясняется тем, что ученые в этой области опасаются, что термины “интеллектуальный агент” и “многоагентная система” станут расхожими терминами, как это и случилось с термином “интеллектуальная система”. Этот вопрос обсуждался, в частности, на нескольких семинарах **FIPA** (Federation of Intelligent Physical Agents) - международной организации, созданной и имеющей целью продвижение идей многоагентных систем в область практических приложений.

Определение агента [6, 28]:

“Агент - это сущность, которая находится в некоторой среде, от которой она получает данные и которые отражают события, происходящие в среде, интерпретирует их и исполняет команды, которые воздействуют на среду. Агент может содержать программные и аппаратные компоненты... Отсутствие четкого определения мира агентов и присутствие большого количества атрибутов, с ним связанных, а также существование большого разнообразия примеров агентов говорит о том, агенты это достаточно *общая технология*, которая аккумулирует в себе несколько различных областей”.

В среде исследователей в этой области принято различать два определения интеллектуального агента - “слабое” и “сильное” [6, 28]:

Под интеллектуальным агентом в *слабом смысле* понимается программно или аппаратно-реализованная система, которая обладает такими свойствами, как:

-*автономность* (способность ИА функционировать без вмешательства человека и при этом осуществлять самоконтроль над своими действиями и внутренним состоянием);

-*общественное поведение* (socialability) (способность функционировать в сообществе с другими агентами, обмениваясь с ними сообщениями с помощью некоторого общепонятного языка коммуникаций);

-*реактивность* (reactivity) (способность воспринимать состояние среды и своевременно реагировать на те изменения, которые в ней происходят);

-*про-активность* (pro-activity) (способность агента брать на себя инициативу, т.е. способность генерировать цели и действовать рационально для их достижения, а не только реагировать на внешние события).

Сильное определение агента подразумевает дополнительно к перечисленным выше свойствам дополнительных свойств – например, наличие у агента хотя бы некоторого подмножества так называемых “ментальных свойств”, называемых также *интенциональными понятиями*, к которым относятся следующие:

-*знания* (knowledge) - это постоянная часть знаний агента о себе, среде и других агентах, т.е. та часть, которая не изменяется в процессе его функционирования;

-*убеждения* (beliefs, вера) - знания агента о среде, в частности, о других агентах; это те знания, которые могут изменяться во времени и становиться неверными, однако агент может не иметь об этом информации и продолжать оставаться в убеждении, что на них можно основывать свои выводы;

-*желания* (desires) - это состояния, ситуации, достижение которых по разным причинам является для агента желательным, однако они могут быть противоречивыми и потому агент не ожидает, что все они будут достигнуты;

-*намерения* (intentions) - это то, что агент или обязан сделать в силу своих обязательств по отношению к другим агентам (ему “это” поручено и он взял эту задачу на себя), или то, что вытекает из его желаний (т.е. непротиворечивое подмножество желаний, выбранное по тем или иным причинам, и которое совместимо с принятыми на себя обязательствами);

-*цели* (goals) - конкретное множество конечных и промежуточных состояний, достижение которых агент принял в качестве текущей стратегии по-

ведения;

-*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 в рамках сотрудничества.

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

Некоторые авторы считают, что агент должен обладать также рядом других свойств. К ним относятся [6, 28]:

-*мобильность* (mobility) - способность агента мигрировать по сети в поисках необходимой информации для решения своих задач, при кооперативном решении задач совместно или с помощью других агентов и т.д.,

-*благосклонность* (benevolence) - готовность агентов помочь друг другу и готовность агента решать именно те задачи, которые ему поручает пользователь, что предполагает отсутствие у агента конфликтующих целей;

-*правдивость* (veracity) - свойство агента не манипулировать информацией, про которую ему заведомо известно, что она ложна;

-*рациональность* (rationality) - свойство агента действовать так, чтобы достигнуть своих целей, а не избегать их достижения, по крайней мере, в рамках своих знаний и убеждений.

Большинство исследователей в области теории и архитектур агентов считают обязательным включение в модель агента некоторого подмножества ментальных свойств, по крайней мере, таких, как знания, убеждения и цели.

Прикладные же разработки в этой области развиваются пока своим путем и только на уровне простых прототипов, делаются попытки реализовать идею агента с подмножеством ментальных свойств.

Рассмотрим главные направления развития МАС [33]. Сегодня основны-

ми направлениями в разработке МАС (см. схема 1.) являются *распределенный искусственный интеллект (РИИ)* и *искусственная жизнь* (в узком смысле этого термина). Ядро РИИ составляют исследования взаимодействия и кооперации небольшого числа интеллектуальных агентов, например, классических интеллектуальных систем, включающих базы знаний и решатели. Главной проблемой в РИИ является разработка интеллектуальных групп и организаций, способных решать задачи путем рассуждений, связанных с обработкой символов. Иными словами, здесь коллективное интеллектуальное поведение образуется на основе индивидуальных интеллектуальных поведений. Это предполагает согласование целей, интересов и стратегий различных агентов, координацию действий, разрешение конфликтов путем переговоров; теоретическую базу здесь составляют результаты, полученные в психологии малых групп и социологии организаций.

Важным разделом РИИ является кооперативное распределенное решение задач (КРРЗ), в которых речь идет о сети слабо связанных между собой решателей, которые совместно работают в целях решения задач, которые выходят за рамки индивидуальных возможностей. Различные узлы подобной сети, как правило, имеют неодинаковый опыт (знания, точки зрения) и разные ресурсы. Каждый узел должен быть способен модифицировать свое поведение в зависимости от обстоятельств, а также планировать свои стратегии коммуникации и кооперации с другими узлами. Здесь показателями уровня кооперации являются: характер распределения задач, объединение различных точек зрения и, конечно, возможность решения общей проблемы в заданное время.

Распределенное решение задач несколькими агентами разбивается на следующие этапы:

- 1) агент-менеджер (центральный орган) проводит декомпозицию исходной проблемы на отдельные задачи;
- 2) эти задачи распределяются между агентами-исполнителями;
- 3) каждый агент-исполнитель решает свою задачу, подчас также разделяя ее на подзадачи;
- 4) для получения общего результата производится композиция, интегра-

ция частных результатов, соответствующих выделенным задачам.

Второе направление – *искусственная жизнь* – в большей степени связано с трактовкой интеллектуального поведения в контексте выживания, адаптации и самоорганизации в динамичной, враждебной среде. В этом направлении глобальное интеллектуальное поведение всей системы рассматривается как результат локальных взаимодействий большого числа простых и необязательно интеллектуальных агентов. Здесь также используются термины «*коллективный интеллект*» (collective intelligence) или «*интеллект роя*» (swarm intelligence).

Сторонники этого направления опираются на следующие положения:

- 1) МАС есть популяция простых и зависимых друг от друга агентов;
- 2) каждый агент самостоятельно определяет свои реакции на события в локальной среде и взаимодействия с другими агентами;
- 3) связи между агентами являются горизонтальными, т.е. не существует агента-супервизора, управляющего взаимодействием других агентов;
- 4) нет точных правил, чтобы определить глобальное поведение агентов;
- 5) поведение, свойства и структура на коллективном уровне порождаются только локальными взаимодействиями агентов.

Здесь механизмы реакций на воздействия среды и локальных взаимодействий в общем случае не включают такие аспекты как прогнозирование, планирование, знания, но подчас позволяют решать сложные задачи. Типичными примерами такого коллективного интеллекта из биологии являются колонии муравьев, пчелиные ульи и т.п. Соответственно, здесь базовыми дисциплинами могут служить различные области биологической науки и, в первую очередь, эволюционная теория и генетика.

Часто проводятся принципиальные различия между *распределенными децентрализованным ИИ*. Идеология распределенного решения задач предполагает главным образом разделение знаний и ресурсов между агентами и, в меньшей степени, распределение управления и властных полномочий; как правило, здесь постулируется наличие единого органа управления, обеспечивающего принятие решений в критических (конфликтных) ситуациях. При этом исход-

ным объектом исследования является общая сложная проблема, для решения которой формируется группа агентов, строится общая концептуальная модель и вводятся глобальные критерии достижения цели.

В полностью децентрализованных системах управление происходит только за счет локальных взаимодействий между агентами. Здесь базовым объектом исследования оказывается уже не распределенное решение некоторой общей задачи, а деятельность автономного агента в динамическом многоагентном мире.

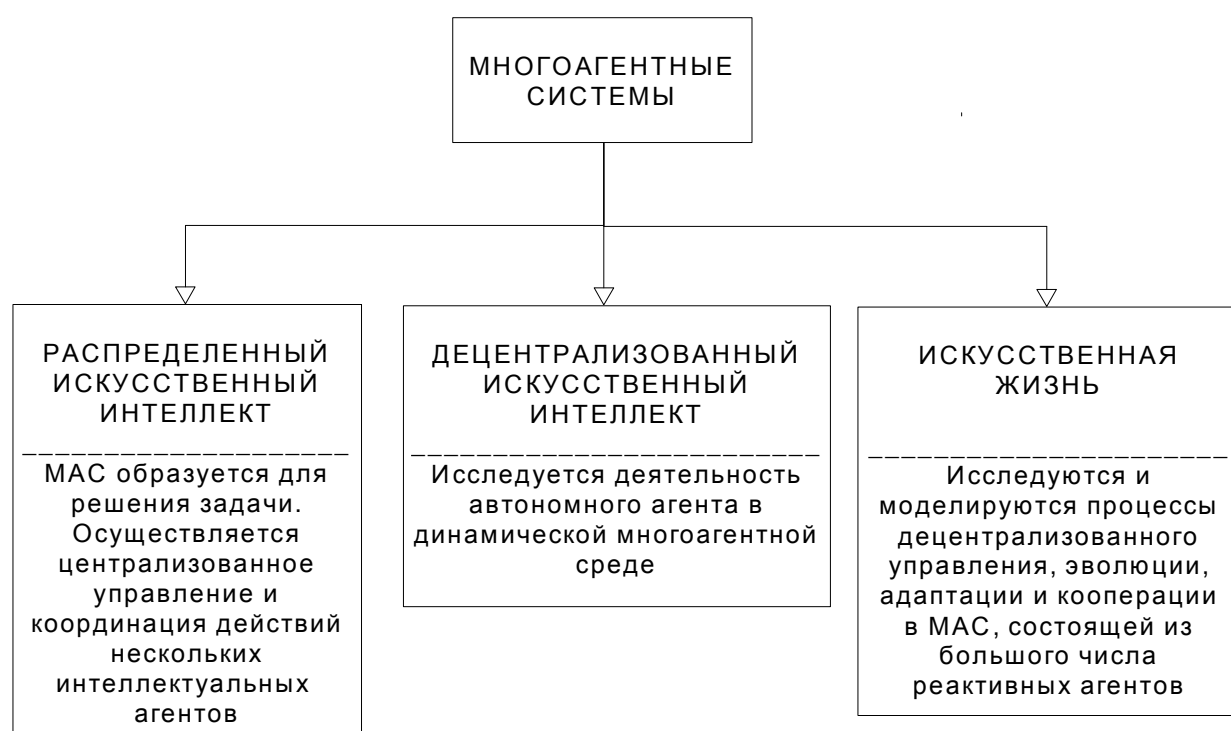


Схема. 1. Классификация многоагентных систем

Задание 2.

Перечень контрольных вопросов по теме:

1. Что из себя представляет - международная организация FIPA (Federation of Intelligent Physical Agents);
2. Проанализируйте в группе (организуя дискуссию) определение агента “Агент - это сущность, которая находится в некоторой среде, от которой она

получает данные и которые отражают события, происходящие в среде, интерпретирует их и исполняет команды, которые воздействуют на среду».

3. Что вы понимаете под интеллектуальным агентом в «слабом смысле»
4. Что вы понимаете под интеллектуальным агентом в «сильном смысле»
5. Какими свойствами должен обладать интеллектуальный агент в «слабом смысле», дайте им характеристику.
6. Какими дополнительными свойствами должен обладать интеллектуальный агент «в сильном смысле», охарактеризуйте их подробнее.
7. Подготовьте реферат по теме «Интеллектуальные агенты», см. книгу «Стюарт Рассел, Питер Норвиг. Искусственный интеллект. Современный подход (2 издание)», глава 2, стр.75-104.
8. Перечислите и охарактеризуйте главные направления развития много-агентных систем.
9. Выберите одно или несколько упражнений из главы 2 книги «Стюарт Рассел, Питер Норвиг. Искусственный интеллект. Современный подход (Второе издание)», стр.106-108; выполните его (их) и предоставьте для ознакомления и оценки преподавателю (вынесите на обсуждение в учебной группе).
10. Ознакомьтесь с резюме по 2 главе книги «Стюарт Рассел, Питер Норвиг. Искусственный интеллект. Современный подход (2 издание)» и попробуйте сформулировать ваши выводы в тезисной форме и представьте их преподавателю для ознакомления.

Лекция 3. Основы теории агентов

3.1. Классификация агентов.

3.2. Архитектуры агентов, языки описания и реализации агентов.

Применяемая в настоящее время теория агентов основана на формальных моделях ментальных понятий и правил манипулирования с ними. Дополнительно к этому введен раздел теории - представление динамических аспектов функционирования, как отдельного агента, так и сообщества агентов, при этом последняя задача выводит проблему за те пределы, которые исследуются традиционно в искусственном интеллекте.

Динамические аспекты функционирования «многоагентной системы» инновационны, и они образуют специальный раздел исследований, который называют теорией кооперативного (коллективного) поведения многоагентных систем [32, 33]. В УМК данный раздел теории агентов рассматривается в отдельной лекции.

Рассмотрим вопрос о моделях ментальных понятий. Первые попытки построить такие модели базировались на языках исчисления предикатов первого порядка, однако они не оказались удачными. Можно привести пример, из которого становится очевидным неадекватность исчисления предикатов первого порядка для формализации ментальных понятий [28]. Например, мы хотим представить формальную модель утверждения "Агент **А** убежден, что агент **Б** имеет информацию **X**":

$$\text{Bel} (\text{Агент_А}, \textit{Имеет} (\text{Агент_Б}, X))$$

Очевидно, что приведенная формула не является правильно построенной формулой исчисления предикатов первого порядка, поскольку вторым аргументом предиката $\text{Bel} (*,*)$ является, в свою очередь, предикат. Однако это чисто синтаксическая проблема, и она преодолевается, например, использованием мета-языка. Но есть потенциально и еще гораздо более серьезная семан-

тическая проблема. Например, предположим, что некий третий агент имеет информацию Y , имя которой является другим именем той же информации X . Тогда допустимое для предикатов первого порядка равенство ($X=Y$) приводит в нем к выводу:

$$\text{Bel} (\text{Агент_А}, \textit{Имеет} (\text{Агент_Б}, Y)).$$

что, по сути, совсем не то, что первое убеждение. Аналогичные, если не более сложные проблемы возникают при попытках формализации в рамках исчисления первого порядка и других ментальных понятий.

Очевидно, что отмеченные выше трудности объясняются тем, что ментальные понятия, вообще говоря, сами по себе являются, как отмечено в работе [28], относительно “темными”, “неясными”, “в которых стандартные правила подстановки исчисления предикатов первого порядка вообще не работают”.

При выборе формализмов для описания ментальных понятий, как следует из вышесказанного, нужно решать два класса проблем: синтаксическую проблему и семантическую, а любой формализм представления ментальных понятий (как и для представления любой другой информации) должен иметь два отдельных аспекта: свой язык формализации и свою семантическую модель.

Как отмечается в работе [28], среди известных к настоящему времени результатов относительно невелик выбор подходов к описанию синтаксиса и семантики. Для описания синтаксиса автор этой работы видит только два варианта:

- использование *мета-языков* (имеется в виду многосортная логика первого порядка с термами, которые обозначают формулы других языков, при этом ментальные понятия представляются предикатами мета-языка);

- использование расширений известных *модальных логик*, содержащих специальные модальные операторы (имеющие “не истинностные” значения),

применяемые к формулам.

Первый из вариантов представления синтаксиса достаточно традиционен и его использование является чисто технической проблемой. Второй вариант более интересен и, как представляется, обладает более мощными выразительными возможностями. Однако, ввиду динамического характера функционирования агента и многоагентной системы, эти логики должны дополняться средствами описания темпоральных (временных) аспектов ментальных понятий, а зачастую, и средствами описания свойств, связанных с реальным временем. Далее в качестве иллюстрации использования расширений модальных логик для формализации ментальных понятий будет приведен подход, предложенный в работах [9, 10].

С точки зрения семантического аспекта языка формализации ментальных понятий исследователями применяются такие варианты, как:

- семантика, представляемая множеством возможных миров.
- интерпретация символических структур с помощью поставленных им в соответствие функций (алгоритмов) и структур данных.

Оба подхода получили свое развитие и применение в области искусственного интеллекта.

В семантике множества возможных миров ментальные понятия интерпретируются множеством возможных миров и отношением достижимости (доступности) между ними. С каждым возможным миром ассоциируется некоторая теория (множество формул и атомарных предикатов – истинных фактов). Например [28], если один из игроков в покер получил какую-то карту, скажем, пиковую даму, то он может сделать некоторые выводы о допустимых и недопустимых раскладах карт у других игроков, и множество всех допустимых раскладов карт образует возможный мир игрока (агента) **A** для теории “Игрок (агент) **A** получил пиковую даму”. Другое дело, что построить такое множество раскладов и им руководствоваться в своей стратегии для игрока (агента) **A** нереально.

Представляется, что семантика возможных миров сложна для приме-

ния в практических задачах многоагентных систем, что видно уже из приведенного только что примера. Семантика множества возможных миров предполагает по умолчанию так называемое “логическое всеведение” (“logical omniscience”), т.е. предположение о том, что агенту на любом шаге выбора решения доступно все, что истинно в каждом возможном мире (может быть выведено из теории). Тем не менее, в ряде формализаций, одна из которых рассматривается ниже в данном разделе, используется именно такой подход к заданию семантики.

Альтернативные варианты задания семантики языка, как уже отмечалось, связаны с использованием интерпретации символических структур с помощью ассоциированных с символами или их цепочками алгоритмов и структур данных. Этот подход у прикладников пользуется большей популярностью и достаточно хорошо известен.

Дополнительно опишем в качестве примера подход к формализации ментальных понятий, предложенный в работах [9, 10], который базируется на синтаксисе предложенного авторами варианта многомодальной логики ветвящегося времени и на семантике в форме множества возможных миров.

Эти исследования интересны в том отношении, что, во-первых, рассматривают в рамках единой логики формализацию трех основных ментальных понятий агента: *убеждений, желаний и намерений*; во-вторых, формализуют динамику внешней среды и поведения агента средствами так называемой логики ветвящегося времени; в третьих, в этой работе вводится строго семантика на базе множества возможных миров и доказываются, что введенная логика полна. Наконец, в работах приводится конструктивная процедура тестирования формул на выполнимость и тождественную истинность на основе таблиц. В некотором смысле эта работа является наиболее продвинутой в области методов формального описания ментальных состояний агента и представляется весьма полезной как для понимания смысла ментальных понятий агента, так и для формализации и конструирования сложных многоагентных систем.

Авторы называют *убеждения* информационной компонентой состояния

агента, *желания* - мотивационным состоянием (компонентой), а *намерения* - его “рассудительной” (deliberative) компонентой. Приведем небольшой пример, построенный по аналогии с примером из работы [9], для того, чтобы пояснить с помощью семантики возможных миров содержание названных компонент ментального состояния агента.

Пусть некий политик Икс, который в настоящее время является Председателем оппозиционной партии, имеет желание перейти в Правительство. При этом может заручиться поддержкой партии или действовать самостоятельно. Партия может поддержать его действие (“Да”) или не поддержать (“Нет”), причем при удаче он должен будет покинуть место председателя партии, а при неудаче “товарищи по партии” могут не простить ему такую попытку, и тогда политик Икс должен будет покинуть политику вообще.

На рис.1 представлено дерево решений. В этом дереве имеются три вида вершин:

- детерминированные, в которых решения принимает сам претендент (они обозначены кружками);

- недетерминированные, в которых решения принимает “внешняя среда” (они обозначены квадратами);

- терминальные (обозначены кружками), около них проставлены числа, имеющие смысл “дохода” претендента от реализации того или иного варианта, отраженного последовательностью вершин-решений и дуг, их связывающих.

На рисунке показаны также вероятности и условные вероятности событий в вершинах, соответствующих недетерминированным действиям среды.

Предположим, что эту непростую задачу ввиду занятости партийными делами и для снятия с себя ответственности политик Икс поручает интеллектуальному агенту. Представим задачу в терминах ментальных понятий этого агента. Рассмотрим сначала возможные миры для убеждений агента. Будем преобразовывать дерево решений таким образом.

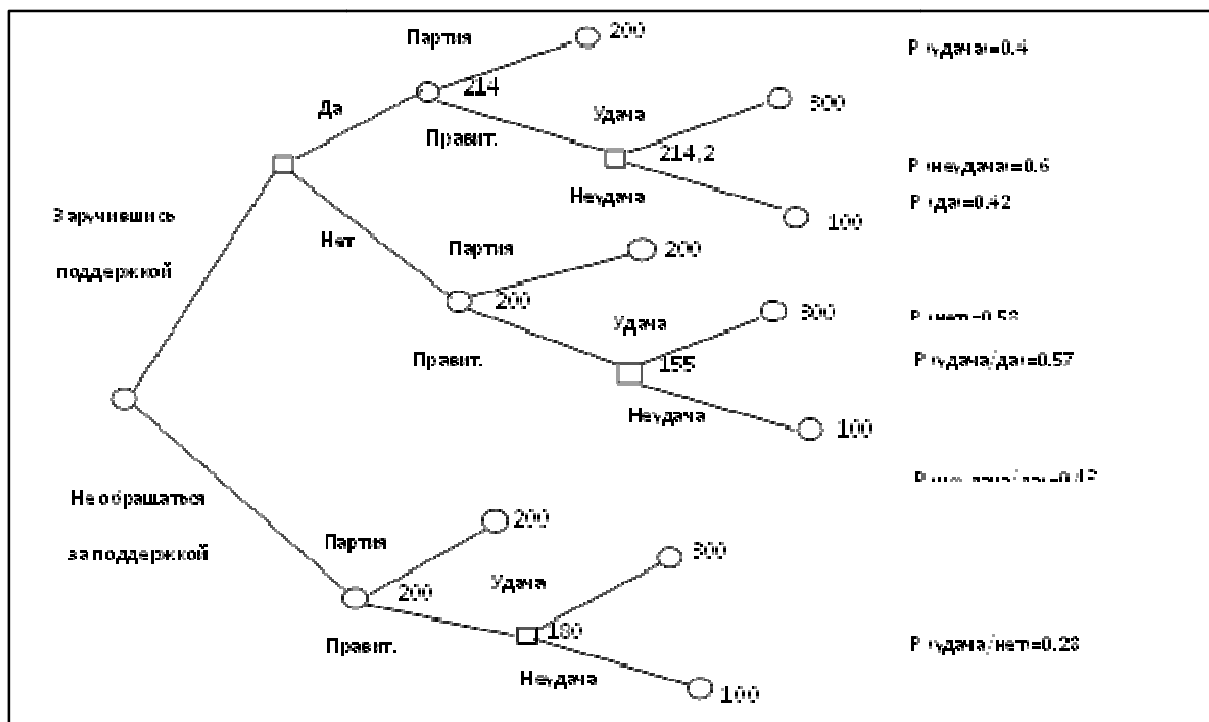


Рис.1. Обычное дерево решений

Начиная с корня дерева, будем фиксировать все пути в терминальные вершины, удаляя при этом вершины, отвечающие недетерминированным действиям среды (“склеивая дуги” входящую в такой узел и выходящую из него) и фиксируя вероятности соответствующих исходов как вероятности возможных миров. Эти возможные миры с сопоставленными им вероятностями представлены на рис.2.

