ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ имени академика С.П. КОРОЛЕВА»

Н.М.БОРГЕСТ, Е.В.СИМОНОВА

ОСНОВЫ ПОСТРОЕНИЯ МУЛЬТИАГЕНТНЫХ СИСТЕМ, ИСПОЛЬЗУЮЩИХ ОНТОЛОГИЮ

Утверждено Редакционно-издательским советом университета в качестве учебного пособия

САМАРА
Издательство СГАУ
2009

УДК СГАУ : 004.9(075)
ББК 32.97
Б 82

Рецензенты: д-р техн. наук, проф. А.Н. Коварцев,
д-р техн. наук, проф. С. В. Смирнов

Боргест Н.М.
Б 82 **Основы построения мультиагентных систем, использующих онтологию**: учеб. пособие / *Н.М.Боргест, Е.В.Симонова.* – Самара: Изд-во Самар. гос. аэрокосм. ун-та, 2009. – 80 с.

ISBN 978-5-7883-0690-2

В учебном пособии описаны основы построения мультиагентных систем, использующих онтологию, в различных областях производственной сферы. Разработано на кафедре конструкции и проектирования летательных аппаратов совместно с кафедрой информационных систем и технологий. Лабораторный практикум базируется на закупленном в рамках ИОП программном обеспечении, предоставленном компанией Magenta Development.

Предназначено для студентов, обучающихся по специальности 220305 – «Автоматизированное управление жизненным циклом продукции», и может быть использовано для других специальностей (направлений подготовки) при изучении онтологий или мультиагентных технологий.

УДК СГАУ : 004.9(075)
ББК 32.97

ISBN 978-5-7883-0690-2

Учебное издание

Боргест Николай Михайлович,
Симонова Елена Витальевна

ОСНОВЫ ПОСТРОЕНИЯ МУЛЬТИАГЕНТНЫХ СИСТЕМ, ИСПОЛЬЗУЮЩИХ ОНТОЛОГИЮ

Учебное пособие

Редактор Спиридонова И.И.
Доверстка Спиридонова И.И.

Подписано в печать 25.03.09 г. Формат 60x84 1/16.
Бумага офсетная. Печать офсетная.
Усл. печ. л. 4,75. Тираж 120 экз. Заказ 50 . Арт. С- 9/2009

Изд-во Самарского государственного аэрокосмического университета.
443086 Самара, Московское шоссе, 34.

СОДЕРЖАНИЕ

ПРЕДИСЛОВИЕ

Содержание учебного пособия соответствует разделам рабочей программы по дисциплине «Онтология производственной сферы», составленной на основании Государственного образовательного стандарта высшего профессионального образования по специальности 220305 – «Автоматизированное управление жизненным циклом продукции».

В **главе 1** представлены сведения о назначении и областях применения мультиагентых технологий, принципах построения мультиагентных систем, методах и средствах построения онтологии для описания процессов динамического распределения ресурсов в открытых производственных системах.

Глава 2 посвящена рассмотрению методологии и инструментальных средств построения открытых мультиагентных систем поддержки принятия решений, разработанных в компании Magenta Development.

В **главе 3** описываются цели, задачи и содержание лабораторного практикума по курсу «Онтология производственной сферы».

Глава 4 содержит описание примеров индивидуальных заданий, выполненных студентами СГАУ, в различных областях производственной сферы.

В учебном пособии содержатся сведения, недостаточно освещенные в учебной литературе, а также заложены новые методики преподавания, активизирующие самостоятельную работу студентов.

ВВЕДЕНИЕ

Мультиагентная технология – это новая программная технология, которая предназначена для поддержки принятия решений в современном сложном и быстро меняющемся мире. Мультиагентные технологии могут работать как в бизнесе, так и в других областях: на предприятиях, в государственном управлении, здравоохранении и социальной сфере, и даже при анализе почты на персональном компьютере.

К числу задач, которые могут быть эффективно решены с использованием мультиагентных технологий, относятся разнообразные задачи планирования ресурсов в реальном времени; задачи понимания смысла текстов на естественном языке (например, поиска страниц в Интернете, систематизации архивов документов или обработки почты); анализа данных для обнаружения скрытых знаний при принятии решений; обучения компьютерных систем путем выявления предпочтений и построения моделей поведения пользователя и ряд других. Все эти задачи отличает априорная неопределенность и высокая динамика, наличие многих взаимосвязей и взаимозависимостей, высокая вычислительная сложность, необходимость учета множества индивидуальных факторов, нелинейность поведения, зависимость решения от истории процессов или даже момента времени и т.п.

В отличие от традиционных систем, в которых решение находят с помощью централизованных, последовательных и детерминированных алгоритмов, в мультиагентных системах решение достигается в результате распределённого взаимодействия множества агентов – автономных программных объектов, нацеленных на поиск возможно не оптимального, но наилучшего из возможных решений на каждый момент времени. Если найденный агентом лучший вариант уже забронирован другим агентом, агенты оказываются способны выявить конфликт и разрешить его путём переговоров, в ходе которых достигается компромисс, отражающий временное, и, как правило, неустойчивое равновесие (баланс) их интересов.