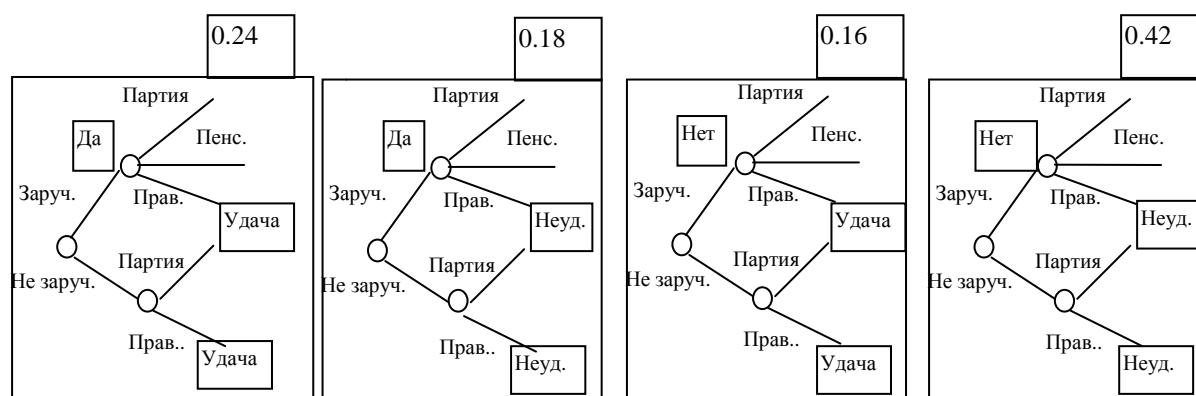


Рис.2. Возможные миры для убеждений

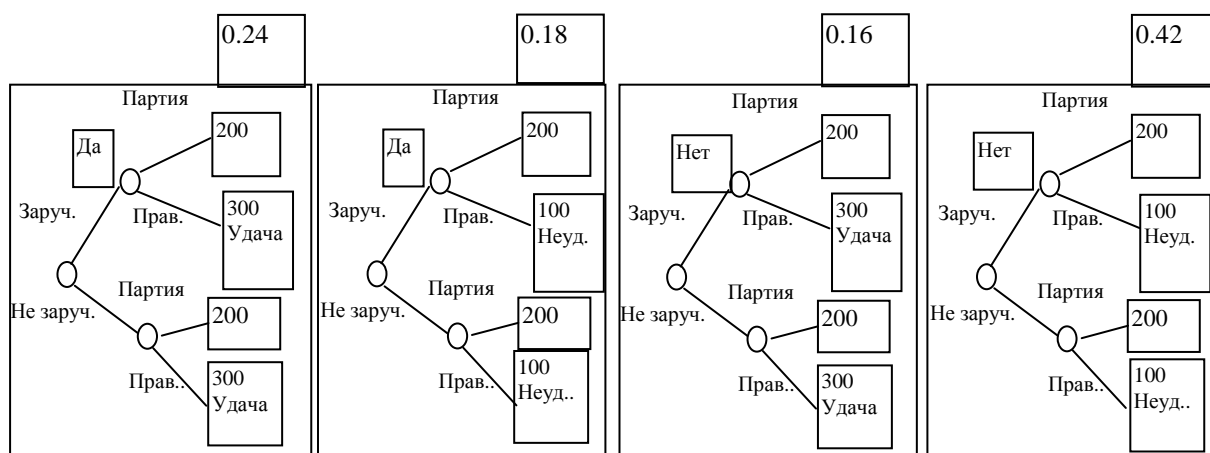


Рис.3. Возможные миры для желаний

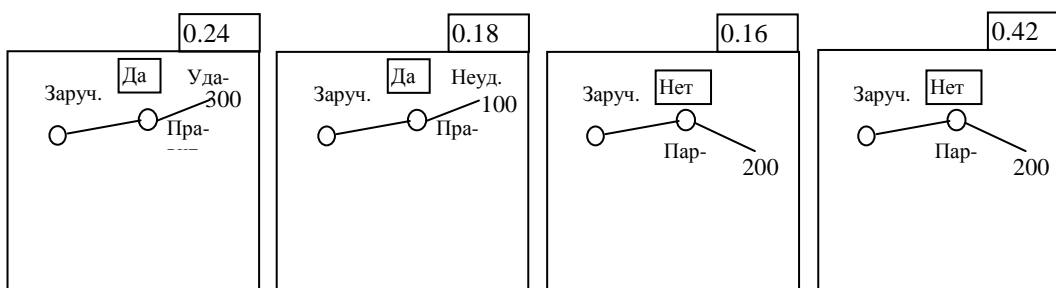


Рис.4. Возможные миры для намерений

Множество возможных (достижимых) миров для желаний отличается от таковых для убеждений тем, что, во-первых, отлучение от политики (соответствующий узел помечен на рис. 2 меткой “пенсия”) не является желаемым состоянием для политика Икс, а во-вторых, в этих мирах терминальным узлам ставится в соответствие еще значение функции выгоды. Множество возможных (достижимых) миров для желаний изображено на рис.3.

Что касается намерений, то в возможных мирах для желаний им отвечает тот стиль поведения, который приводит к оптимальным значениям функции выгоды. Эти миры иллюстрируются на рис.4.

Особенность поиска решения агентом, в отличие от дерева решений, состоит в том, что агенту неизвестны вероятности, приписываемые возможным мирам ментальных понятий и потери от выбора той или иной стратегии. Именно это является причиной и оправданием перехода от численной задачи к логической задаче.

Интересным является вопрос о формальных отношениях, в которых могут находиться ментальные понятия. Принято [9] различать два типа отношений между ними. Одно из них - это теоретико-множественное отношение порядка, а другое - структурное отношение порядка. Возможные варианты теоретико-множественных отношений для пары ментальных понятий - “убеждения - желания” приведены на рис.5 а, б, в, г:

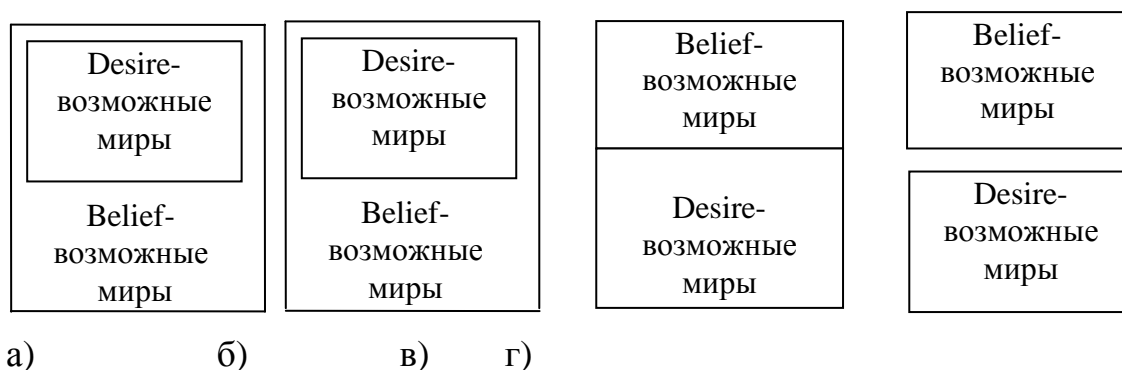


Рис.5. Теоретико-множественные отношения миров ментальных понятий

Рис.5а отвечает случаю, когда множество желаемых миров является подмножеством тех, в возможности которых агент убежден. Например, агент убежден, что он может “разбогатеть”, если будет играть в “Русское лото” и стать “бедным”, если вложит средства в разведение винного сорта кукурузы в условиях вечной мерзлоты. Очевидно, что второй путь для него не включается в множество его желаний. Вариант б) отвечает случаю, когда не все желаемые миры таковы, что агент убежден в их достижимости. Например, агент может желать разбогатеть, играя в “Русское лото” или победив на выборах в мэры г. Махачкалы. Однако, он не убежден, что второй вариант для него достижим. Аналогичные примеры можно привести и для двух других оставшихся вариантов - в) и г), хотя последний из них не имеет практического смысла. Очевидно, что такие же теоретико-множественные отношения могут быть заданы и для пар “желания - намерения” и “убеждения-намерения”.

Рассмотрим теперь структурные отношения порядка. Эти отношения определяются в связи с тем, что убеждения, желания и намерения могут представлять собой древовидные структуры, последовательность ветвей которых отвечает последовательности дискретных моментов времени, т.е. ментальные

понятия в возможном мире являются временными деревьями. Четыре варианта таких отношений приведены на рис. 6 (а, б, в и г) ([9]).

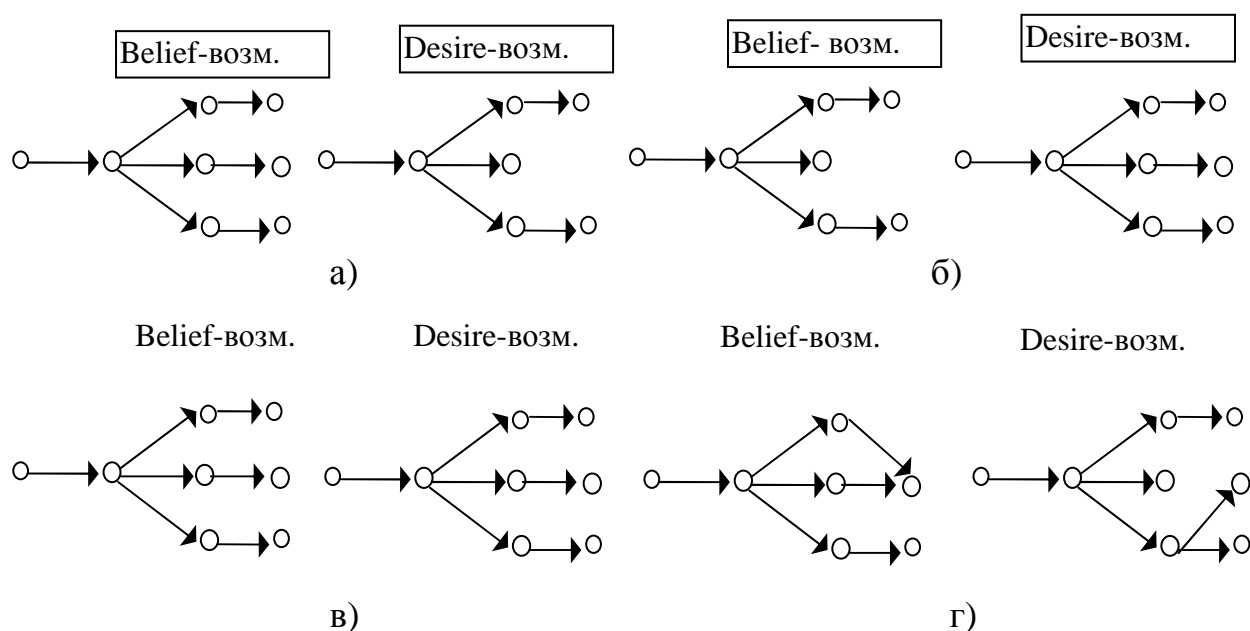


Рис.6. Структурные отношения на временных деревьях

На рис. 6а представлен вариант, когда Desire - возможный мир является подмиром возможного мира убеждений. Рис. 6б отвечает обратному отношению. В случае, представленном на рис.6в, эти миры совпадают, а на рис.6г миры несравнимы.

Все возможные варианты соотношений множества миров убеждений и миров желаний включают в себя различные комбинации, которые получаются при разных сочетаниях теоретико-множественного порядка и структурного порядка. Всего их девять, если не рассматривать варианты несравнимости. Обычно такие отношения рассматриваются для всех трех ментальных понятий. Различные варианты, которые при этом получаются, служат основой для классификации задач представления ментальных понятий [9].

В настоящее время теория (ментальных понятий) агентов, к сожалению, не выходит за пределы логической парадигмы. В этом смысле исследователи в данной области повторяют предпочтения, путь и ошибки, которые были характерны для исследователей в области искусственного интеллекта в 60-70х годах. Представляется, что обращаясь к таким сугубо антропоморфным поня-

тиям, как убеждения, желания, намерения и др., следовало бы учесть уже достаточно большой (положительный и отрицательный) опыт по формализации человеческих рассуждений, накопленный в области искусственного интеллекта. Теория агентов, к сожалению, пока полностью игнорирует достижения в области представления и обработки неполной, недоопределенной и нечеткой информации. Однако, рано или поздно, к таким вариантам формализации придется обратиться и в теории агентов, если не будет предложено нечто более эффективное и эффективное. Практики, разрабатывающие прикладные много-агентные системы уже начали использовать подходы на основе нечетких множеств, не дожидаясь продвижения в теории в эту сторону, и в одной из лекций будет описана одна из архитектур многоагентной системы - “архитектура для распределенных медицинских приложений”, которая уже активно “эксплуатирует” идею нечеткости.

Задание 3.

Перечень контрольных вопросов по теме:

1. На чем основана современная теория агентов?
2. Что мы понимаем под моделями ментальных понятий?
3. Охарактеризуйте исчисление предикатов первого порядка для формализации ментальных понятий.
4. Что из себя представляет синтаксис многомодальной логики ветвящегося времени и на семантике в форме возможных миров.
5. Охарактеризуйте основные ментальные понятия агентов (применительно к выше поставленному подходу) – убеждения, желания и намерения.
6. В каких формальных отношениях могут находиться ментальные понятия?
7. Что из себя представляют структурные отношения на временных деревьях?
8. В чем Вы видите недостатки современной теории агентов?

9. Охарактеризуйте подход, применяемый практиками в процессе разработки многоагентных систем, основанный на теории нечетких множеств.

10. Опишите архитектуру многоагентной системы для распределенных медицинских приложений.

Лекция 4. Коллективное поведение агентов в многоагентных системах

4.1. Модели коллективного поведения

4.2. Конфликты в многоагентных системах

4.3. Протоколы и языки координации

Как мы уже отмечали, идея «многоагентности» предполагает интеграцию агентов при коллективном решении задач, поскольку если в многоагентной системе агент не способен решить некоторую задачу самостоятельно, он может обратиться к другим агентам. Кооперация агентов также необходима для решения одной общей трудной задачи. При этом агенты могут строить планы действий, основываясь уже не только на своих возможностях, но и “думать” о планах и намерениях других агентов. Известно, что коллективы даже простейших автоматов, в которых каждый автомат преследует только свои примитивные цели, в целом способны решать очень сложные задачи [30]. Например, пчелиный улей или муравейник. Специалисты отмечают, что система, в которой агенты могут учитывать планы и интересы других агентов, будет являться во многих случаях еще более гибкой.

Использование идеи коллективного поведения приводит к проблемам - формирование совместных планов действий, возможность учета интересов компаньонов агента, синхронизация совместных действий, наличие конфликтующих целей, наличие конкуренции за совместные ресурсы, организацию переговоров о совместных действиях, распознавание необходимости кооперации, выбор подходящего партнера, обучение поведению в коллективе, декомпозиция задач и разделение обязанностей, правила поведения в коллективе, совместные обязательства и т. д.

Рассмотрим некоторые из вышеперечисленных проблем и обсудим подходы к их решению [32].

4.1. Модели коллективного поведения

В настоящее время предложено множество различных моделей коллективного поведения агентов. Как правило, каждая из моделей концентрирует

внимание на нескольких аспектах такого поведения и рассматривает проблемы в соответствии с выбранной архитектурой (моделью) самого агента. Для знаний, отвечающих за коллективное поведение, в архитектуре агента, обычно, выделяют специальный уровень - уровень кооперации (cooperationlayer).

Рассмотрим некоторые из ныне используемых подходов к формализации задач, решаемых на уровне кооперации агентов. Совместное поведение различных объектов изучается в рамках многих научных дисциплин. Выделим среди них те, которые представляются наиболее адекватными идее коллектива интеллектуальных агентов.

- *Распределенный искусственный интеллект* [27, 28]. Эта область искусственного интеллекта занимается самыми общими аспектами коллективного поведения агентов. Здесь основу составляют результаты, полученные в теории распределенных систем и теории принятия решений.

- *Теория игр* [12, 19, 29]. Аппарат теории игр часто используется для исследования коллективов интеллектуальных агентов. Многие ситуации, возникающие в многоагентной системе, находят подходящие аналоги в теории игр. Исследуются кооперативные игры, различные стратегии ведения торгов (переговоров), игры в размещения и др., которые являются аналогами ряда моделей коллективного поведения агентов.

- *Теория коллективного поведения автоматов* [30], которая исследует поведение больших коллективов автоматов с примитивными функциями. Поведение автомата может рассматриваться как недетерминированное, что позволяет строить различные вероятностные модели. Допускается обучение автомата при помощи штрафов и поощрений. Автомат может быть наделен памятью, в которой он в некоторой форме запоминает предыдущие штрафы и поощрения, и может использовать эту информацию для улучшения своего и коллективного поведения в соответствии с некоторой функцией дохода.

- *Биологические, экономические и социальные модели.*

В последние годы координацией агентов наиболее интенсивно занима-

ются в сообществе исследователей распределенного искусственного интеллекта. Значительное множество работ посвящено исследованию коллективного поведения агентов в процессе совместного решения задач в рамках Belief – Desire – Intention (BDI)- архитектур.

Далее рассмотрим различные модели кооперации агентов.

Модель кооперативного решения проблем (CPS) [27]. Эта модель рассматривает взаимодействие агентов, построенных согласно BDI-архитектуре. В модели ментальные понятия формализуются с помощью операторов временной логики. Используя формализм временной логики, в этих работах вводятся определения для таких понятий, как потенциал для кооперации, групповые действия, достижимость цели агентом и т. д. Остановимся на этих понятиях подробнее.

В процессе формирования кооперативного решения в рамках модели кооперативного решения проблем выделяют четыре этапа:

1. *Распознавание.* Процесс кооперативного решения начинается тогда, когда агент распознает целесообразность кооперативного действия. Например, у агента имеется цель, достичь которую в изоляции (по его убеждению) он не способен, или для ее достижения он предпочитает кооперацию.

2. *Формирование группы агентов.* На этой стадии агент, установивший возможность совместного действия, ищет партнеров. При успешном завершении этой стадии образуется группа агентов, имеющих совместные обязательства для коллективных действий.

3. *Формирование совместного плана.* Это та стадия, на которой агенты переговариваются с целью выработать совместный план, который по их убеждению приведет к желаемой цели.

4. *Совместные действия.* Здесь агенты действуют согласно выработанному плану, поддерживая взаимодействие согласно принятым на себя обязательствам.

Охарактеризуем каждый из перечисленных этапов.

Распознавание основывается на определении потенциала для кооперации агентов [27]:

По отношению к цели f агента i имеется потенциал для кооперации тогда и только тогда, когда (1) имеется некоторая группа g , такая, что i верит, что g может совместно достичь f , и, либо (2) i не может достичь f в изоляции, либо (3) i верит, что для каждого действия a , которое он мог бы выполнить для достижения цели f , он имеет иную цель, влекущую невыполнение действия a .

Формальные процедуры распознавания наличия потенциала для кооперации в работе [27] не приведены, но присутствует описание **формирования группы агентов**. Сама процедура образования группы заключается в том, что агент i (имеющий цель f), у которого имеется потенциал для кооперации с группой g , пытается реализовать в группе g состояние, в котором группа может совместно достичь цели f , и в котором группа g обязуется выполнять действия совместно в соответствии со своими обязательствами.

Формирование совместного плана начинается при условии, если предыдущая стадия была успешной. Тогда имеется группа агентов, обязующихся выполнять действия совместно.

Однако коллективные действия не могут начаться до тех пор, пока в группе не будет достигнуто соглашение, что конкретно будет делать каждый агент. Для выработки такого соглашения служит стадия формирования совместного плана. Переговоры являются механизмом выработки такого соглашения. Протокол переговоров есть распределенный алгоритм поиска соглашения. На стадии формирования совместного плана агенты группы осуществляют совместные попытки добиться такого состояния в группе, в котором все агенты выработали бы совместный план, согласны с ним, и намереваются действовать по нему.

Во время переговоров агенты предлагают планы, уточняют их с другими агентами, модифицируют предложенные планы и т. п. до тех пор, пока все агенты не согласятся с единым планом. Пример формирования совместного

решения будет приведен на следующей лекции.

По завершении предыдущей стадии начинается стадия **совместных действий**. В начальном состоянии стадии совместных действий в группе имеется общий план, и группа имеет намерение продолжать совместные действия. При нормальном ходе этого процесса действия выполняются согласно принятому плану вплоть до его завершения. Однако в некоторых ситуациях совместные действия могут прерываться. Например, в процессе совместных действий некий агент i может прийти к убеждению, что совместная цель f больше не является его целью. В этом случае его совместные обязательства диктуют ему условия, при которых он может отказаться от совместных обязательств, сообщить об этом группе и прекратить совместные действия, если это допустимо.

Подробное описание данной модели приведено в [26, 27].

4.2. Конфликты в многоагентных системах

Вероятность возникновения конфликтов в многоагентной среде является неизбежным следствием децентрализованности таких систем. Локальные убеждения одного агента могут, например, противоречить убеждениям других агентов. Агент может сформировать цель, которая будет конфликтовать с целями других агентов. При этом под конфликтом, обычно, понимают ситуацию, в которой возникает противоречие вида:

$$p \wedge q \Rightarrow false,$$

где p и q - убеждения агентов.

К числу возможных конфликтов в многоагентных системах относят конфликты:

- в системе убеждений агентов, которые могут возникать при получении агентом ложной информации от другого агента или информации, противоречащей убеждениям агента. Для поддержания целостности информации в многоагентной системе выделяют следующие уровни: терминологический, смысловой, временной.

- обусловленные неполнотой имеющейся у агента модели окружающего мира и моделей других агентов. Последние конфликты принято связывать с понятием рефлексии агента (рефлексия от позднелатинского *reflexio* — обращение назад, отражение). Рефлексия - форма теоретической деятельности человека, направленная на осмысление своих собственных действий и их законов; деятельность самопознания, раскрывающая специфику духовного мира человека.

- связанные с конкуренцией за совместные ресурсы или конфликты, связанные с наличием противоречивости целей.

Под разрешением конфликта понимается снятие логического противоречия вида $p \wedge q \Rightarrow false$ за счет отбрасывания одной из альтернатив в соответствии с некоторым критерием, или смены p и q вместе. Существует множество различных механизмов разрешения конфликтов, как например:

- разрешение конфликтов с использованием централизованного механизма (например, при наличии арбитра);
- разрешение конфликтов на основе правил поведения агентов. Например, наличие различных уровней компетентности агентов, при котором агент строит убеждение на основе информации, полученной из более компетентного источника.
- недетерминированный вариант разрешения конфликтов, когда используется подход на основе рандомизации, или жребия.

Приведем несколько примеров.

Пример 1. Механизм разрешения конфликтов, основанный на модели убеждений с приоритетами. При этом варианте агенты обмениваются информацией с целью достичь соглашения. Когда агент получает информацию, несовместимую с его локальными убеждениями, он либо отвергает ее, либо принимает, отбрасывая собственные убеждения. Имеется несколько типов метрик для определения силы убеждения, например, основанные на функции полезности [29], шкалированные величины. Механизм разрешения конфликтов, предложенный в [23], использует символьное шкалирование приоритетов

убеждений. Убеждениям ставятся в соответствие приоритеты трех уровней: ограничения (constraints), предпочтения (preferences) и гипотезы (options).

Для убеждения Q степень доверия BD есть одно из следующих значений: $BD(Q) \in \{N, P, O, U\}$, где N (от necessarily) означает, что Q - необходимо истинно, P (от preferably): Q - предпочтительное убеждение, O (от optionally): Q - возможное убеждение, $U:Q$ есть ложь. Над множеством $\{N, P, O, U\}$, определены [32] операции «Д» и «Е», в соответствии со следующими таблицами:

Д	N	P	O	U
N	N	P	O	U
P	P	P	O	U
O	O	O	O	U
U	U	U	U	U

Е	N	P	O	U
N	N	N	N	N
P	N	P	P	P
O	N	P	O	O
U	N	P	O	U

Тогда, например, $BD(P \wedge Q) = BD(P) \text{Е} BD(Q)$.

Пример 2. Другим возможным вариантом является механизм разрешения конфликтов с помощью введения *уровней компетентности* агентов [1], в соответствии с которым упорядочиваются их убеждения. Приведем определение уровня доверия к убеждению b , заимствованное из работы [1]:

Если:

- (1) $b1$ есть убеждение агента $a1$, имеющего цель в роли $r1$, такое, что $b1$ требуется для достижения этой цели, и
- (2) $b2$ есть убеждение агента $a2$, имеющего цель в роли $r2$, такое, что $b2$ необходимо для достижения этой цели, и
- (3) $b1$ конфликтует с $b2$,

Тогда:

$b1$ имеет больший уровень доверия, чем $b2$ тогда, и только тогда, когда, в соответствии с уровнем компетентности $(a2 \ r2) < (a1 \ r1)$, или, иными словами, агент $a1$, играющий роль $r1$, более компетентен, чем $a2$ в роли $r2$.

4.3. Протоколы и языки координации

Выше мы отмечали, что агенты обмениваются друг с другом информацией посредством переговоров. Протокол взаимодействия агентов как раз и определяет схему (распределенный алгоритм), по которой ведутся такие переговоры. В литературе предложено множество схем переговоров, рассмотрим некоторые из распространенных схем:

Схема 1. Теория речевых актов (SpeechActTheory, [24]), в которой переговоры строятся с использованием небольшого числа примитивов, например, *ASK, TELL, REJECT, REQUEST, COMMIT, NEGOTIATE*. Процесс переговоров начинается тогда, когда агент посылает сообщение, содержащее его точку зрения (позицию, attitude) по некоторому вопросу. Посредством обмена сообщениями *ASK, TELL, REJECT* агенты могут обсуждать некоторую тему и приходить к общему решению. Во время переговоров агенты обновляют свои базы знаний и, тем самым, повышают свои способности отвечать на новые запросы.

Схема 2. Протокол контрактных сетей (ContractNet Protocol,[21, 22]), предназначен для координации в системах распределенного решения проблем. Каждый узел контрактной сети способен выполнять определенные задачи. Если в процессе решения один узел (заказчик) не в состоянии решить некоторую задачу, он ищет другой подходящий узел, который способен ее решить. При этом он рассылает объявление потенциальным подрядчикам (contractors) о вакансии на выполнение некоторого договора. Если на это объявление отвечают несколько узлов, то заказчик, пользуясь некоторым критерием, выбирает наиболее подходящего подрядчика. Посредством торгов заказчик и подрядчик заключают договор (contract). Вариации этого протокола используются в так называемых “электронных предприятиях” (electronicenterprises).

Схема 3. COOL (COOrdination Language, [1, 2]), язык во многом схожий с языком KQML (описание которого приведено в Лекции 6), предназначенный для управления совместными действиями. Важнейшими конструкциями языка являются планы переговоров, определяющие состояния и правила переговоров

(conversationrules), и переговоры, определяющие текущее состояние плана, исполняемого в конкретный момент. Агенты могут иметь активными несколько переговоров и управлять их переключением и приостановкой, а также динамически создавать дочерние переговоры.

Схема 4. UNP (UnifiedNegotiation Protocol, [29]), в котором приводится механизм разрешения конфликтов, который позволяет повысить суммарную полезность, достигаемую агентами. Рассматриваются конфликтные ситуации, для разрешения которых используется “бросание монетки”, т.е. механизм рандомизации. При этом, в зависимости от того, кто выигрывает жребий, удовлетворяются цели либо обоих агентов, либо только одного.

Задание 4.

Перечень контрольных вопросов по теме:

1. Что Вы понимаете под идеей «многоагентности» при коллективном решении задач.
2. Что такое «кооперация агентов».
3. В чем суть идей «коллективного поведения», приведите примеры.
4. Какие модели коллективного поведения Вы знаете? Дайте им краткое описание.
5. В каких научных дисциплинах изучается совместное поведение различных объектов.
6. Охарактеризуйте «Модель кооперативного решения проблем (CPS)».
7. Что мы понимаем под конфликтами в многоагентных системах.
8. Что мы понимаем под разрешением конфликтов в многоагентных системах.
9. Приведите примеры разрешения конфликтов, основанные на различных моделях.
10. Что мы понимаем под протоколами и языками координации.

Лекция 5. Архитектура многоагентных систем

5.1. Архитектура взаимодействия системы агентов

5.1.1. Одноуровневая архитектура взаимодействия агентов

5.1.2. Иерархическая архитектура взаимодействия агентов

5.1. Архитектура взаимодействия системы агентов

С учетом, того, что теория агентов и многоагентных систем имеет дело с разработкой формализованных подходов для описания рассуждений об агентах и для выражения желаемых свойств агентов, научное направление исследований, называемое “архитектура агентов и многоагентных систем”, изучает вопрос о том, как построить компьютерную систему, которая удовлетворяет тем свойствам, которые выражены средствами теории агентов.

При выборе архитектуры многоагентной системы необходимо иметь в виду два ее аспекта:

- архитектуру, поддерживающую методы взаимодействия агентов в процессе функционирования системы в целом;
- архитектуру отдельного агента.

Бесспорно, существующие варианты архитектур многоагентных систем и рациональный выбор архитектуры отдельного агента и многоагентной системы в целом существенно зависят от того, какова концептуальная модель агента и принятый для ее описания формализм и язык спецификаций.

Помимо этого, важно выявить, какова математическая модель кооперации агентов при совместном функционировании в системе, на какое приложение или класс приложений ориентирована Многоагентная система, а также от ряда других факторов. Можно с уверенностью утверждать, что сколько существует и/или разрабатывается агентов и многоагентных систем, столько существует и архитектур. Тем не менее, можно выбрать некоторые характерные варианты, воплощающие в себе основные принципы конструирования архитектур и которые рассматриваются в сообществе специалистов как перспективные.

Дальнейшее описание возможных архитектур строится как их классификация с краткой ее иллюстрацией на примерах.

Первоначально рассмотрим *Архитектуру взаимодействия системы агентов*, назначение которой состоит в обеспечении скоординированного поведения агентов при решении общей и/или своих частных задач.

Здесь можно выделить два основных варианта архитектур:

1 вариант. Агенты не образуют иерархии и решают общую задачу полностью в распределенном варианте.

2 вариант. Координация распределенного функционирования агентов в той или иной мере поддерживается специально выделенным агентом, который при этом относится кмета - уровню по отношению к остальным агентам.

Существуют и более сложные иерархически организованные схемы взаимодействия агентов, однако мы ограничимся кратким рассмотрением названных двух вариантов.

5.1.1. Одноуровневая архитектура взаимодействия агентов

Ярким примером одноуровневой (полностью децентрализованной) архитектуры является архитектура системы для планирования совещаний (встреч) [3]. Данная архитектура относится к тем задачам, которые имеют дело с динамическим составлением расписаний выполнения некоторого вида деятельности в условиях ограниченных ресурсов, например, задачи составления расписания обслуживания судов в морском порту [31], задачи диспетчерского обслуживания движения самолетов в аэропорту [20], планирование работ в гибких автоматизированных производствах [17] и др.

Особенность задачи планирования встреч состоит в том, что расписание всегда составляется в контексте уже существующих назначений каждого из участников планируемой встречи. Использование же этой информации неким централизованным образом в форме базы данных исключается, поскольку, как правило, эта информация носит личный характер и участники предпочитают ее не раскрывать. Они предпочитают также не раскрывать информацию о своих личных предпочтениях по поводу той или иной встречи. Таким образом,

информация, которую нужно использовать в процессе планирования встреч, носит существенно распределенный характер и лишь ограниченно доступна каждому из участников. Наличие какой-либо хотя бы ограниченной централизации здесь исключается. Вообще говоря, подобная ситуация имеет место во многих распределенных задачах.

В такой ситуации каждый участник предстоящей встречи представляется в процессе планирования своим агентом (“электронным секретарем”), который знает все о своем клиенте и совсем немного о других участниках встречи. Стратегия поиска решения в этом случае предполагает использование переговоров для поиска глобально согласованного решения. При этом, хотя агенты остаются равноправными участниками переговоров, каждому из них по специальному алгоритму назначается определенная роль, причем роли агентов могут динамически меняться. Поясним это кратко на примере [3].

1. Начальный вызов. Агент - инициатор встречи устанавливает контакт с агентами заданных участников потенциальной встречи для выяснения их интереса к ней, при этом он указывает параметры встречи: дату, время, длительность, тему. Каждый секретарь спрашивает своего пользователя о том, проявляет ли он интерес к встрече. Если пользователь отвечает положительно, то его секретарь отвечает инициатору встречи о заинтересованности и указывает при этом “фактор ограничений”, который рассчитывается как сумма весов уже запланированных встреч на тот период, который предлагает инициатор встречи. Веса формируются таким образом:

вес=1, если запланированная встреча может быть передвинута;

вес=2, если такая встреча передвинуты быть не может.

Если агент отказывается от встречи, то он просто удаляется из списка участников и далее не рассматривается в задаче планирования.

2. Схема переговоров. Самый простой случай - когда все согласившиеся встретиться агенты передали суммарный вес - фактор ограничений, равный 0. Это означает, что они принимают время, предложенное инициатором, и переговоры на этом завершаются.

В противном случае агент-организатор вычисляет роли участников в предстоящих переговорах.

Агент, который имеет наибольший фактор ограничений (он имеет меньше всех свободного времени для планируемой встречи и сокращает в наибольшей мере пространство поиска компромисса) получает статус MC (MostConstrained) и далее он берет на себя роль координатора переговоров. Фактически этому агенту назначается *роль лидера* в традиционной терминологии теории распределенных вычислений. Остальные агенты получают статусы MC2, MC3,... в порядке убывания их факторов ограниченности, последний из них - LC (LeastConstrained).

Далее процесс переговоров выполняется в соответствии с некоторым алгоритмом (“протоколом”), при этом агенты ищут компромисс за счет сдвигов или отмен уже запланированных встреч с возможностью отказа от встречи, которая является темой переговоров. Роли агентов в зависимости от текущих результатов переговоров могут меняться, но алгоритм действий, отвечающий каждой из ролей, остается фиксированным. Переговоры продолжаются до того момента, когда будет найдено либо согласованное решение, либо встреча будет организована в ограниченном составе, либо она будет вообще отменена.

Описанный вариант взаимодействия агентов в одноуровневой архитектуре типичен в том смысле, что могут быть различные алгоритмы ведения переговоров, но основная суть, в частности, выражаемая назначением *ролей агентов*, в основном, сохраняется.

5.1.2. Иерархическая архитектура взаимодействия агентов

Рассмотрим простейший вариант иерархической организации взаимодействия агентов, который предполагает использование одного агента “мета-уровня”, осуществляющего координацию распределенного решения задач(и) агентами.

Агент, осуществляющий координацию, может быть привязан к конкретному серверу, и тогда он называется “*местом встречи агентов*”. Место встречи агентов (AMP - AgentMeetingPlace) - это агент, играющий роль брокера

ра между агентами, запрашивающими некоторые ресурсы, которыми обладают другие агенты, и теми агентами, которые эти ресурсы могут предоставить. Архитектура АМР есть архитектура обычного агента (рассматривается в следующей лекции), дополненная некоторыми вспомогательными компонентами, наличие которых обусловлено ролью этого агента как координатора взаимодействия других агентов. Эти вспомогательные компоненты должны, с одной стороны, содержать унифицированное описание множества доступных через АМР агентов и их возможностей (ресурсов, функций и пр.) и, с другой, организовать унифицированный доступ к ним. Это обеспечивается такими компонентами АМР [4].

1. Объекты базовых сервисов, в частности, это могут быть удаленный вызов объектов, упорядочение объектов, дублирование объектов и другие базовые возможности, которые обычно поддерживаются той или иной платформой открытой распределенной обработки, например, OMG/CORBA.

2. Связные порты, ответственные за прием и отправку агентов в АМР с помощью соответствующих протоколов.

3. Компонента установления подлинности агента по имени (опознание агента, “авторизация”).

4. Консьерж, выполняющий функции контроля полномочий поступающего агента, наличия на АМР запрашиваемого сервиса, оказания помощи агенту в выборе дальнейшего маршрута перемещения и др.

5. Поверхностный маршрутизатор, который выполняет функции интерфейса между агентами и компонентами АМР, которые сами по себе регистрируются в этом маршрутизаторе; он поддерживает ограниченный словарь для удовлетворения запросов агентов.

6. Лингвистический журнал, который представляет собой базу данных, помогающую агентам и АМР понимать друг друга в процессе коммуникаций. В нем регистрируются словари и языки, но не описания языков или смысл терминов, а лишь ссылки на них, т.е. журнал предоставляет информацию о том, что может быть понято в АМР.

7. Глубинный маршрутизатор, который ассистирует поверхностному при более специальных и сложных запросах.

8. Менеджер ресурсов; он регистрирует агентов на АМР и ассоциированные с ними ресурсы, а также управляет ресурсами АМР.

9. Среда исполнения агента, которая регистрируется в АМР и управляет доступом к компонентам агента; она интерпретирует сценарии, обеспечивает доступ к базовым возможностям и др.

10. Система доставки событий; источниками событий могут быть локальные средства, резидентные агенты АМР и др.; система регистрирует события и выполняет поиск агентов для соответствующего типа событий, сообщений.

В остальном архитектура координирующего агента аналогична архитектуре обычного агента, варианты которой рассматриваются в следующей лекции.

Задание 5.

Перечень контрольных вопросов по теме 5:

1. Что Вы понимаете под термином “архитектура агентов и много-агентных систем”.
2. Как построить компьютерную систему, которая удовлетворяет тем свойствам, которые выражены средствами теории агентов.
3. Какие аспекты необходимо иметь в виду при разработке (выборе) архитектуры многоагентной системы.
4. Что такое «архитектура, поддерживающая методы взаимодействия агентов в процессе функционирования системы в целом».
5. Что такое «архитектура отдельного агента».
6. Охарактеризуйте архитектуру взаимодействия системы агентов, приведите примеры.
7. Проведите сравнительный анализ двух вариантов архитектур

взаимодействия систем агентов, приведенных по данной теме.

8. Дайте полную характеристику одноуровневой архитектуры взаимодействия агентов.

9. Дайте полную характеристику иерархической архитектуры взаимодействия агентов.

10. Самостоятельно подберите литературу с описанием примеров одноуровневой и иерархической архитектуры взаимодействия агентов.

Лекция 6. Архитектура агента

6.1. Общая классификация архитектур

6.2. Архитектуры агентов, основанные на знаниях

6.3. Архитектура на основе планирования (реактивная архитектура)

6.4. Многоуровневость

6.5. Примеры архитектур агентов

6.1. Общая классификация архитектур

Грубая классификация архитектур агентов основывается на парадигме, лежащей в основе принятой архитектуры. По этому признаку различают два основных класса архитектур [25]:

- архитектура, которая базируется на принципах и методах искусственного интеллекта, т.е. систем основанных на знаниях (*deliberative agent architecture*”, “архитектура разумного агента”);
- архитектура, основанная на поведении (*reactive architecture*) или “реактивная архитектура” (основанная на реакции системы на события внешнего мира).

К настоящему времени среди разработанных архитектур не существует таких, которые можно четко классифицировать как поведенческие или основанные только на знаниях. Любая из разработанных архитектур является по существу гибридной, имея те или иные черты от архитектур обоих типов.

С другой стороны, независимо от лежащей в основе формализации парадигмы, архитектуры агентов классифицируются в зависимости от вида структуры, наложенной на функциональные компоненты агента и принятых методов организации взаимодействия его компонент в процессе работы.

На практике, архитектура агента организуется в виде нескольких уровней, и среди многоуровневых архитектур различают горизонтальную организацию взаимодействия уровней и вертикальную организацию.

Естественно, существуют и другие признаки классификации архитектур агентов, однако мы будем придерживаться отмеченных ввиду того, что они более широко распространены. Дадим характеристику архитектур, основанных

на знаниях и поведенческие архитектуры:

6.2. Архитектуры агентов, основанные на знаниях

Определение [28]: «Архитектура на основе знаний есть такая архитектура, которая содержит символьную модель мира, представленную в явной форме, и в которой принятие решений о действиях, которые должны быть предприняты агентом, осуществляется на основе рассуждений логического или псевдо-логического типов. Такой агент может рассматриваться как специальный случай системы, основанной на знаниях».

Сначала идея агента, основанного на знаниях, строилась на чисто логической основе и представлялась весьма перспективной. Однако позднее было обнаружено, что лежащее в основе такого подхода исчисление предикатов первого порядка неразрешимо. Более того, такие ментальные свойства агента, как убеждения, желания, намерения, обязательства по отношению к другим агентам и т.д. (Лекция 2), нельзя выразить в терминах исчисления предикатов первого порядка. Как итог, в ходе исследований были разработаны специальные варианты расширений модальных логик и подобных модальным (Лекция 3), которые оказались с точки зрения реализуемости более удачными. Такие архитектуры были названы Belief-Desire-Intention (BDI) - архитектурами.

Заметим, что идея архитектуры агента на основе знаний в настоящее время уже вышла за пределы логической парадигмы представления и обработки знаний. Имеются архитектуры, исповедующие лингвистический подход (на основе формальных грамматик), а также такие, которые пытаются использовать приближенные знания и правдоподобные рассуждения.

6.3. Архитектура на основе планирования (реактивная архитектура)

Архитектура на основе планирования (“планирующий агент”) рассматривается как альтернатива подходу, рассмотренному в предыдущем подразделе лекции. Данный подход развивался внутри сообщества специалистов по искусственному интеллекту еще с начала 1970-х годов, однако той его частью, которая занималась планированием поведения роботов и тому подобными задачами.

В этом подходе планирование рассматривалось как “конструирование последовательности действий, которая, будучи исполненной, приводила бы в результате к достижению желаемой цели” [28]. Простым примером архитектуры подобного рода является архитектура, в которой реакция агента на внешние события генерируется конечным автоматом. В качестве другого примера системы с архитектурой рассматриваемого типа может рассматриваться и широко известная система STRIPS [8], в которой использовался чисто логический подход совместно с предусловиями и постусловиями, ассоциированными с каждым из действий. В соответствии с принятой стратегией STRIPS, имея описание мира и желаемой цели, пытается найти последовательность действий, которая в итоге приведет к достижению цели с удовлетворением постусловий. К сожалению, система оказалась крайне неэффективной.

6.4. Многоуровневость

Только самые простые приложения агентов могут быть реализованы по одноуровневой схеме. Как правило, функциональные модули агента структурируются в несколько уровней, однако по различным принципам. Как правило, уровни представляют различные функциональности, такие, как восприятие внешних событий и простые реакции на них; поведение, управляемое целями; координация поведения с другими агентами; обновление внутреннего состояния агента, т.е. убеждений о внешнем мире; прогнозирование состояний внешнего мира; определение своих действий на очередном шаге и др.

Наиболее часто в архитектуре агента присутствуют уровни, ответственные за:

- восприятие и исполнение действий;
- реактивное поведение;
- локальное планирование;
- кооперативное поведение;
- моделирование;
- формирование намерений и обучение агента.

Существует два основных класса многоуровневых архитектур в зависи-

мости от того, как организуется взаимодействие уровней:

- горизонтально организованная архитектура;
- вертикально организованная архитектура.

В горизонтально организованной архитектуре, все уровни агента имеют доступ к уровню восприятия и действий (в общем случае - все уровни могут общаться между собой в стиле “бродкастинга”). Вариант такой архитектуры приведен на рис.7а.

В вертикально организованной архитектуре только один из уровней имеет доступ к уровню восприятия и действий, а каждый из остальных уровней общается только с парой непосредственно смежных с ним уровней. Примеры таких архитектур приведены на рис.7.б и 7.в.

Примерами горизонтально организованной архитектуры являются рассматриваемые далее архитектуры TouringMachine [7] и Will-architecture [16] (D.Moffat and N.H.Frijda). Хорошим примером вертикально организованной архитектуры является InteRRaP - архитектура, подробно рассматриваемая ниже [18].

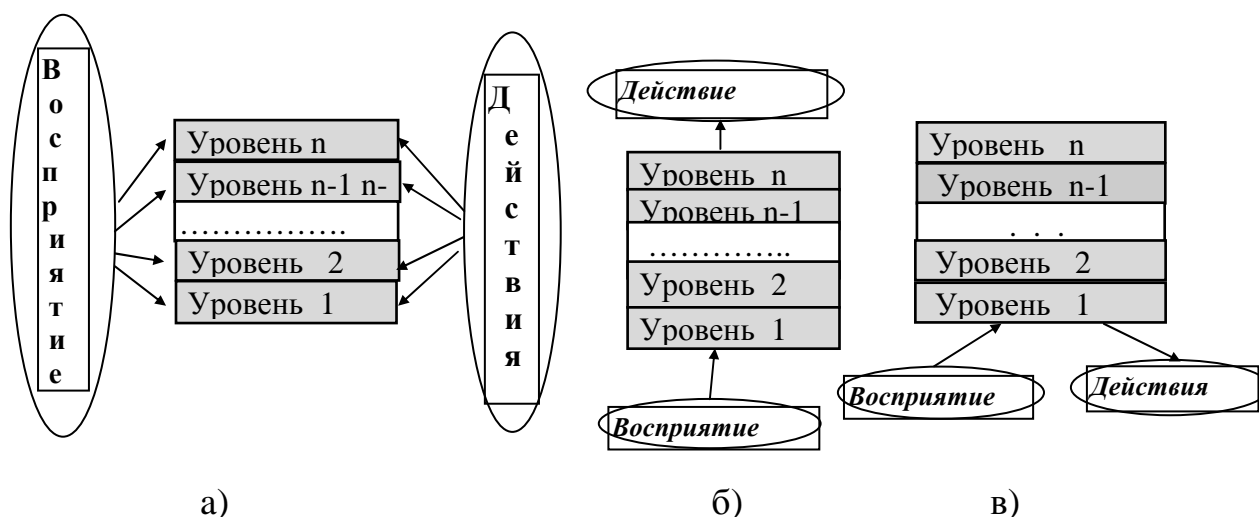


Рис.7 Организация взаимодействия уровней в многоуровневой архитектуре агента

Основные проблемы реализации горизонтально организованной архитектуры обусловлены сложностью организации согласованной работы всех

уровней.

Так, в архитектуре **TouringMachine** эта проблема решается с помощью специального алгоритма, который подавляет входы некоторых уровней, если соответствующая информация не имеет к ним отношения, и осуществляет «цензурирование» выходов. Это выполняется с помощью специального набора правил.

В архитектуре **Will**, задача управления согласованной работой уровней выполняется с помощью введения специальных функций совместимости входных событий с “интересами” (concerns) уровней. Здесь делается попытка ввести некоторую самоорганизацию, однако из имеющихся работ не вполне понятно, как это может быть реализовано в различных приложениях.

В отличие от горизонтально организованной архитектуры, в вертикально организованной архитектуре проблема управления взаимодействием уровней не является столь сложной, поскольку выходная информация каждого из уровней всегда имеет адресат.

В известных вертикально организованных архитектурах распределение функциональных модулей по уровням выполняется по одному из двух принципов. Согласно одному из них, различные уровни отвечают различному уровню абстракции, в основном, одного и того же набора функциональностей (такой принцип используется в **InteRRap** - архитектуре).

Согласно другому принципу каждый уровень отвечает некоторой функциональности или их набору. По такому принципу построена **МЕССА** - архитектура [18], в которой цикл функционирования агента состоит из четырех фаз: активация цели, планирование, конкретизация плана в набор действий и исполнение. В соответствии с этими фазами архитектура агента состоит из четырех уровней.

Недостатком вертикально организованной архитектуры считается то ее свойство, что оказывается перегруженным уровень исполнения (действий), Например, в **IntRRaP** - архитектуре нижний уровень должен реагировать на непредвиденные события, отслеживать исполнение команд, полученных с

уровня локального планирования, следить за выполнением ограничений, наложенных контекстом локального планирования (временных, ресурсных) и, наконец, он должен функционировать в соответствии с дополнительными кооперативными обязанностями (обязательствами), которые возложены на агента другими агентами многоагентной системы.

6.5. Примеры архитектур агентов

Далее при рассмотрении примеров архитектур многоагентных систем даются их авторские названия в соответствии с источниками на английском языке и в русском переводе.

Композиционная архитектура многоагентной системы

Описана в работе [5], название - DESIRE (англ. "framework for Design and Specification of Interacting Reasoning components"), базируется на понятии композиционной архитектуры, которая позволяет "описывать сложного агента в прозрачной манере, а также интегрировать рассуждения и действия в единой (декларативной) логической среде".

Авторы данной архитектуры предполагают, что агент в процессе работы выполняет действия следующего типа:

- активно воспринимает и фильтрует информацию из внешнего мира;
- строит заключения по этой информации;
- инициализирует и выполняет коммуникации с другими агентами в интересах кооперации;
- генерирует и обновляет свои убеждения, делая и отклоняя дополнительные предположения;
- изменяет внешний мир, воздействуя на него.

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

- (а) знания о материальном мире;
- (б) знания о ментальном мире самого агента;
- (в) знания о ментальных мирах других агентов;
- (г) знания о взаимодействии с материальным миром

(д) знания о коммуникациях с другими агентами (какие коммуникации возможны и полезны для получения дополнительной информации).

Другим важным моментом данной архитектуры является необходимость принимать во внимание динамику знаний и ее неполноту. В рамках принятой модели различают часть структуры знаний, зависящую от времени (“динамическое состояние информации, или базу фактов”) и ее инвариантную часть, которая не изменяется во всех состояниях.

Главная идея композиционной архитектуры состоит в том, чтобы можно было любого сложного агента создать как композицию компонент - примитивов, каждая из которых описывает одну из подзадач, которая должна им выполняться. Компоненты должны соединяться друг с другом в соответствии с предопределенной семантикой связи. 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 система в целом может композироваться из отдельных агентов. Компоненты описываются в терминах многосортной логики предикатов.

Пример агента с использованием композиционной архитектуры приведен на рис.8. Агент, структура которого представлена на этом рисунке, имеет имя **А**.

Он состоит из трех главных компонент:

- его собственное ментальное состояние, которое включает в себя убеждения агента, знания агента о себе (что он знает и чего он не знает), знания о стратегиях управления и т.д., компоненту, генерирующую предположения, позволяющие восполнять неполноту знаний, и управляющую часть;

- компоненту коммуникации, которая связывает агента **А** с внешним материальным миром и другими агентами (например, с агентом **В**); эта компонента обеспечивает связь с внешним миром путем генерации наблюдений и генерации действий и то же самое по отношению к другим агентам (ставит вопросы и получает ответы);

-компоненту анализа состояния мира, которая содержит предметные знания о материальном мире.

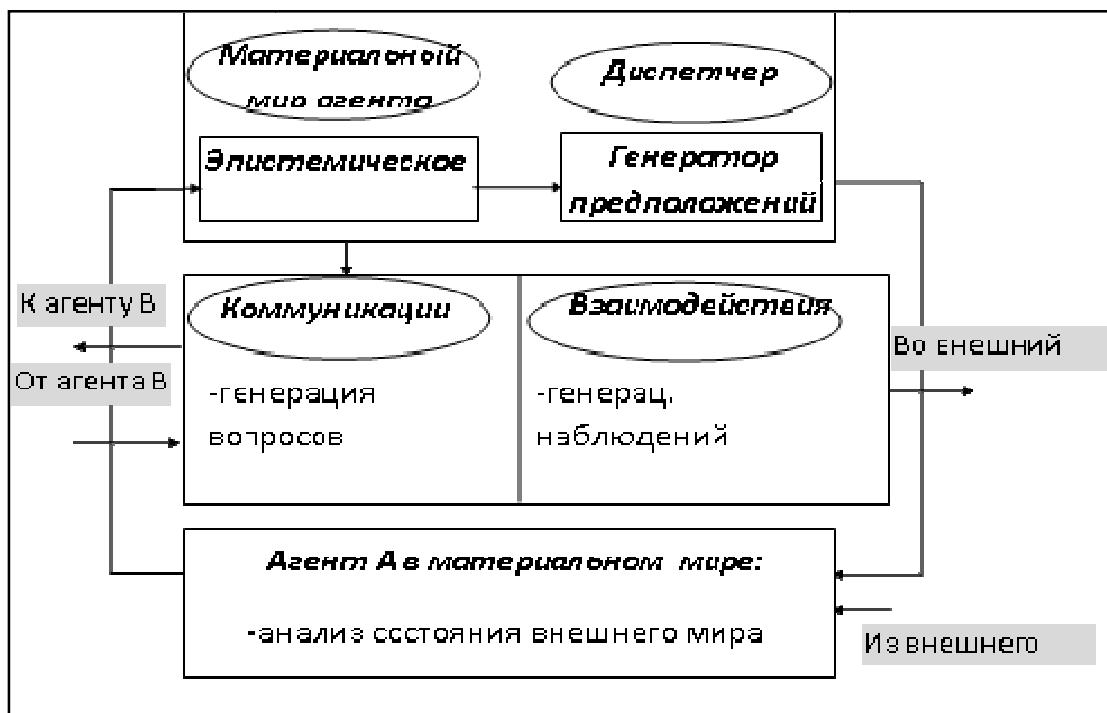


Рис.8. Агент в композиционной архитектуре

Можно видеть, что эта архитектура не структурирована по уровням и компоненты соответствуют заложенным функциям.

Достоинства данной 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.

Приведенная архитектура не реализована в рамках какого-либо приложения, авторы только намереваются использовать ее для диагностики электрических сетей. Данная архитектура не кажется перспективной уже хотя бы ввиду ее одноуровневой структуризации. Формализация задачи обладает

весьма ограниченными возможностями, т.к. в рамках чисто предикатной логики невыразимо большинство свойств агента.

Многоуровневая архитектура для автономного агента (“TouringMachine”)

Эта архитектура разработана для специального приложения автономного агента-подвижного робота [7]. В отличие от других, она рассчитана на реальное приложение. Разработчики данной архитектуры предполагают, что:

- В реальном приложении агент имеет дело с непредвиденными событиями внешнего мира, как в пространстве, так и во времени и в присутствии других агентов.
- При этом он должен сохранять способность адекватно реагировать на них и принимать решения.
- Предлагаемая архитектура агента и является, как правило, гибридной.
- Агент должен иметь архитектуру, которая позволит ему справляться с неопределенностью и неполнотой информации, реагировать на непредвиденные события, пользуясь относительно простыми правилами.

Данная архитектура представлена на рис.9. [7], и демонстрирует хорошее поведение в соответствии с контекстом - состоянием внешней среды.

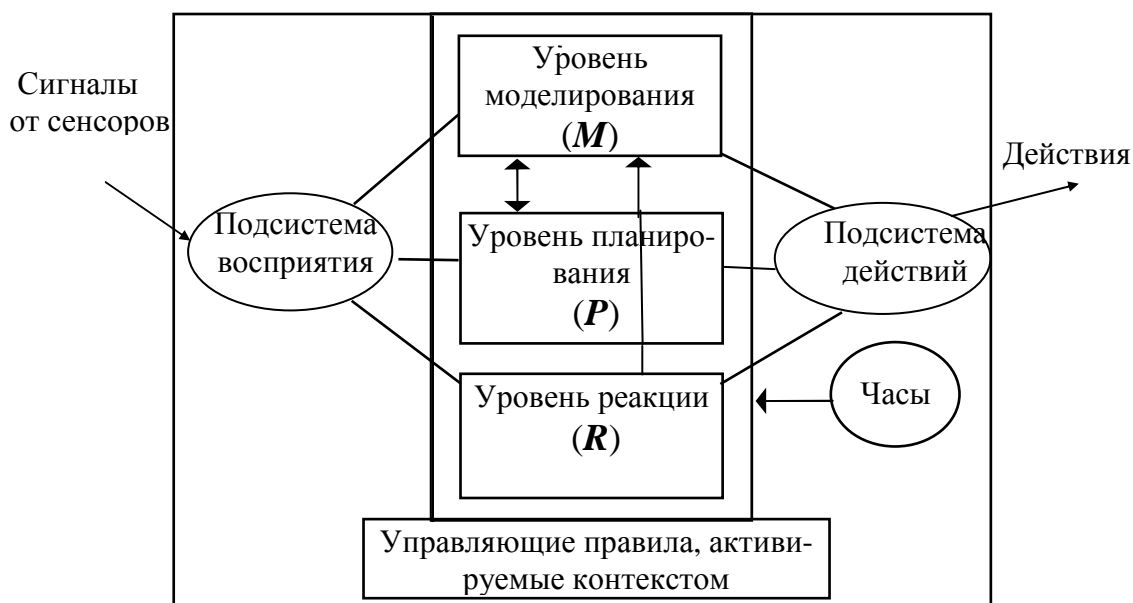


Рис.9. Многоуровневая архитектура (“TouringMachine”)

Она включает в себя три уровня, каждый из которых соответствует различ-

ным типам способностей агента.

-уровень *реакции* на события **R** поддерживает способность агента быстро реагировать на события, выдаваемые вышележащим уровнем, даже если они ранее не планировались;

-уровень *планирования* **P** генерирует, исполняет и динамически реконструирует частичные планы, например, для выбора маршрута подвижного робота;

-уровень *предсказания* или *моделирования* **M** моделирует поведение сущностей внешней среды и самого агента, что может использоваться для объяснения наблюдаемого поведения и предсказания возможного их поведения в будущем.

Каждый из этих уровней имеет модель мира агента на соответствующем уровне абстракции и содержит возможности, соответствующие уровню. Каждый из уровней напрямую связан с компонентой восприятия и действия и в состоянии независимо от других уровней решать, реагировать или не реагировать на текущее состояние мира.

В архитектуру включена *Подсистема управления* на основе правил, активируемая контекстом с задачей обеспечить подходящее поведение агента в случае конфликта вариантов поведения, инициируемого различными уровнями. Система реализована как комбинация технологии обмена сообщениями и контекстной активации управляющих правил (в соответствии со спецификой предметной области), выступающей в роли посредника, который исследует данные разных уровней (воспринимаемый вход и выходы разных уровней), вводит на различные уровни новые данные и удаляет некоторые данные.

Синхронизация входов и выходов уровней также обеспечивается этой подсистемой. Фактически правила подсистемы выступают в роли фильтра между сенсорами агента и внутренними уровнями агента (“supressors”) и между уровнями и их исполнительными элементами (“sensors”). Посредничество это остается активным все время работы агента, однако оно “прозрачно” для уровней, каждый из которых продолжает действовать так, как если бы он был

единственным при управлении агентом, не заботясь о возможном конфликте.

Данная архитектура имеет реализацию и работоспособна. Она интегрирует в себе ряд традиционных механизмов рассуждений на основе знаний и механизмов чисто поведенческого, “реактивного” характера, и является весьма характерным представителем горизонтально организованной многоуровневой архитектуры.

Многоуровневая архитектура для распределенных приложений

Эта архитектура [11] была разработана специально для системы здравоохранения, включает в себя многоуровневую структуру знаний, рабочую память, менеджера коммуникаций и человеко-машинный интерфейс (см. рис.10.).

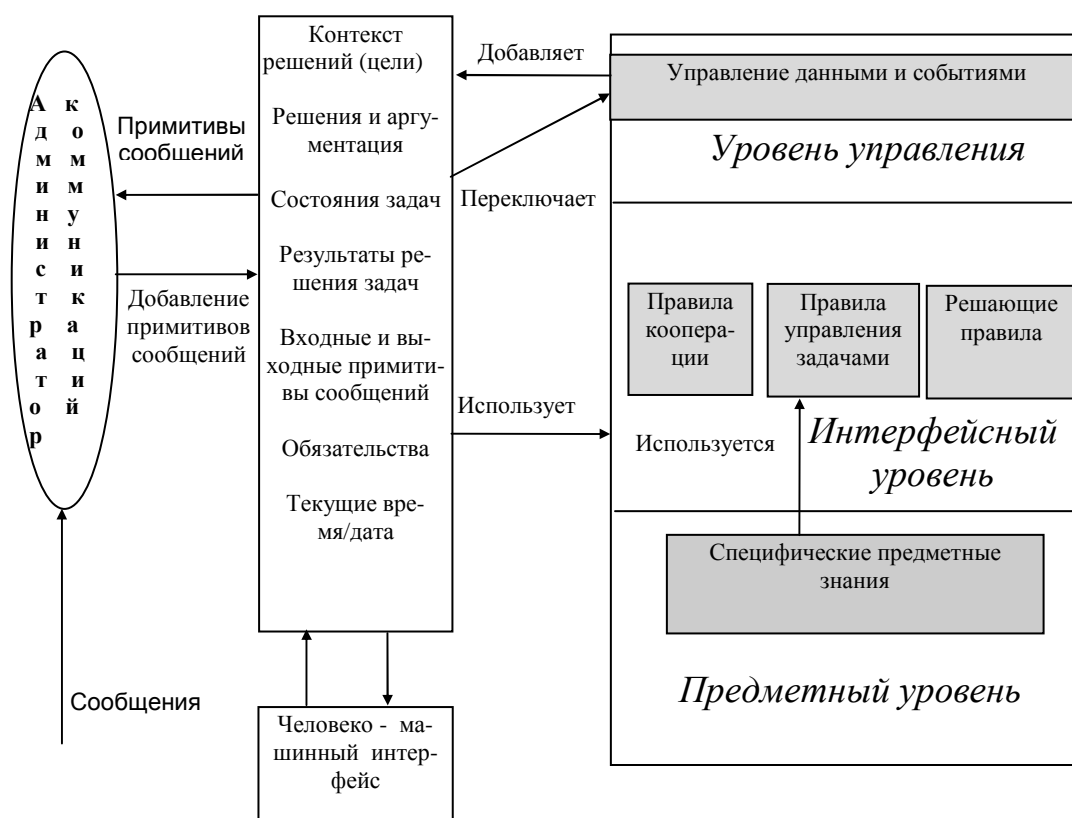


Рис.10. Архитектура для распределенных медицинских приложений

Поскольку данная архитектура должна быть релевантной медицинским приложениям, агент должен обладать обоими типами поведения - как поведением на основе знаний (например, для выбора планов, декомпозиции задач, размещения задач), так и поведением на основе быстрой реакции на события

(например, для формирования ответов в реальном времени на поступающие новые данные, изменение имеющихся данных, на изменение текущих соглашений с другими агентами). Таким образом, эта архитектура, как и все ранее рассмотренные, является гибридной.

В этой архитектуре интеллектуальное поведение поддерживается совместной работой таких компонент, как блок решающих правил для вычисления плана, блок правил для управления задачами, их декомпозицией и размещением, а также блок правил для поддержки соглашений с другими агентами при кооперативном решении задач. Реактивное поведение реализуется с помощью управляющего уровня, который реагирует на изменение состояния рабочей памяти (например, при поступлении новых результатов решения задачи, целей или сообщений, а также при изменении имеющихся данных, целей, межагентских соглашений или состояний задач). Ключевым моментом данной архитектуры является трехуровневая организация знаний, при этом выделяются следующие уровни:

1. *Уровень специфических предметных знаний*, в котором содержатся медицинские знания о болезнях, знания о планах управления лечением болезней (“протоколы”), база данных о пациентах (истории болезней) и база данных о доступных ресурсах. Однако предметные знания не содержат какой-либо информации о том, как их следует использовать, здесь представлены только свойства предметной области.
2. *Уровень знаний о процедурах вывода*; он содержит декларативные правила вывода, которые должны применяться к предметным знаниям о конкретном пациенте, чтобы вывести новые данные. Этот уровень - основной в архитектуре. В свою очередь он подразделяется на компоненты принятия решений в условиях неопределенности, управления задачами и управления кооперацией агентов. Например, модуль управления задачами содержит декларативную схему вывода для управления переходами состояний задачи. Особенности системы вывода решений состоят в том, что она не использует понятия ментального состояния агента (убеждения,

желания, намерения) и не использует какой-либо логический язык для вывода, для этого она использует стратегии аргументации в условиях неопределенности. Это означает, что эта архитектура не является BDI-архитектурой.

3. *Менеджер задач* ответственен за декомпозицию задач на подзадачи и их распределение по соответствующим агентам, а также за управления переходами состояний задач. Управление кооперацией агентов использует механизм, основанный на взаимных обязательствах агентов (“любой агент согласен предпринимать схему действий, которая имеет целью исполнить задачу за подходящее время”), и соглашениях о том, при каких условиях агент вправе отказаться от своих обязательств и как он должен себя вести по отношению к другим агентам, когда такие обстоятельства возникнут.
4. *Уровень управляющих знаний*, который применяет знания о процессе вывода к предметным знаниям, чтобы генерировать схему вывода, если в рабочую память добавляются новые знания.

Приведенное в модели функциональное разделение знаний на предметные знания, знания о процедурах вывода и управляющие знания существенно упрощает их представление, повторное использование и эксплуатацию, поскольку эти компоненты могут создаваться и поддерживаться независимо. Кроме того, эта архитектура позволяет просто встраивать программы извлечения знаний, каждая из компонент которых может получаться и модифицироваться независимо друг от друга.

Другие три компоненты рассматриваемой архитектуры - это *рабочая память*, *менеджер коммуникаций* и *человеко-машинный интерфейс*.

Рабочая память служит для запоминания текущих данных, генерируемых уровнем управления, пользователя и менеджера коммуникаций. Типы информации, которая хранится в рабочей памяти, таковы: цели, которые должны быть достигнуты; состояния задач, которые находятся в текущем состоянии процесса выполнения соглашений с другими агентами. Фактически, в

привычной нам терминологии, рабочая память есть ни что иное, как доска объявлений.

Менеджер коммуникаций содержит в себе сообщения, которые должны быть посланы другим агентам, представленные на языке коммуникаций с примитивами типа примитивов языка KQML: *обратиться с просьбой, принять, отвергнуть, изменить, предложить, проинформировать, запросить данные, отказаться и подтвердить.*

Человеко-машинный интерфейс определяет схему взаимодействия между системой и пользователем, поскольку данная многоагентная система не является автономной, что связано с личной ответственностью пользователя за здоровье пациента.

Эта архитектура основана на знаниях, имеет горизонтальную схему взаимодействия уровней. Главная ее особенность в том, что она достаточно сильно ориентирована на приложение.

IDS-архитектура

Эта архитектура возникла [14] в результате комбинирования двух направлений исследований. Первое из них - это логика рассуждений о действиях и изменениях с исходным понятием "населенной (живыми существами) динамической системы" ("InhabitedDynamicSystem"-IDS). Второе направление - это построение эффективной реализации интеллектуальной системы.

Архитектура имеет трехуровневую структуру и является гибридной. Полагается, что IDS - система размещается в некотором мире (среде) и состоит из двух базовых частей - "Мыслящей части" ("Я", "Ego") и "Машины" ("Подвижной части объекта", "тела", "vehicle"). Автор интерпретирует понятие "Мыслящая часть" как интеллектуальную, основанную на знаниях часть автономного агента, его "мозг", в то время как "машина" - это тело агента, т.е. его бессознательная часть, которая в порядке реакции на восприятие и приказы на исполнение что-то делает. IDS воспринимает внешнюю среду. Используя процесс восприятия, она редуцирует и существенно обобщает воспринимаемую информацию, и посылает выход в "Мыслящую часть". В свою оче-

редь, “Мыслящая часть” посылает команды на свою подвижную часть, которая их отрабатывает без какого-либо дополнительного управления или изменения, вызывая соответствующие изменения во внешнем мире (см. Рис.11).

Эта идея реализуется в виде трехуровневой архитектуры, представленной на рис.12. 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.

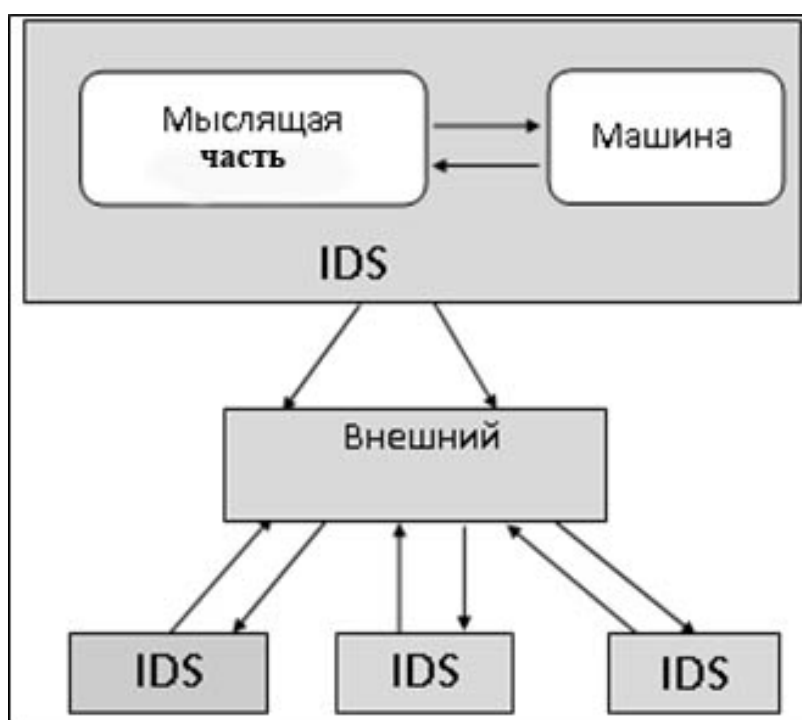


Рис.11. IDS – архитектура

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

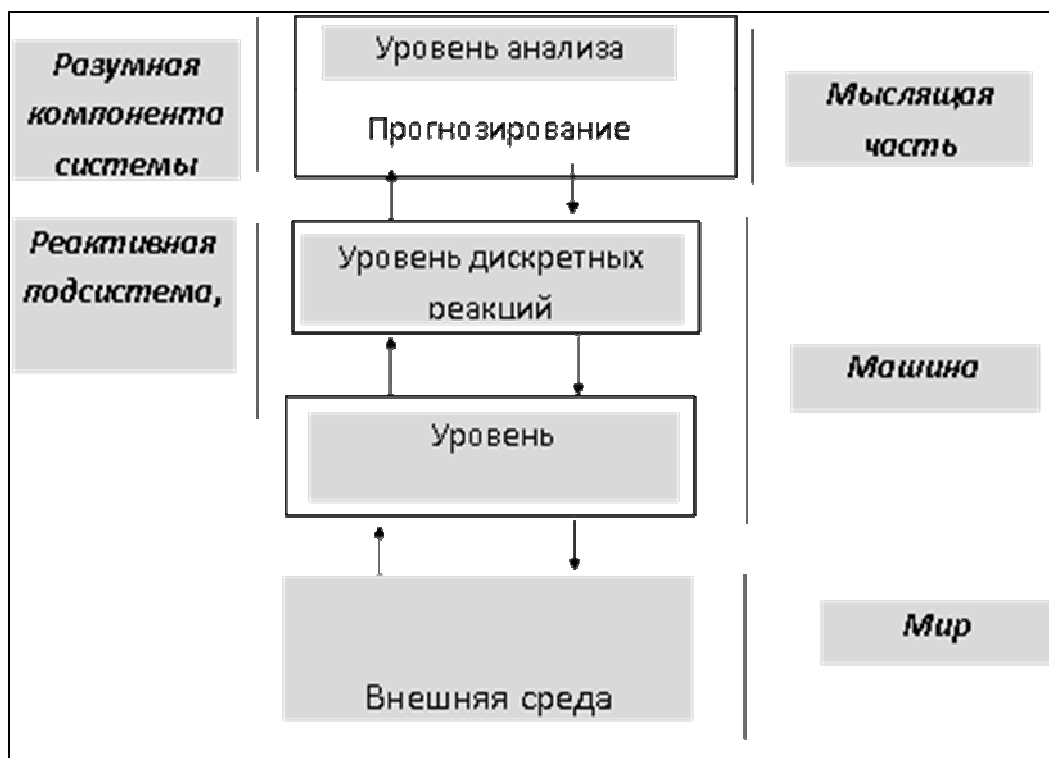


Рис. 12. IDS – архитектура (детализация)

Уровень анализа выполняет символические рассуждения, такие, как предсказание, планирование и перепланирование, а также является тем местом, где располагается компонента обучения агента.

Данная архитектура является типичным представителем многоуровневой архитектуры, которая относительно близка к архитектуре “TouringMachine” и отличается от нее вариантом распределения задач по уровням.

Достоинства архитектуры приведем в трех аспектах:

- имеет место явное разделение задач, которые требуют различных концептуальных и вычислительных рамок;
- позволяет при проектировании использовать различные инструментальные средства (языки, алгоритмы) для упрощения разработки;
- позволяет поддерживать процесс проектирования простыми программными инструментальными средствами, обеспечивая простоту процесса прототипирования, которыми автор располагает.

WILL-архитектура

Эта архитектура интенсивно использует метафоры и понятия, традиционно применяемые к описанию человеческой интеллектуальной деятельности, что делает ее привлекательной и понятной (Рис.13.).

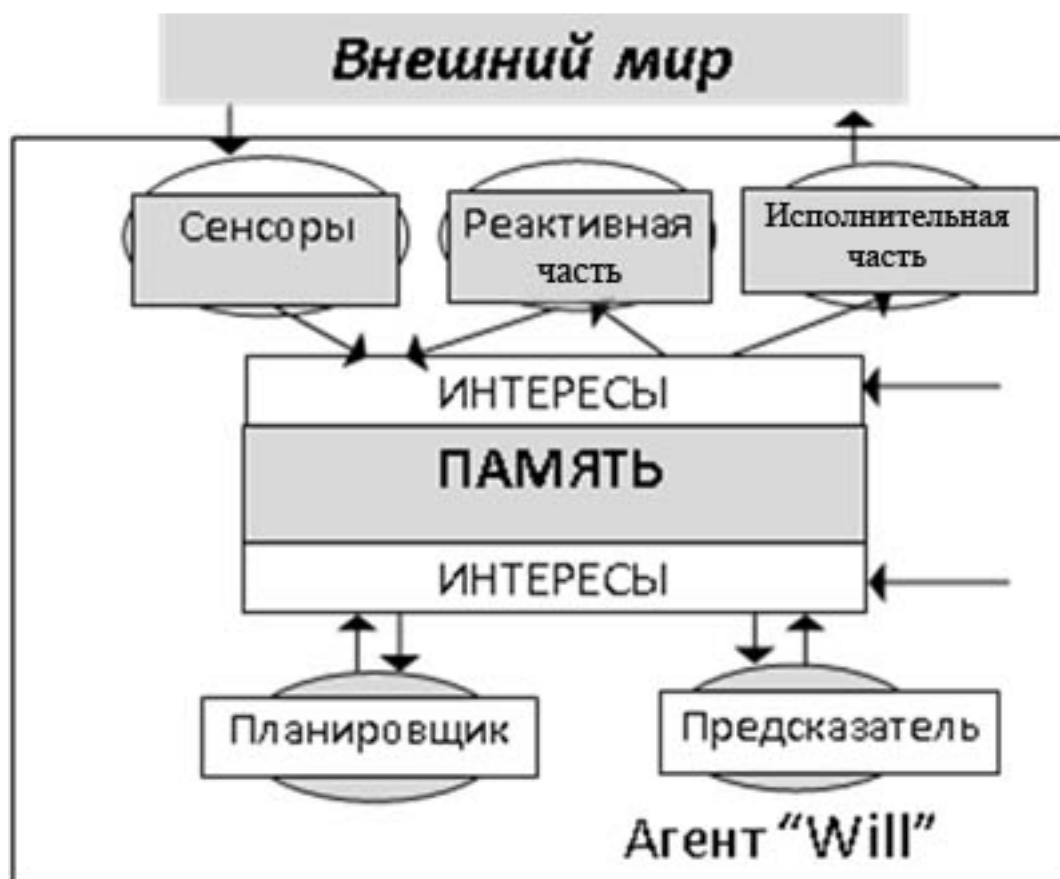


Рис.13. Will - архитектура агента

Данная архитектура рассчитана на одного агента, который имеет одну цель и его функционирование направляется его собственными мотивами, которые автор называет интересами ("concerns"). Вопрос о методах кооперации и коммуникации агентов такой архитектуры авторы оставляют без внимания.

Для того, чтобы агент функционировал, ему необходимы различные функции, включая восприятие. Авторы предполагают, что агент имеет для каждой из этих функций отдельный модуль. В частности, они предполагают, что агент имеет *Сенсорный блок*, *Планировщик* и *Исполнительное устройство* в качестве базовых модулей, которые каким-то образом должны быть интегри-

рованы.

Главной проблемой при этом является вопрос о том, как организовать совместную согласованную работу этих модулей, в частности, согласовать взаимодействие потоков информации и потоков управления. Чтобы решить проблему согласованного взаимодействия потоков информации, они предлагают применить нечто вроде схемы “бродкастинга”, когда соединены все входы и все выходы модулей между собой, так что любое сообщение, генерируемое тем или иным блоком, становится доступным любому другому блоку. Все эти сообщения собираются в глобальном буфере, который называется *Памятью*. Все блоки могут читать информацию из Памяти, кроме Сенсоров, и все они могут писать информацию в Память, кроме Исполнительного устройства. Каждый модуль может просто брать информацию из памяти, когда ему это нужно.

Авторы архитектуры полагают, что цели системы могут меняться и генерироваться “изнутри” агента, будучи обусловленными некими фундаментальными целями агента, которые авторы называют “интересами” (“concerns”).

Они определяют, как некие предпочтения агента находиться в каких-то состояниях и каких-то состояний избегать. Когда агент получает информацию, которая в соответствии с его интересами отвечает предпочтительному состоянию, то генерируется внутренний сигнал о том, что желательно, чтобы в этом состоянии среда оставалась и в будущем. Для каждого состояния внешней среды агент должен уметь оценивать меру его релевантности своим интересам. Это означает, что когда некий модуль обращается к памяти, он “видит” тот ее фрагмент, который имеет “наибольший заряд” и обрабатывает этот фрагмент. Наибольшее внимание модуля привлекается к тому событию, с которым агент не знает, что делать.

Разработчики данной архитектуры не анализируют сложность проблемы организации согласованной работы различных модулей агента в рамках архитектуры, которая по существу может быть реализована только при высо-

ком уровне самоорганизации системы, алгоритмы которой могут оказаться самым тонким местом при попытке реализации.

InteRRaP-архитектура

Основная идея этой архитектуры [18] в том, чтобы представить агента как множество уровней, которые связаны через управляющую структуру и используют общую базу знаний (см. Рис.14.):

Она состоит из пяти основных частей: *интерфейса с внешним миром; компоненты, основанной на поведении; планирующей компоненты; компоненты, ответственной за кооперацию с другими агентами и базы знаний агента.*

Интерфейс с внешним миром содержит 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. Базируется на понятии “фрагмента поведения” как некоторой заготовки реакции агента на некоторые стандартные ситуации, что позволяет агенту в стандартных ситуациях не обращаться к планированию на основе знаний и реализовывать значительную часть своего поведения рутинным образом с хорошей эффективностью.

Из базы знаний ей доступны только знания нижнего уровня абстракции, где содержится информация о фрагментах поведения.

Компонента, ответственная за планирование, содержит механизм планирования, позволяющий строить локальные планы агента, т.е. планы, не связанные с кооперативным поведением.

План представляется в виде графа, узлами которого могут быть либо конкретные наборы действий вплоть до элементарных шагов поведения, либо новые субпланы, подлежащие дальнейшей конкретизации. Таким образом, планирующая компонента активирует поведение (через нижележащую компоненту), направляемое целями. Она же участвует и в планировании, связанном с кооперативным поведением агентов. Эта компонента может использовать

знания двух нижних уровней абстракции.

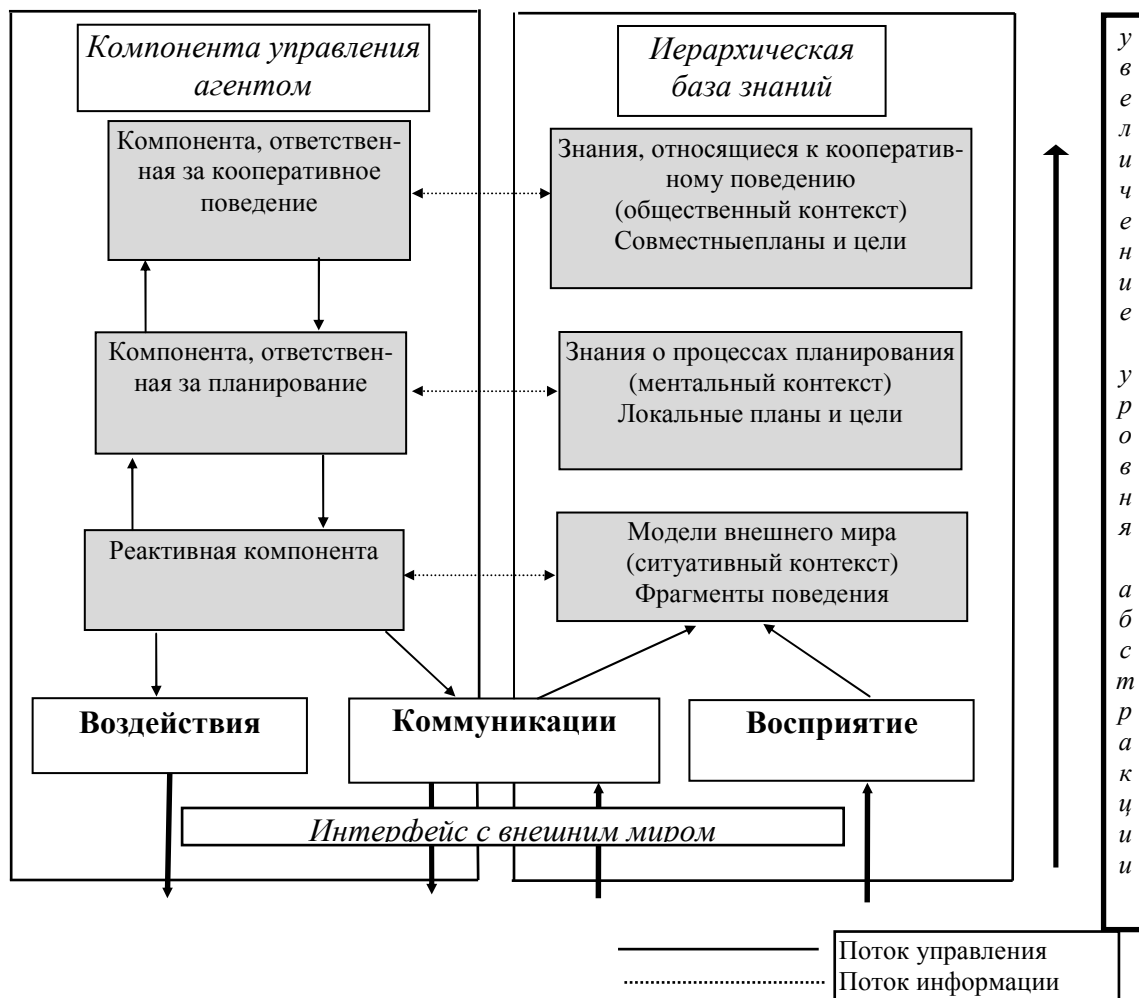


Рис.14. InteRRaP-архитектура агента

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. Этой компоненте доступны знания всех трех уровней абстракции.

База знаний агента имеет трехуровневую структуру и построена по принципу доски объявлений. 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 знаний агента и знаниям о текущем ментальном состоянии агента

(намерения, цели, планы). Наконец, третий уровень содержит знания и убеждения агента о других агентах, информацию о совместных планах, целях и намерениях, т.е. то, что связано с “общественным контекстом”.

Общее управление поведением осуществляется путем коммуникаций между уровнями. При некотором входном событии агент пытается распознать ситуацию во внешнем мире, и управление постепенно сдвигается снизу вверх до тех пор, пока не достигнет уровня, способного справиться с возникшей ситуацией. Существует три варианта реакции агента на внешние события:

- реакция с использованием только поведенческого уровня, когда этот уровень находит фрагмент поведения, адекватный ситуации, без явного привлечения локального планирования;

- реакция с использованием локального планирования, когда задача перемещается с нижнего уровня на уровень локального планирования, где и конструируется план;

- реакция с использованием уровня кооперативного планирования, когда поиск плана с уровня локального планирования перемещается дальше на уровень планирования кооперативного поведения.

Задание 6.

Перечень контрольных вопросов по теме 6:

1. Приведите общую классификацию архитектур агента.
2. Охарактеризуйте архитектуру, которая базируется на принципах и методах искусственного интеллекта (deliberative agent architecture”).
3. Охарактеризуйте архитектуру, основанная на поведении (reactive architecture).
4. Охарактеризуйте “реактивную архитектуру” (основанную на реакции системы на события внешнего мира).
5. Дайте полную характеристику архитектуры агентов, основанной на знаниях.
6. Дайте полную характеристику архитектуры на основе планирования (реактивной архитектуры).
7. Что Вы понимаете под многоуровневостью агентов в МАС?
8. Какие уровни чаще всего присутствуют в архитектуре агента?
9. Дайте характеристику горизонтально организованной архитектуре агента.
10. Дайте характеристику вертикально организованной архитектуре агента.
11. Приведите примеры композиционных архитектур многоагентной системы.

Лекция 7. Языки программирования агентов

7.1. Требования к языкам программирования агентов

7.2. Классификация

7.3. Сравнительная характеристика языков

В настоящее время не существует одного языка программирования, который в полной мере отвечал бы потребностям технологии многоагентных систем. Разрабатываемые в настоящее время агентские системы используют большой спектр различных базовых языков, но, к сожалению, ни один из них не может рассматриваться как истинно “агентно-ориентированный”. Имеются попытки расширить существующие языки, а также попытки использовать традиционные языки программирования. Существует ряд проектов по разработке новых специализированных агентских языков.

В рамках лекции рассмотрим основные требования к языкам программирования агентов (ЯПА), которые представляются наиболее существенными. Во второй части приведем классификация ЯПА и в третьей части приведем краткую характеристику и сравнение языков, которые используются при написании агентских систем в настоящее время.

7.1. Требования к языкам программирования агентов

Вначале приведем основные требования, предъявляемые к ним. К числу наиболее важных относятся следующие.

1. Обеспечение переносимости кода на различные платформы. Это требование возникает всегда, когда необходимо обеспечить агента свойством мобильности. Чтобы обеспечить мобильность агента, язык должен поддерживать механизм посылки, передачи, получения и выполнения кодов, содержащих агентов.

Имеются два различных подхода, которые решают проблему мобильности:

- передача агента в текстовой форме, как специального сценария (script) с последующей интерпретацией этого сценария на принимающей машине.

- 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. Оба из этих методов имеют свои преимущества и недостатки.

2.Доступность на многих платформах. Это требование непосредственно вытекает из предыдущего. Интеллектуальные агенты должны работать в гетерогенной компьютерной среде. Любой компьютер, получающий агента, должен быть способен принять и выполнить его.

3.Поддержка сетевого взаимодействия. Свойство агентов участвовать в переговорах и многие другие особенности агентов нуждаются в доступе к удаленным ресурсам. Поддержка сетевых услуг может включать семейства соответствующих программных интерфейсов (APIs) таких как: sockets, интерфейсы к базам данных, интерфейсы взаимодействия объектов (CORBA, OLE, ActiveX и т.д.), специальные механизмы, встроенные в язык (типа RemoteMethodInvocation в языке Java), специальные примитивы языка для осуществления переговоров агента и т.д.

4. Многопоточная обработка (“Multithreading”). Агент может 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.). Кроме того, процесс который выполняет всех агентов (и может, фактически, рассматриваться как мета-агент) должен поддерживать параллельное выполнение агентов. Последнего можно добиться с помощью отдельной виртуальной машины с реализованным режимом вытесняющей многозадачности и собственной стратегией разделения времени.

5. Поддержка символьных вычислений. Так как в рамках современных взглядов агент должен активно использовать достижения и методы искусственного интеллекта, было бы полезно иметь поддержку символьных вычислений и, возможно, логического программирования, встроенную в язык (подоб-

но PROLOG и LISP), а также иметь встроенный механизм выводов, включающий различные стратегии поиска решения. Автоматическое управление памятью и сборка мусора - стандартные средства для таких языков.

6. Безопасность, в частности, наличие системы защиты от несанкционированного доступа и “плохих кодов”. Это важно по многим причинам. Мобильные агенты, которые приходят из сети, могут таить в себе множество опасностей для принимающей машины, так как они выполняются в ее адресном пространстве. Для обеспечения безопасности перед передачей управления каждому агенту необходимо выполнить процесс авторизации агента, т. е. проверить, зарегистрирован ли он и имеет ли соответствующие полномочия (привилегии), чтобы выполнить то или иное действие или обратиться к некоторым ресурсам. Система безопасности должна предотвращать любые несанкционированные действия агента.

7. Истинная объектная ориентированность. Язык должен иметь механизмы наследования, рассматривать вызовы процедур как сообщения, передаваемые от одних объектов другим, включать возможности синхронного и асинхронного взаимодействия объектов, а также допускать параллельность внутри объекта.

8. Языковая поддержка свойств агента. Было бы, вероятно, удобно 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.

9. Эффективность, достаточная для потребностей прикладных программ.

7.2. Классификация ЯПА

Ниже приведем классификацию для языков, наиболее часто используемых в технологии интеллектуальных агентов. Заметим, что границы между группами зачастую условны.

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)

7.3. Сравнительная характеристика языков

Сводные характеристики языков и их сравнительные характеристики приведены соответственно в таблицах 1 и 2.

Java - вероятно, один из наиболее популярных языков используемых в последнее время для программирования агентов. Java представляет из себя язык программирования, подобный C++ по синтаксису, но более схожий со Smalltalk и Objective C по идеологии. Система программирования на Java включает в себя виртуальную машину Java и транслятор с Java в bytecode.

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 VirtualMachine*. Тем самым обеспечивается платформо-независимость Java - приложений на уровне байт-кода, который может быть прислан откуда угодно, включая Web-страницу, на которой содержится ссылка на код программы. Java VirtualMachine работает в среде вытесняющей мультизадачности и поддерживает облегченные процессы (threads). 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.

Java представляет собой истинно объектно-ориентированный язык про-

граммирования с сильной типизацией. Схожесть с C++ делает его простым для изучения программистами. В нем отсутствует предельно ясное распределение памяти и для повышения надежности программ из языка исключена арифметика указателей. Каждый тип данных понимается как класс объектов, любая функция является методом класса. Ее вызов рассматривается с объектно-ориентированных позиций как посылка сообщения объекту. Имеется встроенная расширяемая библиотека классов, включающая AbstractWindowToolkit (AWT) для создания пользовательских интерфейсов, классы поддержки основных типов данных, threads, сетевых возможностей, графики, мультимедиа, и т. д. К средствам повышения надежности следует отнести встроенную в язык обработку исключительных ситуаций (exceptions) и run-time контроль за выполнением программы, такой, как проверка выхода за границы массивов и т.д.

Язык Java может быть использован для создания двух типов программного обеспечения: автономных приложений, которые могут включать «родной» (т.е. машинно-зависимый) код и апплетов - платформо-независимых приложений, которые могут приходить из сети и запускаться с помощью поддерживающих Java Web - браузеров. Апплеты могут встраиваться в Web страницы после специальной метки языка HTML, указывающей ссылку на их код, и запускаться автоматически, когда страница открывается в браузере.

Язык Java был спроектирован с учетом возможности создания приложений работающих в распределенной среде. Кроме поддержки возможностей TCP/IP, таких как чтение URLs, работу с сокетами и обмен сообщениями на уровне датаграмм, в Java предусмотрен механизм удаленного вызова объектов, определенный в спецификации RMI (RemoteMethodInvocation). Этот механизм позволяет вызывать методы удаленных объектов, объявленные в специальном интерфейсе, причем синтаксически такой вызов выглядит идентично вызову простого метода. Эта схема предоставляет более гибкие возможности по сравнению с традиционным протоколом RPC (RemoteProcedureCall). Механизм сериализации (Serialization) позволяет сохранять объекты и графы

объектов в потоках данных (файлах, сетевых каналах) и восстанавливать их при необходимости. При этом обеспечивается достаточный уровень секретности передаваемой информации. С выходом спецификации Java IDL и нового продукта BlackWidow компании PostModernComputingTechnologies открылась возможность к сближению Java приложений и коммуникационного стандарта CORBA консорциума OMG (Object Management Group). Специальный компилятор позволяет, исходя из описаний интерфейсов на языке IDL CORBA, генерировать код Java объектов. Это позволит использовать вызовы объектов, находящихся в корпоративных сетях, из Java приложений и сделать Java - объекты доступными для этих вызовов.

KQML - язык и протокол для поддержки взаимодействия агентов в распределенных приложениях. Примеры возможных применений языка включают интеллектуальные многоагентные системы, такие как интеллектуальные агенты планирования, поддерживающие распределенную обработку. Перед определением интерфейсов систем, основанных на знаниях, перечислим типичные компоненты таких систем:

- *Приложение, рассчитанное на неподготовленного пользователя (EndUserApplication);*
- *Систему, основанную на знаниях (KnowledgeBasedSystem);*
- *Хранилище специальных знаний (Knowledge Base Repository) - хранилище знаний о содержимом базы знаний - используется для отслеживания истории доступа, целостности и других аспектов функционирования баз знаний;*
- *Система управления базами данных (DatabaseSystem);*
- *Активные сенсоры (Active Sensors)- отвечают за обмен данными (знаниями) с “внешним миром”.*

Возможны различные способы взаимодействия перечисленных компонент.

Разработчики KQML выделяют 3 направления, особенно важных с их точки зрения для коммуникаций интеллектуальных агентов:

- *Связность (Connectivity)* - фокусируется на способе связывания агентов между собой;
- *Архитектура (Architecture)*- акцентирует внимание на способе построения будущей системы (будет ли система статической, когда все компоненты известны уже на этапе проектирования/реализации или динамической);
- *Коммуникации (Communication)*- данное направление рассматривает синхронность/асинхронность обмена сообщениями между агентами.

При разработке языка были сделаны специальные предположения об основных моментах его использования для описания коммуникаций много-агентных систем:

Архитектура. Агенты - это отдельные процессы, работающие в одном адресном пространстве или на различных машинах, соединенные посредством Internet.

Коммуникации. Язык поддерживает следующие стратегии передачи информации: pointtopoint, multicast, broadcast.

Синтаксис. Сообщения между уровнями содержимого, сообщений и коммуникаций представлены в виде *s-выражений* языка Lisp. Между процессами выражения передаются как потоки ASCII -символов.

Протокол. Для транспортной поддержки KQML был создан специальный протокол - SKTP (*SimpleKnowledgeTransfer Protocol*).

Опишем уровни языка KQML, которые включают три уровня:

- Содержимое;
- Сообщения;
- Коммуникации.

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”. Тип сообщений определяет, что можно сделать с предложениями, содержащимися в сообщении, с этим связано само название KQML-сообщений на английском языке.

Все действия-сообщения можно разделить на следующие категории:

- *языка содержимого KQML*, который предполагает модель баз в форме множества предложений некоторого языка, который может быть объектным;

- *контекста рассуждений* - подмножества предложений (фрагмента) базы знаний; запросы и отклики должны делаться относительно некоторого контекста, при этом сообщения данного типа должны позволять определять контекст и переключаться между фрагментами базы знаний;

- *определения*, которые имеют целью активизировать и отменять определения;

- *ответы на вопросы*, которые позволяют задавать вопросы относительно истинности предложений;

- *отклики*, которые определяют набор сообщений для получения ответов на заданные вопросы.

SKTP - Simple Knowledge Transfer Protocol. Первые две реализации SKTP были написаны на языках CommonLisp и Пролог соответственно. В настоящее время разрабатываются интерфейсы на других языках. SKTP - это реализация стека протокола KQML. Как и KQML, SKTP состоит из нескольких уровней: содержимого, сообщений и коммуникации. Коммуникации между приложениями происходят на языке приложений, поддержку данного процесса обеспечивает уровень содержимого (*contentlayer*). Выражения, помечен-

ные для передачи, “обертываются” в сообщения - это т.н. уровень сообщений (*messagelayer*), реализуемый с помощью интерфейса библиотеки посредника (*FacilitatorInterfaceLibrary* - *FIL*). Уровень, отвечающий за стратегию передачи сообщений (*multicast, broadcast и т.д.*) называется коммуникационным и реализован в виде отдельного агента, называемого посредником (*facilitator*). На самом нижнем уровне, реально несущем структурированные данные между посредниками, находится протокол TCP/IP.

KIF - это язык 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 понято без обращения к специальному интерпретатору); он логически полон (обеспечивает выражение произвольных предложений логики предикатов первого порядка), обеспечивает представление знаний о представлении знаний; обеспечивает представление немонотонных правил вывода, определение объектов, функций и отношений.

KIF не претендует на звание 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. KIF используется при коммуникациях с внешними программами. Цель языка KIF аналогична цели языка Postscript - он не является языком, достаточно эффективным для представления внутренних знаний программ, но он удобен для независимой разработки программ, работающих со знаниями.

Перечислим основные свойства языка KIF.

1. Язык является декларативным, чем отличается от языков типа Prolog и Emucin.
2. Язык является логически полным.
3. Язык является транслируемым (*translatability*) - он предполагает значения для трансляции декларативных баз знаний в/из типичных языков пред-

ставления знаний.

4. Язык является хорошо читаемым, что упрощает его использование разработчиком баз знаний.

5. Язык может быть использован как язык представления знаний.

Существуют две разновидности языка KIF: *линейный* и *структурированный*. В линейном варианте все его выражения - это строки ASCII-символов, что удобно для хранения на устройствах с последовательным доступом. В структурированном варианте языка правильные выражения - это структурированные объекты. Структурированный язык удобен для использования при коммуникациях программ, работающих в одном адресном пространстве. Между линейным и структурированным представлениями одной и той же сущности существует взаимно-однозначное соответствие.

Соответствие между линейным и структурированным представлениями определяется следующим образом: строка ASCII-символов будет правильным выражением линейного варианта языка KIF тогда и только тогда, когда (1) она допускается интерпретатором языка CommonLisp, и (2) структура, порожденная интерпретатором CommonLisp будет правильным выражением структурированного KIF.

В структурированном KIF определены следующие синтаксические конструкции: word, expression, operator, constant, term, sentence, definition, rule и т.д.

Применимость KIF в рамках разработки агентов видится следующим образом. Ядро агента, а именно: подсистемы управления памятью, планирования, база знаний и т.д. пишутся на C++ или на одном из языков 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 отпадает необходимость в упомянутом выше трансляторе.

April: Agent Process Interaction Language. Это язык высокого уровня, который предлагает простой интерфейс к другим языкам программирования типа C. April ориентирован на реализацию многоагентных систем. Однако, April не является языком программирования агентов в том смысле, что он не предлагает таких высокоуровневых возможностей, как планировщики, механизмы вывода на основе правил и системы представления знаний. April скорее является объектно-ориентированным языком со средствами поддержки параллельного выполнения задач и рассматривающим объекты как процессы. Он является подходящей основой расширений для задач распределенного искусственного интеллекта и для прикладного программирования многоагентных систем.

AgentSpeak. Аналогом класса в данном языке выступает семейство, представителем (экземпляром) семейства является агент. Каждый агент обладает базой данных отношений с публичной и приватной частями, множеством сервисов и множества планов - процедур, о которых известно лишь то, выполнены они или нет. Язык обеспечивает распределенность хранения 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 синхронно/асинхронная. Каждому агенту присваивается личный почтовый ящик, в который складываются все приходящие на его адрес сообщения. Помимо 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, за-

прашивать с ожиданием ответа, запрашивать без ожидания ответа.

Несколько слов об операционной семантике языка. Язык требует, чтобы все семейства агентов были определены до начала выполнения средой агентских заданий (во время компиляции), динамическое создание агентов не поддерживается.

Каждый агент может находиться в одном из трех состояний: активный, ожидающий, неработающий (*idle*).

По прибытии к агенту сообщения оно кладется в его почтовый ящик, далее агент начинает обрабатывать сообщение. Это заключается в следующем. 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 предыдущем 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. Как следствие из этого, агент является многопоточковым процессом. Внешние и внутренние сервисы агента функционируют, конкурируя за ресурсы, поскольку агент может выполнять как внешние, так и внутренние сервисы.

Данный язык близок по духу таким языкам как *Agent0* and *PLACA*, от них он отличается тем, что здесь ментальное состояние агента рассматривается как объединение убеждений (отношений), целей (сервисов), планов и намерений. *Agent0* рассматривает ментальное состояние, состоящее из списков способностей, убеждений и намерений.

Agent0 и *Placa* трактуют агентно-ориентированное программирование как вид объектно-ориентированного программирования, создатели же *AgentSpeak* считают, что первое является расширением (продолжением) второго.

TeleScript. Первая коммерческая реализация концепции мобильного агента была сделана в среде TeleScript - технологии фирмы GeneralMagic. Данная технология основана на метафоре электронного рынка - общедоступной сети (*publicnetwork*), которая позволяет продавцам и потребителям товаров и услуг находить друг друга и заниматься совместным бизнесом.

TeleScript-технология оперирует следующими понятиями: места (*places*), агенты (*agents*), перемещения (*travels*), встречи (*meetings*), соединения (*connections*), полномочия (*authority*) и разрешения (*permits*). Далее кратко разъясняются перечисленные понятия:

- *Места.* TeleScript-технология 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 поэтому он может занимать несколько различных мест в одно и то же время. Агентские процедуры выполняются параллельно. В модели электронного рынка на типичном месте постоянно присутствует один, выделенный агент.

- *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. TeleScript позволяет коммуникационному пакету (*computerpackage*) - агенту (его процедурам и состоянию) перемещаться между компьютерами. Для перемещения между компьютерами агент выполняет инструкцию **go**. Инструкция включает *ticket* - данные о желаемом месте доставки, и других параметрах перемещения. В случае, если перемещение прошло успешно, агент получает уведомление об этом (его следующая инструкция выполняется уже на новом месте). В модели электронного рынка инструкция **go** позволяет агентам покупателей и продавцов располагать друг друга для более эффективного взаимодействия.

- *Встречи.* Встреча позволяет агентам вызывать процедуры друг друга. Встречи - это то, что “заставляет” агента перемещаться. Для встречи с расположенным рядом (co-located) агентом, агент выполняет инструкцию **meet**. Данная инструкция содержит требование (*petition*) - данные, определяющие агента, который “хочет” встретиться и другие параметры встречи. **Meet**-инструкция позволяет покупателям и продавцам осуществлять транзакции.

- *Соединения* (пока не реализованы на 1995 год). Они позволяют агентам обмениваться информацией с разных мест. Для соединения агент выполняет **connect**-инструкцию. Данная инструкция содержит несколько параметров, таких как цель (*target*) соединения. **Connect**-инструкция позволяет агентам обмениваться информацией на расстоянии.

- *Полномочия.* Технология позволяет агенту или месту распознавать полномочия другого агента/места, причем агент или место не могут ни скрывать, ни фальсифицировать свои полномочия. Анонимность исключена. Технологией предусмотрена проверка полномочий при перемещении агента между регионами (*networkregions*) сети - набором мест, расположенных на компьютерах, обладающих одинаковыми полномочиями. Для проверки полномочий агент или место выполняет инструкцию **name**. Результатом выполнения инструкции является *telename* - данные, позволяющие распознавать полномочия в рамках региона сети. 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 С и С++. Приложение может быть написано целиком на языке TeleScript, но чаще разработчики поступают иначе: агенты и оболочки мест пишутся с помощью TeleScript, а стационарные части приложения (интерфейсы с пользователем, базами данных и т.д.) - на С или С++.

TeleScript обладает следующими характеристиками:

- Полнотой.
- Объектно-ориентированностью.
- Динамичностью (*dynamic*). Агент может переносить информацию с места на место. Даже если при пересылке объект не известен на месте назначения, его класс следует вместе с ним по сети (код, определение класса).
- Сохранением (*persistency*). На каждом шаге выполнения агент и переносимая им информация безопасно сохраняется в не-*volatile* - памяти (*постоянной* - видимо, служебной памяти интерпретатора). Эта операция позволяет предотвратить крах компьютерной системы.
- Переносимостью и безопасностью. Компьютер выполняет инструкции, составляющие агента не напрямую, а посредством *engine*-интерпретатора. Агент может выполняться на любом компьютере, на котором установлен интерпретатор.
- Ориентированностью на коммуникации (*communication-centric*). В язык встроены инструкции, позволяющие агенту просто выполнять сложные сетевые задачи.

Agent-Tcl. Agent-Tcl - это система мобильных агентов, в которой агенты написаны на Tcl 7.4 и Tk 4.0. Agent-Tcl активно используется в задачах информационного поиска и прикладных программах информационного управления. Agent-Tcl в целом аналогичен языку TeleScript, за исключением того, что Agent-Tcl более облегчен и в настоящее время обеспечивает ограниченную защиту. Альфа - версия доступна на Unix платформах.

Oz - параллельный, объектно-ориентированный язык программирования, который был разработан в DFKI (Германия). Имеются несколько проектов в DFKI, использующих Oz совместно с архитектурой InteRRaP (см. предыдущий раздел). InteRRaP представляет из себя многоуровневую архитектуру, которая построена для модели взаимодействующих автономных агентов. DFKI предлагает параллельный язык программирования, приспособленный для прикладных программ, которые требуют сложных символьных вычисле-

ний, организации кооперации агентов и некоторых возможностей управления в реальном масштабе времени. Реализация Oz является законченной средой программирования, включающей объектно-ориентированный интерфейс к Tcl/Tk. Прикладные программы на Oz уже использовались для моделирования многоагентных систем, обработки естественного языка, виртуальной реальности, графических пользовательских интерфейсов, планирования и создания расписаний.

Obliq - это интерпретируемый язык без контроля типов, который поддерживает, распределенные объектно-ориентированные вычисления. Вычисление в Obliq может охватывать многократные потоки управления внутри одного адресного пространства, многократные адресные пространства в пределах одного компьютера, множество компьютеров в гетерогенной локальной сети, и, наконец, даже множество сетей в Internet. Вычисления в Obliq могут перемещаться по сети, сохраняя сетевые соединения.

Facile - язык программирования высокого уровня для систем, которые требуют комбинации сложного манипулирования данными с параллельными распределенными вычислениями.

AKL (Agent Kernel Language) - параллельный язык программирования, разработанный в Шведском Институте Информатики (SICS). В AKL вычисления выполняются агентами, взаимодействующими через хранилища ограничений и условий (storesofconstraints). Этот подход объединяет сразу несколько парадигм программирования. В соответствующих контекстах об AKL - агентах можно думать как о процессах, объектах, функциях, отношениях, или ограничениях. AGENTS - система для программирования в AKL. PENNY - это параллельная реализация AKL, для которой были получены очень перспективные результаты, и которая будет развиваться далее.

Scheme 48. Scheme - один из диалектов языка LISP. Широкое разнообразие парадигм программирования, включая императивный и функциональный стили, а также передачу сообщений, находят удобное выражение в данном языке. Реализация Scheme, основана на архитектуре виртуальной машины.

Python - переносимый, интерпретируемый, объектно-ориентированный язык программирования, разработанный в CWI (Амстердаме). Язык имеет изящный синтаксис; в него встроено небольшое число мощных типов данных. Python может быть расширен, путем добавления новых модулей, выполненных на компилируемом языке типа C или C++. Такие модули расширения могут определять новые функции и переменные, а также новые объектные типы.

Phantom - это интерпретируемый язык, разработанный для крупномасштабных интерактивных распределенных программ типа систем конференц-вязи, игр со многими игроками, и совместных инструментальных средств работы. Ядро языка основано на безопасном, расширенном подмножестве языка программирования Modula-3 и поддерживает ряд возможностей современного программирования, включая статическую типизацию, неявные объявления, объекты, облегченные процессы, и высокоуровневые функции.

Penguin - модуль языка Perl 5, который обеспечивает, набор функций для: (1) отправки зашифрованного Perl-кода с цифровой подписью к удаленной машине, на которой он будет затем выполнен; (2) получения кода и, в зависимости от того, кем он подписан, его выполнения с соблюдением соответствующих прав. Комбинация этих функций дает возможность для непосредственного кодирования на языке Perl алгоритмов обработки безопасных финансовых сделок по Internet, мобильных агентов, собирающих информацию, “оживленных” Web-браузеров, распределенных вычислений, удаленного обновления программного обеспечения, удаленного администрирования, распространения информации, конструкторов сетевых программ, и так далее.

Таблица 1. Сравнение языков.

Язык	Переносимость кода	Доступность	Сетевые возможности	Параллельность
Java	байт-код, виртуальная машина Java	Windows XP/NT/7/8, Solaris SPARC /Intel, HP-UX, OS/2, Macintosh, Linux	APIs (библиотеки классов), Remote Method Invocation, сетевая сериализация	Threads синхронизованные с помощью мониторов
TeleScript	интерпретация скриптов	Solaris SPARC, HP-UX, OS IRIX	TCP/IP, UDP, сетевая сериализация	множественные процессы, работающие в режиме вытесняющей многозадачности
Tcl/Tk	интерпретация скриптов	Macintosh, Windows XP/NT/7/8, Solaris		
Oz		Solaris SPARC, SunOS, SGII-RIX, HP-UX, DEC Ultrix, IBM RS6000, Linux		
Obliq	интерпретируемый язык		гетерогенные распределенные сетевые вычисления	множественные threads внутри одного адресного пространства, множественные адресные пространства в пределах машины, распределенные вычисления в гетерогенных сетях
April			UDP	
AKL				
Scheme 48				
Penguin				
Python	интерпре-	Windows	интерфейс к	

	тируемый язык	XP/NT/7/8, DOS, Macintosh, UNIX	TCP/IP	
Facile				
Agent Speak	интерпретируемый	пока не реализован		аналог потоков - намерения

Таблица 2 (продолжение). Сравнение языков.

Язык	Символьные вычисления	Обеспечение безопасности	Объектная ориентация	Встроенные агентские свойства
Java	не поддерживаются	есть	есть, без множественного наследования	нет
TeleScript	не поддерживаются	встроенные в язык и библиотеку классов средства	да	агенты, места, маршруты, встречи, соединения, авторизация, полномочия, инструкции: go, meet, connect, permit, name
Tcl/Tk	не поддерживаются			
Oz	поддерживаются			
Obliq	не поддерживаются		распределенные объектно-ориентированные вычисления легко встраиваемые в приложения на Modula-3	нет
April				намерения
AKL				
Scheme 48				
Penguin		кодирование (и цифровая подпись) Perl кода		

		переда- ваемого удаленной машине		
Python			да	
Facile				
Agent Speak			Агентно- ориентиро- ванный язык	BDI (beliefs-desires- intentions) архитекту- ра

Задание 7.

Перечень контрольных вопросов по теме 7:

1. Перечислите последовательно основные требования к языкам программирования агентов.
2. Приведите классификацию языков программирования агентов.
3. Осуществите сравнительную характеристику языков программирования агентов.
4. Охарактеризуйте язык программирования Java.
5. Дайте характеристику языку (ЯПА) KQML и протоколу для поддержки взаимодействия агентов в распределенных приложениях.
6. Что Вы понимаете под SKTP - SimpleKnowledgeTransfer Protocol.
7. Что из себя представляет язык программирования KIF.
8. April: Agent Process Interaction Language. Для решения каких задач предназначен данный язык программирования?
9. Язык AgentSpeak, его специфика и возможные приложения.
10. TeleScript. Первая реализация концепции мобильного агента в среде TeleScript.
11. Дайте подборку языка Oz - параллельного, объектно-ориентированного языка программирования.

Лекция 8. Программные интеллектуальные агенты

8.1. Самоорганизации и кооперация в компании.

8.2. Процесс самоорганизации в мультиагентной систем.

8.3. Архитектура и интерфейс мультиагентной системы.

8.4. Примеры применения многоагентных систем.

В последние годы в ряде публикаций [34,35] введен термин «Программные интеллектуальные агенты», под которым подразумевается новый класс систем программного обеспечения, которое действует либо от лица пользователя, либо от лица системы делегировавшей агенту полномочия на выполнение тех или иных действий. Рассмотрим в рамках данной лекции возможности и характеристику подобных систем и возможности их адаптации для промышленных предприятий и организаций.

Динамическое развитие предприятий в рыночных условиях предполагает высокую степень специализации и кооперации работ. Кооперация - одна из важнейших сторон современной индустрии, охватывающая все сферы деятельности человека: науку и производство, торговлю, услуги и т.д. Проблемы кооперации компаний тесно связаны с проблемами их внутренней организации. Чем больше направлений в деятельности компании, чем сложнее выпускаемые изделия, чем разнообразнее сама деятельность, тем больше роль кооперации, и тем более высокоорганизованным должно быть предприятие.

Для работы в этих условиях предприятия должны обладать высокой интеллектуальностью, гибкостью и мобильностью, что в результате должно обеспечивать, с одной стороны, возможность постоянной эволюционной адаптации к условиям рынка и, с другой стороны, возможность совершать революционные и неожиданные для конкурентов скачки в развитии, резко повышающие его конкурентоспособность.

Становится актуальным создание интеллектуальных систем управления и поддержки групповой согласованной деятельности нового класса, которые бы не только упрощали для пользователей принятие столь необходимых со-

гласованных решений, но и стимулировали процесс самоорганизации предприятия на всех уровнях деятельности. Как показывается ниже, главным свойством таких систем становится способность выявлять и разрешать потенциальные конфликты (противоречия) интересов, как во внешней, так и внутренней кооперативной деятельности предприятия.

Поскольку общепринятого определения “агента” пока не существует, введем на рассмотрение еще одно определение: «Мультиагент – это аппаратная или программная сущность, способная действовать в интересах достижения целей, поставленных перед ним владельцем и/или пользователем».

Отсюда следует, что в рамках мультиагентных систем (МАС) мы рассматриваем агенты, как автономные компоненты, действующие по определенному сценарию. Классифицируются агенты на:

- простые (simple);
- умные (smart);
- интеллектуальные(intelligent);
- действительно интеллектуальные(trulyintelligent).

Интерес для построения МАС в задачах инженерии знаний представляют в большей степени интеллектуальные и действительно интеллектуальные агенты, которые отличаются тем, что поддерживают помимо автономного выполнения, взаимодействия с другими агентами и слежения за окружением – способность использовать абстракции, адаптивность поведения, обучение на прецедентах и толерантность к ошибкам. Проблемы в создании МАС на принципах искусственного интеллекта состоят в том, что при проектировании точной и полной модели представления исследуемого объекта, процессов и механизмов рассуждения в нем – достаточно сложно создать адекватную и полную его модель.

Несмотря на определенные трудности, идея использовать агентов для решения разноплановых задач очень популярна в последнее время. Однако задача проектирования МАС и действительно интеллектуальных агентов требует специальных знаний и является ресурсоемкой задачей.

Программные интеллектуальные агенты – это новый класс систем программного обеспечения, которое действует либо от лица пользователя, либо от лица системы, которая делегирует агенту свои полномочия на выполнение тех или иных действий.

Они являются, по сути, новым уровнем абстракции, отличным от привычных абстракций типа – классов, методов и функций. Но при этом, разработка МАС позволяет создавать системы обладающие расширяемостью/масштабируемостью, мобильностью/переносимостью, интероперабельностью (технологическим партнерством для устойчивого развития), что несомненно важно при разработке систем, основанных на знаниях.

Рассмотрим еще раз свойства агента для программных интеллектуальных систем:

- Автономность: агенты функционируют без прямого вмешательства людей или кого-либо другого и владеют определенной способностью контролировать свои действия и внутреннее состояние.
- Методы (способы) общения: агенты взаимодействуют с другими агентами средствами некоторого коммуникационного языка.
- Реактивность: агенты способны воспринимать окружающую среду (которая может быть физическим миром, пользователем, взаимодействующим через графический интерфейс, коллекцией других агентов, Интернетом, или, возможно, всем вместе взятым) и адекватно реагировать в определенных временных рамках на изменения, которые происходят.
- Активность: агенты не просто реагируют на изменения среды, но и обладают целенаправленным поведением и способностью проявлять инициативу.
- Индивидуальная картина мира: каждый агент имеет собственную модель окружающего его мира (среды), которая описывает то, как агент видит мир. Агент строит свою модель мира на основе информации, которую получает из внешней среды.

- Коммуникабельность и кооперативность: агенты могут обмениваться информацией с окружающей их средой и другими агентами. Возможность коммуникаций означает, что агент должен получать информацию об его окружающей среде, что дает ему возможность строить собственную модель мира. Более того, возможность коммуникаций с другими агентами является обязательным условием совместных действий для достижения целей.

- Интеллектуальное поведение: поведение агента включает способность к обучению, логичной дедукции или конструированию модели окружающей среды для того, чтобы находить оптимальные способы поведения.

Считается, что каждый агент - это процесс, который владеет (располагает) определенной частью знаний об объекте и возможностью обмениваться этими знаниями с другими агентами.

Классификацию агентов в рамках программных интеллектуальных систем можно провести в двух направлениях - по их инструментальной реализации (языку программирования агентов) и по основным приметам, которыми они владеют. Как нами отмечено на предыдущей лекции, на сегодня не существует языка программирования или инструментальной системы разработки, которая бы полностью соответствовала требованиям построения агентов.

С точки зрения принципов распределенного объектно-ориентированного программирования (ООП) необходимость передачи методов может быть существенно сокращена в том случае, если может быть обеспечен удаленный доступ к общим методам посредством передачи ссылок на удаленные объекты, данных экземпляров этих объектов и их состояний. Однако в дополнение к концепции ООП, каждый агент имеет возможность создания копий самого себя с полной или ограниченной функциональностью, обеспечивая возможность настройки на среду путем исключения неэффективных методов и замены их новыми методами.

Традиционная для ООП схема класс/объект нарушается, т.к. агент имеет возможность постоянного изменения сценария поведения без его изменения в родительском классе.

Многозначное наследование позволяет создавать экземпляры агентов, смешивая сценарии поведения, схемы наследования и атрибуты, определенные в родительских классах. Следовательно, система разработки, которая бы полностью соответствовала требованиям построения агентов, должна была бы соответствовать таким требованиям: обеспечение перенесения кода на различные платформы, доступность на многих платформах, поддержка сетевого взаимодействия, многопоточковая обработка и некоторые другие.

Чаще всего в агентных технологиях используются, как мы уже отмечали: универсальные языки программирования (Java); языки, “ориентированы на знания”, такие, как языки представления знаний (KIF), языки переговоров и обмена знаниями (KQML, AgentSpeak, April), языки спецификаций агентов; специализированные языки программирования агентов (TeleScript); языки сценариев и scripting languages (Tcl/Tk); символьные языки и языки логического программирования (Oz).

Определяющим свойством агента является *интеллектуальность*. Интеллектуальный агент владеет определенными знаниями о себе и об окружающей среде, и на основе этих знаний он способен определять свое поведение. Интеллектуальные агенты являются основной областью интересов «агентной» технологии. Важна также среда существования агента: это может быть как реальный мир, так и виртуальный (компьютерный), что является важным в связи с наличием сети Internet.

От агентов требуют способности к обучению и даже самообучению. Поскольку обучение обуславливает наличие знаний у обучаемого, то обучаемым или самообучаемым может быть только интеллектуальный агент. Свойство «Умение планировать» подразделяет агентов на регулирующие и планирующие. Если умение планировать не предусмотрено (регулирующий тип), то агент будет постоянно переоценивать ситуацию и заново вырабатывать свои действия на окружающую среду.

Планирующий агент имеет возможность запланировать несколько действий на различные промежутки времени. При этом агент имеет возможность

моделировать развитие ситуации, что дает возможность более адекватно реагировать на текущие ситуации. При этом агент должен учитывать не только свои действия и реакцию на них, но и сохранять модели объектов и агентов окружающей среды для предсказания их возможных действий и реакций. Агент может иметь доступ к локальным и глобальным ресурсам. При этом агентов, которые имеют доступ к локальным ресурсам (ресурсы, к которым имеет доступ пользователь, в том числе и сетевые), называют персональными помощниками, они автоматизируют работу текущего пользователя, помогая ему в выполнении некоторых операций. Соответственно сетевой агент самостоятельно получает доступ к информации, не доступной пользователю напрямую либо доступ к которой не был предусмотрен. Важным свойством классификации есть мобильность - возможность менять свое местонахождение в окружающей среде.

Для программного агента под мобильностью понимается возможность передвигаться по сети от компьютера к компьютеру. Переходя от одного компьютера к другому, такой агент может обрабатывать данные и передавать по сети только результаты своей работы. Система, в которой несколько агентов могут общаться друг с другом, передавать друг другу некоторую информацию, взаимодействовать между собой, называется многоагентной (МАС).

Введем понятие многоагентной системы распределенного искусственного интеллекта.

Направление “многоагентной системы” распределенного искусственного интеллекта рассматривает решение одной задачи несколькими интеллектуальными подсистемами. При этом задача разбивается на несколько подзадач, которые распределяются между агентами. Еще одной областью применения МАС есть обеспечение взаимодействия между агентами, когда один агент может выработать запрос к другому агенту на передачу некоторых данных или выполнение определенных действий. Также в МАС есть возможность передавать знания. Построение программных систем по принципу МАС может быть обусловлено следующими факторами:

- некоторые предметные области применяют МАС в тех случаях, когда логично будет каждого из участников процесса представить в виде агента. Например, социальные процессы, в которых каждый из участников играет свою роль;
- если предметная область легко представляется в виде совокупности агентов, то независимые задачи могут выполняться различными агентами;
- устойчивостью работы системы: когда контроль и ответственность за выполняемые действия распределены между несколькими агентами. При отказе одного агента система не перестает функционировать. Таким образом, логично поместить агентов на различных компьютерах;
- модульностью МАС, что позволяет легко наращивать и видоизменять систему, т.е. легче добавить агента, чем изменить свойства единой программы. Системы, которые изменяют свои параметры со временем, могут быть представлены совокупностью агентов. Модульность обуславливает легкость программирования МАС.

Мультиагентные системы подразделяются на *кооперативные, конкурирующие и смешанные*.

Агенты в кооперативных системах являются частями единой системы и решают подзадачи одной общей задачи. Понятно, что при этом агент не может работать вне системы и выполнять самостоятельные задачи.

Конкурирующие агенты являются самостоятельными системами, хотя для достижения определенных целей они могут объединять свои усилия, принимать цели и команды от других агентов, но при этом поддержка связи с другими агентами не обязательна.

Под смешанными агентами понимаются конкурирующие агенты, подсистемы которых также реализуются по агентной технологии. Кроме общения с другими агентами должна быть реализована возможность общения с пользователем.

8.1. Самоорганизации и кооперация в компании

Для того, чтобы понять процесс самоорганизации в мультиагентной

системе следует довольно детально рассмотреть этот процесс в реальном мире, например в какой-нибудь компании. Зарождение и развитие компаний всегда связано с организацией коллективов людей, объединяющих свои ресурсы для достижения общих целей. Все начинается обычно с кооперации отдельных творческих людей или организации рабочих групп менеджеров и специалистов (в больших компаниях).

По мере получения результатов, сложившееся "вольное объединение" трансформируется в малую компанию или новые подразделения в большой компании, далее - в группу компаний, консорциум и т.д. При этом какова бы ни была форма существования организации, решения на всех уровнях принимаются некоторыми коллективами людей, способных сообща спланировать и скоординировать свою деятельность при решении общих задач. Простейшей формой такой организации коллективов людей являются широко распространенные на практике рабочие совещания, построенные по принципу "круглого стола".

Цель этих совещаний (в отличие от совещаний других видов) состоит в выявлении и разрешении общих проблем (противоречий) в кооперативной деятельности. Амбиции и иерархии здесь отходят на второй план, уступая место знаниям и опыту. Чем глубже дискуссия за "круглым столом", тем более остро обнажаются противоречия, и тем более продуманными и согласованными будут принимаемые решения (при условии признания всеми общих правил игры и некоторых других ограничений).

Можно утверждать, что на время проведения любого совещания в компании как бы организуется новое виртуальное структурное подразделение - временный творческий коллектив специалистов. В результате, возможно незримо, но именно эти коллективы начинают интеллектуально управлять компанией. При этом руководитель любого уровня высоко организованной компании постепенно приобретает роль арбитра (что вовсе не отрицает его участие в спорах как рядового специалиста), следящего за соблюдением общих для всех правил игры в условиях заданных ограничений. Высшей формой

групповой коллективной работы самоорганизация.

Самоорганизация - основа интенсивного развития компании, способность компании чутко реагировать на изменения во внешней среде, обоснованно и своевременно изменяя не только свое внешнее поведение, но и основополагающие принципы собственного устройства и функционирования.

В рассматриваемом случае самоорганизация должна проявлять себя в самостоятельном создании специалистами новых рабочих групп по каждому направлению деятельности компании, независимо от ведомственной принадлежности и уровней подчиненности специалистов (множество открытий и изобретений делается на стыке различных направлений деятельности, если обеспечены условия для общения специалистов).

Таким образом, деятельность компании (или группы компаний) в каждый момент времени можно представить конфигурацией "круглых столов" (рабочих групп), относительно постоянно действующих или создаваемых на самое короткое время. Внедрение принципов самоорганизации в компании должно начинаться не столько с нижних уровней управления, сколько с момента создания компании, начиная с уровня отдельных специалистов, микрогрупп, небольших отделов и далее (при этом обычно легче вырастить новую компанию, чем реорганизовать существующую).

В идеале, вся структура такой компании является виртуальной: она есть в каждый момент времени, но она столь изменчива, что фактически ее нет, как нет и иерархии управления - управляющие воздействия в равной степени иницируются как "сверху-вниз", так и "снизу-вверх" (если по-прежнему под "верхом" понимать традиционный менеджмент и финансы, а под "низом" - специалистов, технологию и т.п.). Однако, довольно быстро определяются новые лидеры (с точки зрения знаний и опыта), которые идут нарасхват в постоянно образующиеся "круглые столы". При этом каждый рядовой сотрудник самостоятельно работает, возможно, в целом множестве вертикальных и горизонтальных команд (групп), согласованно принимая важные решения в пределах своей компетенции и постоянно развиваясь и соответственно со-

вершенствуя собственные должностные инструкции. Любая новая идея, рожденная даже на самом нижнем уровне, подхватывается и обсуждается всеми другими звеньями и, в результате, возможно, начинает менять план действий и текущую структуру компании, не нарушая ее обязательств перед партнерами.

Интуитивно ясно, что очень дорогой платой за столь высокую степень согласованности принимаемых решений (а следовательно, и организованности компании) является экспоненциальное возрастание накладных затрат на сам процесс согласования решений на всех вертикальных и горизонтальных уровнях компании, так как теоретически возможно установление отношений "каждый - с каждым". В обычной повседневной практике компаний столь сложный процесс согласования решений просто невозможен.

В результате весьма распространенным в практике становится феномен "перегруженного руководителя", замыкающего в себе всю "организацию" предприятия. Жизнь всей компании начинает зависеть от одного человека. Доступ к такому руководителю и его ресурсы настолько ограничены, что любые реформаторские идеи, для восприятия которых требуется глубокое понимание и, следовательно, время, обычно рано или поздно становятся просто обреченными на гибель.

Для таких компаний и руководителей в первую очередь и предназначен предлагаемый ниже подход. Потенциальная мощь самоорганизующегося "коллективного разума" предприятия способна превосходить возможности любого, даже гениального руководителя. Именно это и заставляет искать новые пути решения проблемы кооперации и самоорганизации организаций, основанные на применении компьютеров для поддержки согласованной деятельности. Именно эту схему самоорганизации взята за основу при разработке моделей деятельности организаций и людей в различных предметных областях в мультиагентных системах интеллектуальными агентами.

В работах [34, 35] для решения этой проблемы предлагается подход, базирующийся на идее Миров и Агентов деятельности. В отличие от известных

подходов, данный подход реализуется созданием общего Мира деятельности кооперирующих сторон и Миров деятельности каждой из них, т.е. путем создания единой комплексной среды деятельности менеджеров и специалистов, обеспечивающей воспроизводство главных компонент процесса деятельности каждого предприятия и взаимопонимание между людьми.

Использование Интеллектуальных Агентов в комбинации с Мирами действий и рассуждений позволяет менеджерам моделировать все три главные ипостаси кооперации: коллективное поведение, мышление и коммуникацию участвующих сторон. За счет этого каждой из сторон на своего Агента может быть возложена миссия согласования большинства возникающих проблем, для чего ниже рассматривается возможность реализации виртуального круглого стола, предлагающего Агентам общий Мир действия. В рассматриваемом подходе мир действий - это модель среды деятельности, базирующаяся на знаниях.

Главное ее отличие от традиционных систем моделирования состоит в том, что эта система содержит модель пространства и предоставляет прямой доступ к объектам Мира в этом пространстве для выполнения действий, моделируя реакцию на эти воздействия в соответствии с законами Мира. Модели физической реальности, позволяющие моделировать эффект присутствия субъекта деятельности, обычно называют виртуальными. Рассмотрим конструкцию предлагаемых Миров более подробно.

Принципы построения и функционирования Миров кратко могут быть описаны следующим образом:

- Мир состоит из объектов, способных взаимодействовать друг с другом в соответствии с законами Мира; для пользователя Мира представляются сценами, состоящими из заданных объектов, с определенными отношениями между ними, и объектами, потенциально применимыми в сцене;

- объекты Мира определяются своими свойствами, обеспечивающими их способность вступать во взаимодействие с другими объектами; состояния объектов определяются их свойствами и отношениями; потенциально воз-

можные свойства объектов определяются законами Мира, действующими в сцене;

- законы Мира задаются сценариями действий, которые определяются как правила изменений состояния объектов Мира; более простые сценарии позволяют составлять более сложные сценарии;

- отношения между объектами определяют связи между ними; наиболее распространенными отношениями являются "целое-части", "принадлежность", "меры" и ряд других;

- основные концепты выражаются в атрибутах вещества, пространства и времени, энергии и информации.

Представленные базовые категории позволяют конструировать Миры действий в различных предметных областях. Рассмотренные принципы позволяют также создавать Миры рассуждений, такие как Механика и Оптика, Алгебра и Геометрия и т.д. Эти же принципы оказываются пригодными для построения Миров экономики и политики, технологии и торговли и т.п. Фактически, при создании Мира конкретного сектора рынка или Мира отдельного предприятия, строится полипредметная база знаний (в форме семантической сети), которая в дальнейшем и используется в рассуждениях Агентов. Ее основное отличие от принятых подходов состоит в ориентации на описание "действий" и использование соответствующей логики действий. Рассмотрим теперь соответствующую модель устройства памяти и мышления Агента. В структуре памяти Агента выделены следующие компоненты (рис. 15):

- долгосрочная память Агента, содержащая полные семантические сети предметных областей знаний; эта память пополняется знаниями в процессе обучения Агента и постоянно трансформируется и систематизируется, через нее, как через сито, пропускаются все входные факты;

- память (пространство) сознания, содержащая образы объектов Миров, являющаяся среднесрочной. В этой памяти содержатся описания сцены в каждом из Миров (также в форме семантической сети) и здесь же выполняются основные умственные операции над образами объектов;

- память фактов, а также память сценариев - наиболее часто изменяемые структуры памяти (оперативная память).

В памяти фактов находятся исходные и конечные, а также все промежуточные факты, получаемые в процессе рассуждений и расчетов.

Память сценариев также подвергается преобразованиям, в первую очередь, связанным с обобщением и конкретизацией сценариев.

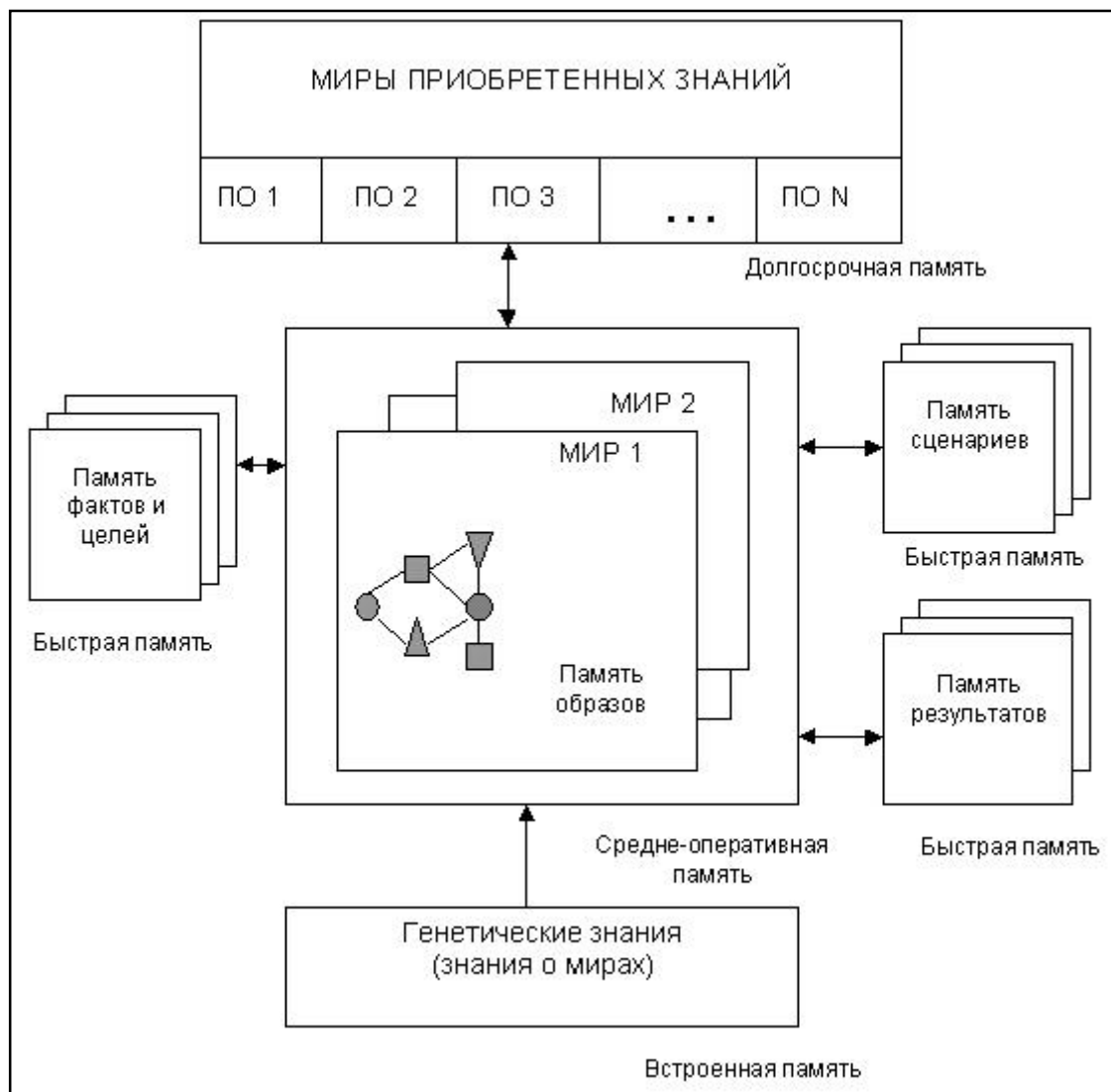


Рис.15. Структура памяти Агента

Память генетических знаний - это жестко встроенные в систему и неизменяемые знания, здесь - знания о конструкции и функционировании Миров.

Общие принципы мышления Агента являются вполне традиционными и включают следующие три основные фазы (Рис. 16):

- восприятие, в процессе которого осуществляется построение модели сцены в загруженном Море;
- познания, в процессе которого формируется сценарий действий субъекта для достижения поставленных целей;
- исполнения, в процессе которого осуществляется исполнение намеченного сценария с постоянным сопоставлением ожидаемых и наблюдаемых результатов.

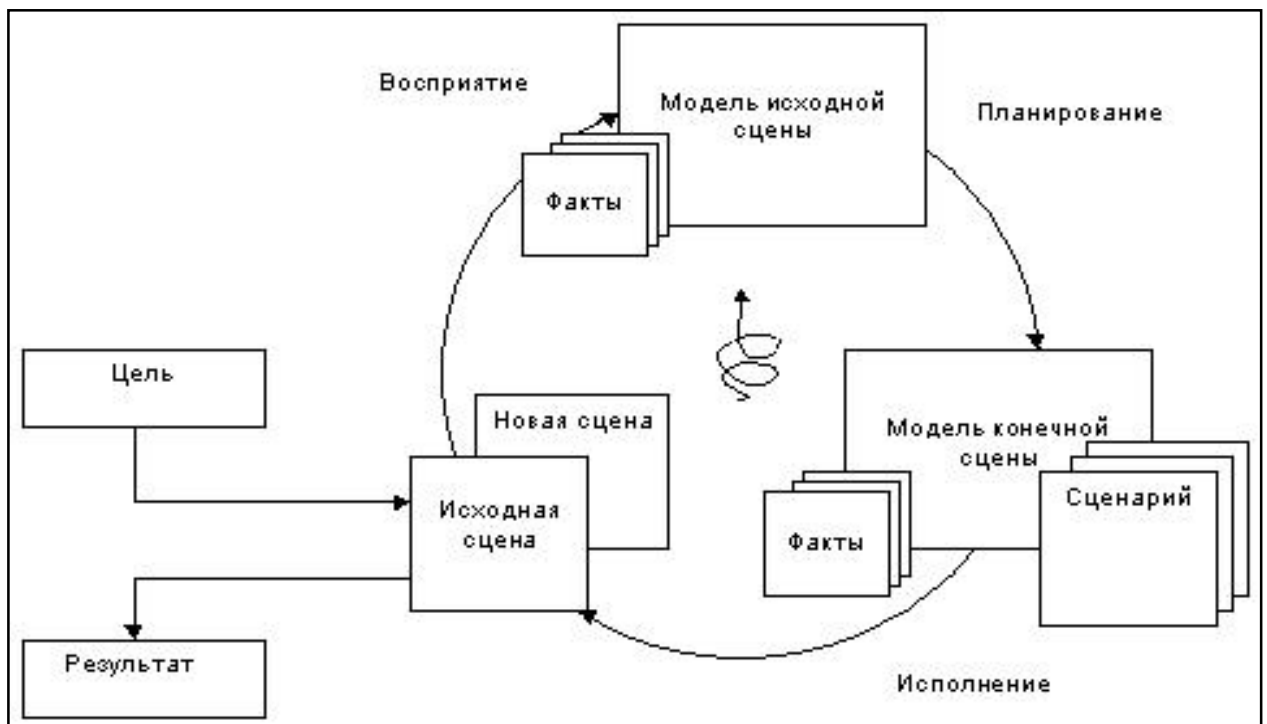


Рис. 16. Основные фазы мышления Агента

Однако, в отличие от других систем, в настоящей системе реализация этих фаз осуществляется через два базовых механизма: абстрагирования и конкретизации, тесно связанных между собой. Образно говоря, мышление Агента напоминает движение поршней в двигателе: вверх путем абстрагирования, вниз - путем конкретизации и т.д. При "послойных" рассуждениях основное время тратится на выполнение умственных операций - действий с образами объектов (понятиями), которые изменяют состояния сцены и, тем самым, ограничивают применение дедуктивных рассуждений. Используемая при этом логика действий также существенно отличает предлагаемую модель

от традиционных дедуктивных систем. Еще одна важная особенность предлагаемой модели - ориентация на выявление противоречий. В рассмотренной выше системе деятельности типовыми являются противоречия между знаниями и орудиями, целями и средствами деятельности, сценариями действий индивида и его внутренними способностями и ряд других. Типология этих противоречий исходно задается в системе и далее постоянно пополняется.

8.2. Процесс самоорганизации в мультиагентной системе

Для моделирования процесса переговоров между членами временно организуемых рабочих групп или их Агентами в разрабатываемой мультиагентной системе реализуется виртуальный круглый стол. Виртуальный круглый стол может реализовываться как через локальную сеть, так и через глобальную сети (Рис. 17):

Процедура согласования решений организуется следующим способом:

1) конфигурируется начальная сцена общего для всех Агентов Мира действий и задаются цели (задача), общие ресурсы и ограничения;

2) каждый из Агентов считывает состояние сцены и запускает процесс восприятия, планирования действий и их исполнения (при этом загружаются и перезагружаются необходимые Миры знаний и строится модель исходной сцены в этих Мирах); первый из Агентов, спланировавший свою деятельность делает первый ход, предлагая первое действие из своего сценария;

3) если действие удовлетворяет общим ограничениям и не вызывает противоречий с планами других Агентов, оно считается предварительно принятым. Если нарушены общие ограничения, Агент обязан поменять свои планы, если эти ограничения не нарушены, необходимо решить, кто будет вынужден изменять свои планы: первый Агент или другие, сделавшие свои ходы ранее;

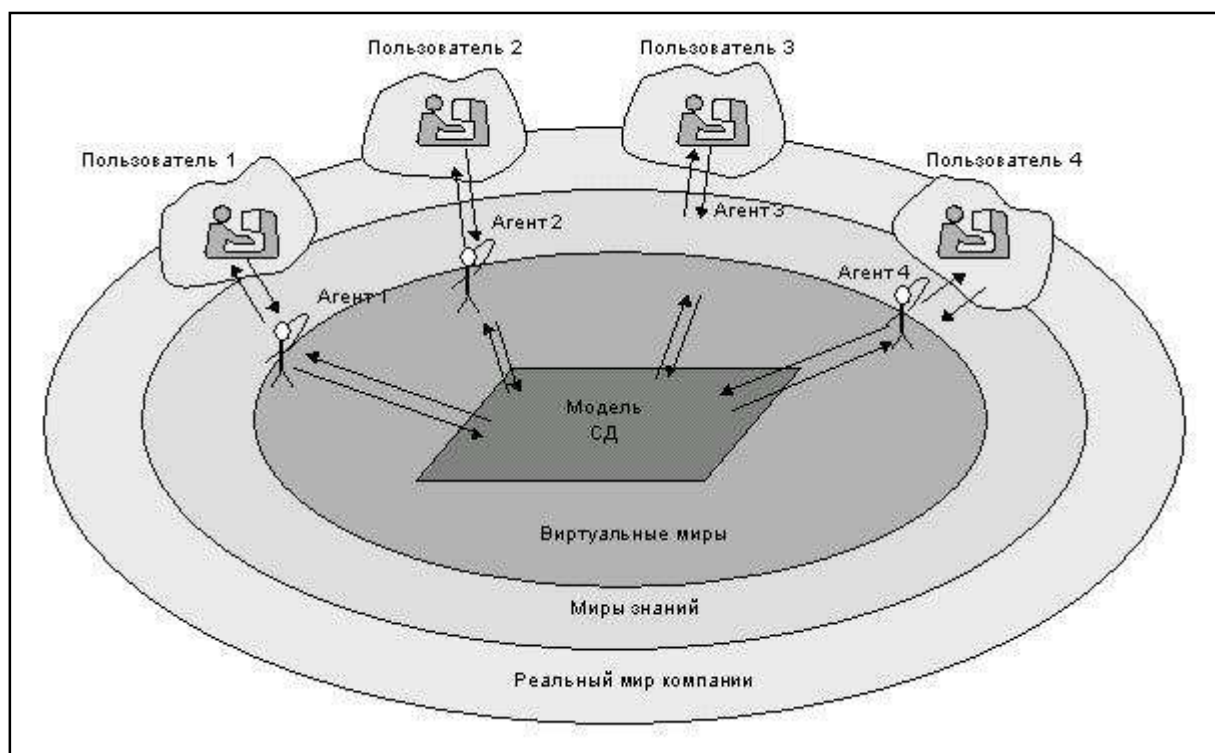


Рис. 17. Виртуальный круглый стол

4) очередные Агенты делают свои ходы, выполняя очередные действия из своих сценариев. Если какой-либо Агент вынужден поменять свое решение на каком-либо ходу, делается откат всего процесса переговоров для этого этапа и весь процесс согласования начинается вновь и т.д.

5) процесс согласования заканчивается, когда достигнута заданная цель. Очевидно, что данная процедура связана с возможным перебором всех вариантов решений - скорость ее сходимости зависит от глубины базы знаний и интеллектуальных способностей Агентов.

Для людей подобная процедура оказывается слишком трудоемкой. В данном же случае, один и тот же Агент менеджера или специалиста может принимать участие одновременно в целом ряде рабочих совещаний. Чтобы в полной мере ощутить проблему, достаточно представить себе объем согласований, выполняемых, например, при разработке месторождений нефти, когда за круглым столом могут оказаться геофизик и бурильщик, специалист по прокладке трубопроводов и строитель, экономист и социолог, специалист по охране окружающей среды и т.д. Что будет, если спустя полгода общих уси-

лий выясняется, что один из проектировщиков заложил в сценарий неверные данные и всем другим также придется начинать заново? Не меньше согласований происходит при подготовке больших сделок и в рассматриваемых примерах, если в этот процесс вовлекаются все потенциальные участники кооперации. Несмотря на наличие множества других сложностей, связанных со сходимостью данного процесса, разрабатываемая система не знает этой главной проблемы и реализует его, например, через Internet, независимо от местоположения участников переговоров.

8.3. Архитектура и интерфейс мультиагентной системы

Предлагается следующая архитектура интеллектуальной системы поддержки согласованной кооперативной работы, позволяющая моделировать деятельность и рассуждения специалистов или менеджеров с целью выявления потенциальных конфликтов между ними (Рис. 18).

База знаний среды деятельности (СД) - содержит описания среды деятельности, целей и задач, знаний и орудий, сценариев действий, а также всех других компонент рассмотренной выше структуры систем деятельности.

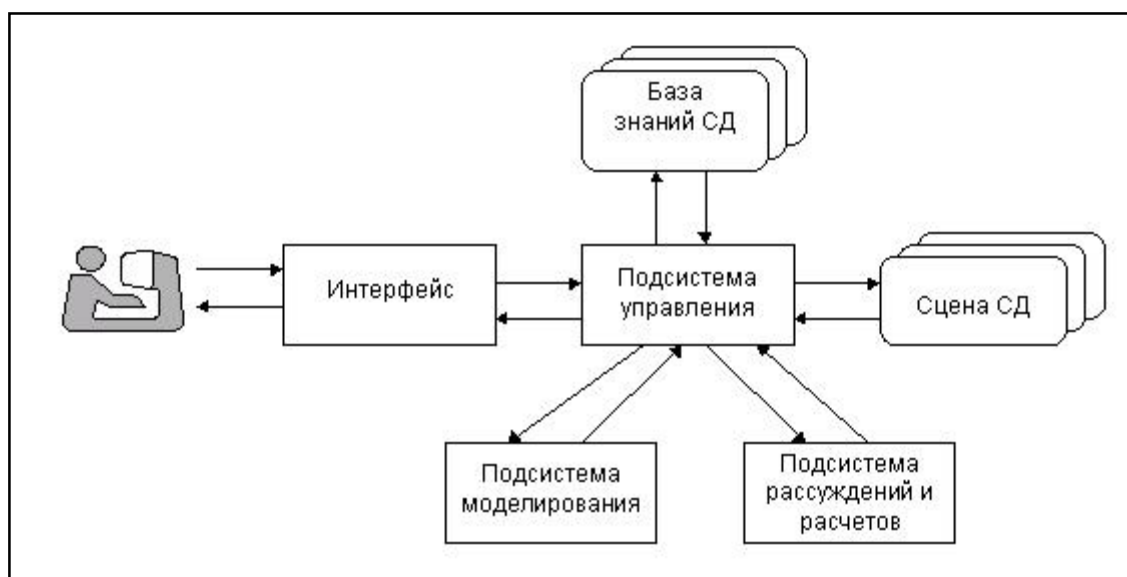


Рис.18. Архитектура интеллектуальной системы поддержки согласованной кооперативной работы

- Моделирующая подсистема - позволяет моделировать процессы деятельности (поведения субъектов деятельности).

- Подсистема расчетов и рассуждений - позволяет моделировать процессы рассуждений (мышления субъектов деятельности).
- Управляющая подсистема - реализует процессы поддержки согласования решений (процедуры виртуального круглого стола - процедуры коммуникации субъектов деятельности). Кроме того, данная подсистема выполняет функции конструктора Миров и конструктора сцен Миров.
- Интерфейсная подсистема - обеспечивает взаимодействие с пользователем; Сцены СД - текущие сцены деятельности.

Как видно из предлагаемой схемы, основные компоненты структуры системы связаны с основными моделируемыми компонентами деятельности: поведением, мышлением и коммуникацией. Аналогичным образом устроен и интерфейс системы, в котором выделены следующие основные поля:

Поле задания - формализованная постановка задачи;

Поле действия - рабочее поле для построения сценариев действий (здесь представлен интерфейс одной из подсистем для моделирования деятельности коммерческой компании, работающей в сфере агропродукции), в котором создаются и моделируются сцены общего Мира действий и индивидуальных Миров рассуждений;

Поле Агентов - здесь отображаются Агенты текущей рабочей группы, которые либо сами активизируются при совершении каких-либо действий или рассуждений, либо могут быть активизированы по инициативе пользователя;

Магазин объектов - список возможных партнеров по кооперации (или внутренних подразделений компании), которые могут быть размещены в рабочем поле;

Магазин договоров - список возможных отношений между компаниями (договора учредительские, кредитные и лизинговые, договора реализации, договора купли - продажи и т.д.).

В рассматриваемой подсистеме пользователь может вызвать в начальный момент, например, пиктограмму (модель) банка, птицефабрики, торгового дома и магазина. Далее конфигурировать начальную сцену для моделиро-

вания, описав интересующие его отношения между ними. Например, торговый дом может взять кредит в банке, осуществить оптовую закупку товара на птицефабрике, отдать товар на реализацию в магазин. При появлении крупного оптового покупателя товара птицефабрики, он вводится в систему и при этом моделируется возможность успешной реализации соответствующей сделки. Все операции осуществляются путем активизации соответствующих пиктограмм, что открывает для каждого объекта его индивидуальное поле действий. При этом, например, можно в ручном режиме осуществить взятие кредита или отгрузку товара, приобрести акции какого-либо предприятия и т.п. В системной части меню имеются клавиши доступа к базе знаний (для режимов просмотра и обучения), конфигурирования моделируемых параметров, помощи пользователю и ряд других.

8.4. Примеры применения многоагентных систем

В настоящее время выполняется несколько проектов по разработке многоагентной системы моделирования процессов кооперации и самоорганизации. В подобных проектах участвуют такие компании, как:

- Компания по разработке мультимедиа компакт-дисков со смежным производством печатной и видео продукции.
- Крупная коммерческая компания, занимающаяся экспортом и импортом продуктов питания, а также инвестициями в производство агропродовольственной продукции.
- Холдинговая компания, владеющая крупным магазином бытовой электроники, рестораном и сервис - центром.

Главная задача этих систем - организация деятельности круглых столов с привлечением специалистов из различных подразделений компании и поддержка процесса переговоров между ними. Ниже приведены примеры взаимодействия между системами, создаваемыми для различных компаний и их работниками.

Пример 1. Менеджер по маркетингу обнаруживает, что на рынке вот-вот появится смартфон, по функционалу аналогичный недавно запущенному в

производство в компании. Ввод этой информации в систему, актуализирует весь ряд подразделений компании, связанных с расчетом прибыльности проекта, его реализацией, рекламой продукта и т.п. Система ведет список подразделений, согласовывающих решение и состояние этого вопроса. По мере решения вопроса по подразделениям, система пересматривает важность других дел сотрудников в соответствии с их должностными инструкциями, отдавая приоритет решению данного вопроса. В результате проводимых обсуждений данный проект может быть остановлен вовсе, либо, наоборот, завершен в ускоренные сроки с привлечением дополнительных внешних специалистов и концентрацией других ресурсов, что в свою очередь вносит существенные коррективы в деятельность подразделений. Организованные системой рабочие группы представлены на рисунке 19.

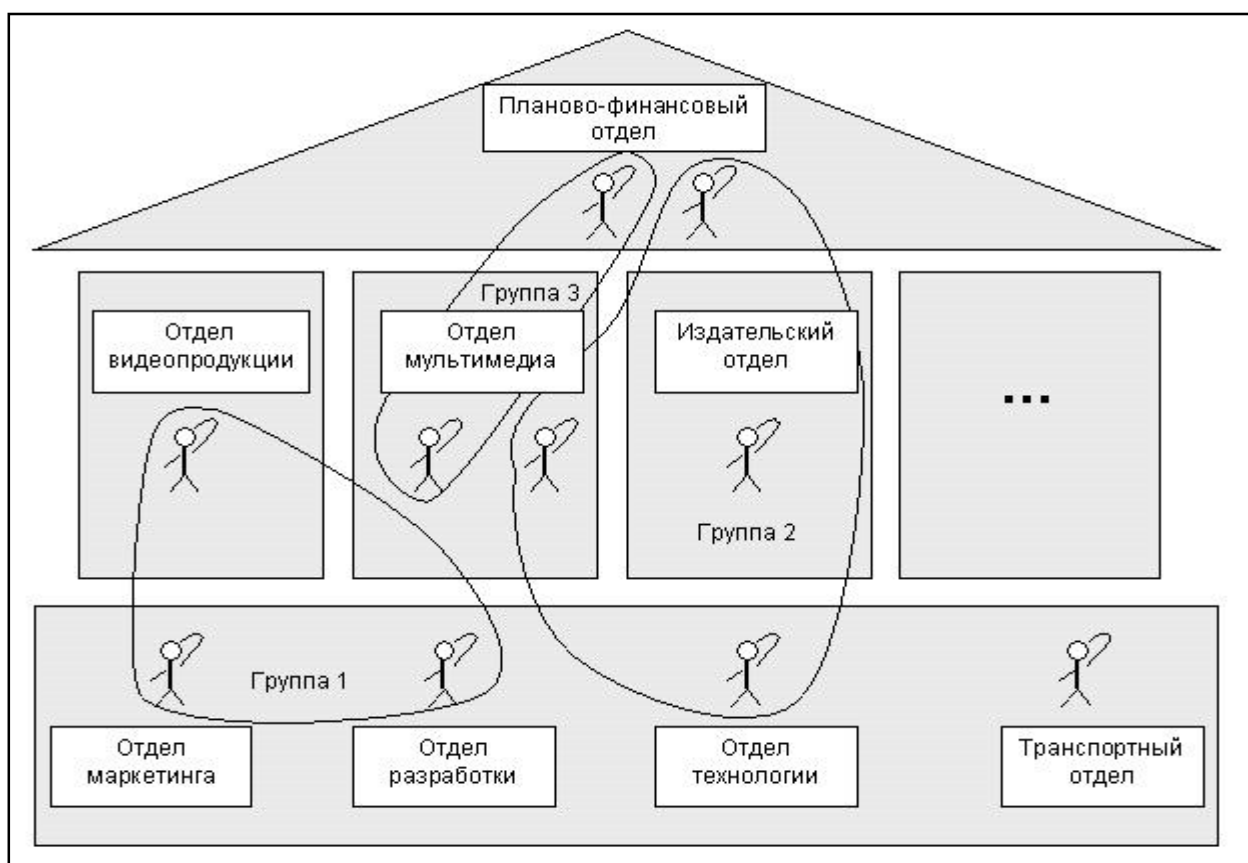


Рис.19. Структура мультимедиа-компании

Пример 2. Менеджеры, ежедневно заключающие контракты на оптово-розничную продажу продуктов питания, открывая систему, видят все множество заключенных контрактов с состоянием каждого из них, а также склады-

вающуюся общую ситуацию на рынке и в компании. Цель каждого из них - максимизировать прибыль по своей сделке (от сумм которых они получают комиссионные). Однако, в ряде случаев, максимизация прибыли по одной из сделок, может принести убытки компании в целом. Во избежание этой ситуации каждый из менеджеров должен промоделировать свою сделку на фоне общей деятельности, задавая планируемый сценарий по шагам. Результаты моделирования каждого шага показывают, насколько соотносится сделка с имеющимися в распоряжении менеджера кредитными ресурсами, складскими помещениями, холодильниками, транспортом и т.д. В случае возникновения противоречий, сделки ранжируются и заново пересогласовываются с другими менеджерами и ответственными за соответствующие ресурсы, а далее утверждаются и окончательно упорядочиваются по времени. Рабочие группы ("круглые столы"), организованные системой в процессе решения этой проблемы представлены на Рис.20.

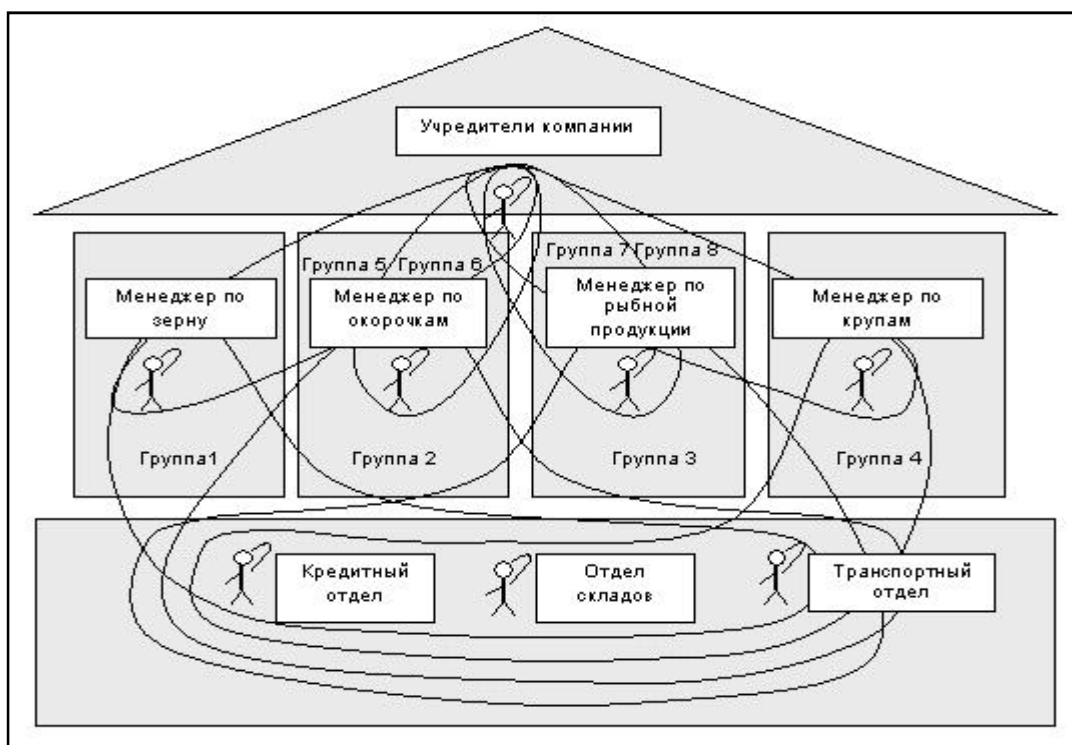


Рис.20. Структура оптово-розничной компании

Пример 3. Менеджер сервис-центра, готовящий ежемесячный баланс по отремонтированному оборудованию, обнаруживает расхождение между оформленными бланками заказов и имевшимся в начале месяца на складе за-

пасными частями, например, по ассортименту или количеству. Менеджер вызывает систему и дает формализованное описание проблемы. Анализируя сложившуюся проблемную ситуацию, система приходит к выводу, что причина либо в неверном заполнении бланков заказов на работы приемщицами центра или ошибка в только что внедренной программе. Для разрешения этой проблемы система активизирует создание рабочей группы, включающей менеджера, программиста, приемщиц и кладовщика, каждый из которых должен проверить свои действия (обратим внимание, что каждый - из своего подразделения). Пусть в результате обнаруживается ошибка в программе подготовки отчетов, вызванная методикой учета товара на складе. Изменение этой методики, в свою очередь, должно быть согласовано с бухгалтером центра (и, возможно, с директором), а если изменения коснулись технологии заполнения бланка, менеджер и приемщицы должны быть заново обучены программистом. Это вызывает организацию новых временных рабочих групп, которые действуют до момента восстановления ситуации. Вся эта процедура может потребовать как полного блокирования всех действий по оформлению заказ - нарядов, так и осуществляться в фоновом режиме.

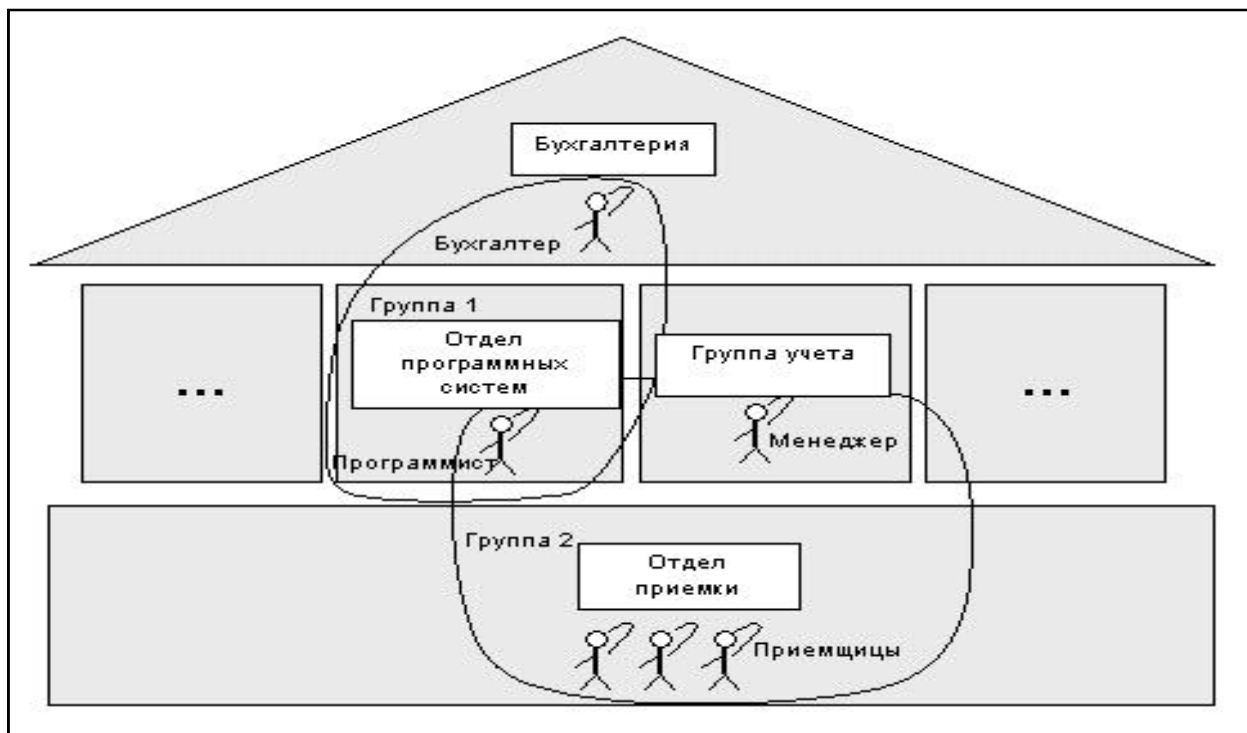


Рис.21. Структура серверного центра

Рабочие группы, сформированные системой в процессе решения этой проблемы, представлены на рисунке 21.

Таким образом, в лекции приведен мультиагентный подход к моделированию процессов самоорганизации и кооперации на современных предприятиях и компаниях, который основан с конструированием виртуальных миров деятельности специалистов и созданием интеллектуальных агентов для этих специалистов.

Задание 8.

Перечень контрольных вопросов по теме 8:

1. Дайте толкование термина «Программные интеллектуальные агенты».
2. Дайте классификацию агентов в рамках мультиагентных систем.
3. Программные интеллектуальные агенты – характеристика, новый уровень абстракции.
4. Приведите свойства программных интеллектуальных агентов.
5. Охарактеризуйте отдельно свойство интеллектуального агента – «интеллектуальность».
6. Охарактеризуйте «понятие многоагентной системы распределенного искусственного интеллекта».
7. В чем заключается самоорганизации и кооперация в компании.
8. Охарактеризуйте структуру памяти агента и основные фазы мышления.
9. В чем заключается процесс самоорганизации в мультиагентной системе.
10. Опишите архитектуру интеллектуальной системы поддержки согласованной кооперативной работы.
11. Приведите примеры применения многоагентной системы – «Структура мультимедиа-компании», «Структура оптово-розничной компании», «Структура серверного центра».

Задания для текущего контроля усвоения материала

- Дайте определение многоагентных систем (МАС).
- Охарактеризуйте современные подходы к решению распределенных задач. Приведите примеры задач, решаемых посредством агентов.
- Что Вы понимаете под искусственным интеллектом.
- Опишите психологический подход и современное развитие ИИ.
- Какие изменения имеют место в теории искусственного интеллекта.
- Охарактеризуйте основы теории агентов.
- Приведите классификацию агентов, переход от объектов к агентам.
- Опишите имеющие место архитектуры агентов, языки описания и реализации агентов.
- Дайте общую характеристику многоагентных систем.
- Охарактеризуйте основы распределенного искусственного интеллекта.
- Приведите примеры построения многоагентных систем.
- Как организовано взаимодействие между агентами многоагентных систем. Критерии и ситуации взаимодействия агентов.
- Каким образом устанавливаются базовые типы сотрудничества и соперничества.
- Что такое «Кооперация агентов».
- Способы формирования различных архитектур многоагентных систем в процессе взаимодействия агентов.
- Опишите организации: естественные и искусственные. Понятие организации и его роль в создании многоагентных систем.
- Приведите классификацию организаций.
- Охарактеризуйте деятельность агента и ее моделирование.
- В чем основы психологической теории деятельности и теории действия.
- Роль обязательств в формировании коллективных действий агентов.
- Коммуникация в многоагентных системах.

- Основы семиотики. Прикладная семиотика. Эволюционная семиотика.
- Базовые функции коммуникации агентов.
- Модели коммуникации агентов.
- Теория и средства коммуникации, базирующиеся на речевых актах.
- Использование XML для коммуникации агентов.
- Протоколы общения агентов
- Программирование многоагентных систем на различных платформах
- Проектирование многоагентных систем и искусственных организаций.
- Восходящий и нисходящий подходы к проектированию многоагентных систем. Эволюционное и коэволюционное проектирование многоагентных систем. Проектирование многоагентных систем на основе обобщенного объектно-ориентированного подхода.

Задания для итогового контроля остаточных знаний по дисциплине.

Разобрать представленные ниже статьи, и по аналогии предложить самостоятельно в процессе прохождения экзамена постановки задач и алгоритмы их решения с учетом разобранных Вами литературных источников - Лаборатория искусственных обществ <http://www.artsoc.ru/>.

1. Использование модели окружающей среды для координации намерений агентов при решении сложных задач.
2. Конкуренция в киноиндустрии: теоретико-игровой метод в сравнении с агент-ориентированной моделью.
3. Объединение вычислительной динамики подвижности и агент-ориентированного моделирования: новый подход к планированию эвакуации.
4. Агент-ориентированная модель политики фирмы в области корпоративной социальной ответственности.
5. Агент-ориентированная модель политического решения.

Задания для самостоятельной работы студентов

- Перечень вопросов по дисциплине для самостоятельного изучения [11].
- Определение и свойства программного агента. Историческая справка. Основные понятия. Архитектура агентов.
- Взаимодействие агентов. Протоколы обмена.
- Жизненный цикл агента. Системы Agentuilder и Bee-gent.
- Стандартные протоколы обмена данными и знаниями.
- Языки KIF и KQML. Синтаксис KIF и KQML.
- Онтологии для разделения знаний. Проекты ONTOWEB и ONTOBROKER.
- Онтологии и онтологические системы. Системы и средства представления онтологических знаний. Онтологии в Интернет.
- Многоагентные системы в Интернет. Проекты Autonomy и Webcompass. (ONTO)2 – агент поиска и выбора онтологий.

Список рекомендуемой литературы

1. Barbuceanu, M. S. Fox, Conflict Management with a Credibility/Deniability Model, University of Toronto.
2. Barbuceanu, M. S. Fox, Integrating Communicative Action, Conversations and Decision Theory to Coordinate Agents, University of Toronto.
3. O.Belakhdar and J.Ayel. Meeting Scheduling: an Application for Protocols Driven Cooperation. In: *Proceedings of the first International Conference "The Practical Application of Intelligent Agents and Multi-Agent Technology" (PAAM 96)*, London, 1996, p.25-44.
4. D.Chess et al. Itinerant Agent for Mobile Computing. Internal IBM. Technical Report, published in IEEE Personal Communications Magazine, 1995.
5. B.Dunin-Keplicz and J.Treuer. Compositional Formal Specification of Multi-Agent System In: *Intelligent Agents. ECAI-94 Workshop on Agent Theories, Architecture and Languages*. Amsterdam, The Netherlands, August 8-9, 1994, (Eds. M.J.Wooldridge and N.R.Jennings). Proceedings. SpringerVerlag: 102-117, 1994.
6. E.A.Emerson and J.Y.Halpern. 'Sometimes' and 'not never' revisited: on branching time versus linear time temporal logic. *Journal of the ACM* , 33(1), 1986.
7. I.A.Ferguson. Integrated Control and Coordinated Behaviour: A case for Agent Models. In: *Intelligent Agents. ECAI-94 Workshop on Agent Theories, Architecture and Languages*. Amsterdam, The Netherlands, August 8-9, 1994, (Eds. M.J.Wooldridge and N.R.Jennings). Proceedings. SpringerVerlag: 203-218, 1994.
8. R.E.Fikes and N.Nilsson. STRIPS: A new Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence*, 5(2): 189-208, 1971.
9. M.P.Georgeff, and A.S.Rao. Formal model and Decision Procedures for Multi-Agent Systems. Technicalnote 61, AustralianArtificialIntelligenceInstitute, 1995.

- 10.M.P.Georgeff and A.S.Rao. BDI Agents: From Theory to Practice. In Proceedings First International Conference on Multi-Agent Systems (ed. V. Lesser). AAAI Press/ The MIT Press, p. 312-319, 1995.
- 11.Jun Huang, N.R.Jennings and J.Fox. An Agent Architecture for Distributed Medical Care. In: *Intelligent Agents. ECAI-94 Workshop on Agent Theories, Architecture and Languages*. Amsterdam, The Netherlands, August 8-9, 1994, (Eds. M.J.Wooldridge and N.R.Jennings). Proceedings. Springer Verlag:219-232, 1994.
- 12.Luce and Raiffa H. Games and Decisions, John Wiley & Sons, Inc, New York, 1957.
- 13.P.Maes. Agent that Reduce Work and Information Overload. In: Communication of the ACM, v.37, No.7, July 1994, p. 30-40.
- 14.J.Malec. A Unified Approach to Intelligent Agency. In: *Intelligent Agents. ECAI-94 Workshop on Agent Theories, Architecture and Languages*. Amsterdam, The Netherlands, August 8-9, 1994, (Eds. M.J.Wooldridge and N.R.Jennings). Proceedings. SpringerVerlag: 233-244, 1994.
- 15.T.Magedanz. Intelligent Agent: State-of-the-Art and Potential Application In: 1 International Workshop on High Speed Networks and Open Distributed Platforms (Participants Pre-proceedings). St.Petersburg, 1995.
- 16.D.Moffat and N.H.Frijda. Where there's a Will there's an Agent. n: *Intelligent Agents. ECAI-94 Workshop on Agent Theories, Architecture and Languages*. Amsterdam, The Netherlands, August 8-9, 1994, (Eds. M.J.Wooldridge and N.R.Jennings). Proceedings. SpringerVerlag:245-259, 1994.
- 17.MorlyR., and Schelberg C. An Analysis of Plant Specific Dynamic Scheduler. In: Proceedings of the NSF Workshop of Dynamic Scheduling. Cocoa Beech, Florida, USA, 1993.
- 18.J.P.Muller, M.Pishel, and M.Thiel. Modelling Reactive Behaviour in Vertically Layered Agent Architectures. In: *Intelligent Agents. ECAI-94 Workshop on Agent Theories, Architecture and Languages*. Amsterdam, The Netherlands,

- August 8-9, 1994, (Eds. M.J.Wooldridge and N.R.Jennings). Proceedings. SpringerVerlag: 261-276, 1994.
19. Myerson R. Game Theory: Analysis of Conflict. Harward University Press, Cambridge, Massachusetts. 1991.
 20. Neiman D., and al. Exploiting Meta-Level Information in a Distributed Scheduling System., In: Proceedings of 12th National Conference on Artificial Intelligence, Seattle, WA, USA, 1994.
 21. R.G.Smith. A Framework for Distributed Problem Solving. UMI Research Press, 1980.
 22. G Smith The Contract Net Protocol: High Level Communication and Control in a Distributed Problem Solver, IEEE transactions on computers Vol 29, 1980.
 23. Wang, L. Liao, Constraint Based Framework for Multi-agent Coordination, in proceedings of EXPERSYS-96.
 24. Weerasooriya, A. S. Rao, and K. Ramamohanarao, "Design of a concurrent agent-oriented language," Tech. Rep. 52, Australian Artificial Intelligence Institute, Melbourne, Australia, Oct 1994.
 25. A. Walker and M. Wooldridge. Understanding the emergence of Conventions in Multi-Agent Systems. 1995.
 26. M. Wooldridge and N. Jennings. The cooperative problem solving process: A formal model. Technical report, Department of Computing, Manchester Metropolitan University, Chester St., Manchester M1 5GD, UK, 1994.
 27. M. Wooldridge and N. Jennings. Towards a Theory of Cooperative Problem Solving
 28. M. Wooldridge and N.R. Jennings. Agent Theories, Architectures, and Languages: A Survey. In: *Intelligent Agents. ECAI-94 Workshop on Agent Theories, Architecture and Languages*. Amsterdam, The Netherlands, August 8-9, 1994, (Eds. M.J.Wooldridge and N.R.Jennings). Proceedings. SpringerVerlag: 3-39, 1994.
 29. Zlotkin, J. S. Rosenschein, Mechanisms for Automated Negotiation in State Oriented Domain, Journal of Artificial Intelligence Research 5, 1996.

- 30.Багиров Т. К. Автоматизированная система анализа защищенности корпоративной вычислительной сети на основе многоагентного подхода: дис. канд. техн. наук: 05.13.19. Уфа: РГБ, 2007.- 204с.
- 31.Варшавский В.И., Поспелов Д.А.. Оркестр играет без дирижера. М: Наука 1984 .
- 32.Гаврилова Т. А. Онтологический подход к управлению знаниями при разработке корпоративных информационных систем.- Новости искусственного интеллекта. 2003. № 2. с. 24–30.
33. Городецкий В.И., Лебедев А.Н.. Планирование и составление расписаний автоматическое удовлетворение ограничений на временную структуру процесса. “Проблемы информатизации”, N3- 4, 1994, стр. 49-55.
- 34.Котенко И. В. Многоагентные технологии анализа уязвимостей и обнаружения вторжений в компьютерных сетях. - Конфидент. 2004. Часть 1, № 2., часть 2, № 3. - с.72–76. с.78–82.
- 35.Рассел С., Норвиг П. Искусственный интеллект: современный подход. - М.: Вильямс, 2007.-1408с.
- 36.Тарасов В.Б. От многоагентных систем к интеллектуальным организациям. М., 2002.- 352с.
- 37.Хайкин С. М. Нейронные сети: полный курс. 2-е изд. Издательский дом Вильямс, 2006. -1104с.

Ресурсы сети Интернет

- 38.Гаврилова Т. А.Программа курса «Интеллектуальные информационные системы» СПбПУ, кафедра интеллектуальные технологии в проектировании, 25с. - <http://www.spbstu.ru/>.
- 39.Городецкий В.И., Грушинский М.С., Хабалов А.В. Многоагентные системы (обзор).- URL: <http://do.gendocs.ru/docs/index-234450.html>
- 40.Котенко И. В., Уланов А. В. Кооперативная работа команд агентов при защите от сетевых атак нарушения доступности. www.comsec.spb.ru.

- 41.Тарасов В.Б. Агенты, многоагентные системы, виртуальные сообщества: стратегическое направление в информатике и искусственном интеллекте, URL - <http://www.raai.org/library/ainews/1998/2/tarasov.zip>
- 42.V. A. Vittikh. Multi - agent systems for modeling of self-organization and co-operation processes//URL - <http://www.cs.brandeis.edu/dept/faculty/mataric>.
- 43.Project of multi-agent technology in difficult systems // Open University of the Netherlands, URL - <http://www.ouh.nl/>
- 44.Устюжанин А.Е. Многоагентные интеллектуальные системы: Учебный курс (лекции, лабораторные и курсовые работы, презентации, примеры выполненных курсовых работ). МИФИ, <http://window.edu.ru>.