В целом, мультиагентные системы (или агентно-ориентированное программирование) являются следующим шагом в развитии объектно-ориентированного программирования (ООП) и интегрируют в себе достижения последних десятилетий в сфере искусственного интеллекта, параллельных вычислений и телекоммуникаций.

Мультиагентные технологии обеспечивают высокую персонализацию решений для пользователя – знания о каждом пользователе могут быть описаны в системе через набор соответствующих концептов и отношений. В результате агент пользователя может следовать установленным для него специальным целям, предпочтениям и ограничениям. Это позволяет агентам учитывать индивидуальные для каждого пользователя сведения и принимать решения адресным образом, с учетом всей специфики каждого пользователя, действуя от его имени и по его поручению. Агенты достигают взаимовыгодного решения путем переговоров, направленных на достижение компромисса.

1. МУЛЬТИАГЕНТНЫЕ ТЕХНОЛОГИИ ОПЕРАТИВНОЙ ОБРАБОТКИ ИНФОРМАЦИИ ДЛЯ ПОДДЕРЖКИ ПРОЦЕССОВ ПРИНЯТИЯ РЕШЕНИЙ

Открытый характер современного информационного общества и глобальной рыночной экономики приводит к необходимости использовать новые методы и средства организации и управления, направленные на более качественное и эффективное удовлетворение ***индивидуальных запросов*** пользователей.

1.1. Проблема управления процессами динамического распределения ресурсов в открытых системах

В открытых производственных системах возникают задачи управления, связанные с динамическим перераспределением ресурсов при постоянно изменяющихся спросе и предложении. Традиционно проблема принятия решений рассматривается и решается в случае, когда все ресурсы и потребности в этих ресурсах известны заранее и не меняются в ходе принятия решений. Но в современных условиях спрос на ресурсы и предложение ресурсов подчиняются весьма изменчивым и непредсказуемым процессам, а сами ресурсы и заказы могут появляться и исчезать в непредвиденные заранее моменты времени.

Примеры областей, где требуется постоянное динамическое распределение ресурсов:

- Производственная логистика – определение места и времени нахождения материалов или компонентов на производственных мощностях (сборочные устройства, конвейеры, металлообрабатывающие станки и т.д.) в условиях неопределенности, создаваемой постоянными изменениями спецификаций производимых продуктов и поведением поставщиков и потребителей.
- Планирование работы персонала – распределение задач для персонала в условиях быстро развивающегося рынка.
- E-коммерция – распределение доступных товаров и услуг (предложение) в соответствии с требованиями клиентов (спрос), когда клиенты и/или поставщики неожиданно появляются или покидают торговую площадку.
- Управление знаниями – распределение записей по кластерам в ситуациях частого обновления баз данных.
- Понимание текста – размещение (приписывание) смыслов (значений) словам в процессе непрерывного диалога между человеком и машиной.

Один из новых подходов к управлению предприятиями связан с построением сетевых организаций (*Networking Organizations*), подразделения которых могут рассматриваться как автономные предприятия, напрямую взаимодействующие между собой или с внешними организациями. В отличие от традиционных предприятий, сетевая организация по своему устройству является открытой, способной приобретать новые знания и гибко адаптировать свою структуру или принципы функционирования в зависимости от изменений в среде. Для

оперативной реакции на эти изменения рассматриваемые предприятия должны непрерывно осуществлять идентификацию потребностей и возможностей на рынке и принимать согласованные решения по динамической реконфигурации собственных производственных, финансовых, кадровых и других ресурсов, своевременно внедряя новые технологии, обновляя номенклатуру продукции, расширяя круг партнерских связей и т.д.

Типичные примеры изменений в среде, вызывающие необходимость заново идентифицировать потребности и возможности предприятия, могут быть связаны с появлением нового выгодного заказа, для исполнения которого недостаточно собственных ресурсов предприятия; возникновением на рынке новых ресурсов, обладающих большей эффективностью для предприятия; отзывом принятого ранее и уже запущенного в производство заказа; выходом из строя части имеющихся ресурсов, а также изменением критериев принятия решений.

В этих условиях классические подходы к принятию решений оказываются недостаточно эффективными и должны быть дополнены методами и средствами для мониторинга изменений в среде, а также частого и быстрого согласованного перераспределения ресурсов.

Одно из перспективных направлений реформ управления современными предприятиями связано с холистическим (целостным) подходом и переходом от закрытых централизованных иерархических структур с жесткими связями к открытым сетевым организациям с гибкими связями [4,5]. При этом кардинальное изменение претерпевает характер взаимодействия между всеми подразделениями предприятия или лицами, принимающими решения, и на смену классическому «бюрократическому» управлению с выдачей команд «сверху-вниз» приходят переговоры, построенные на принципах взаимодействия равных партнеров, где при необходимости каждый может коммуницировать с каждым и структура такого взаимодействия заранее не предписана и никак не ограничена.

Дальнейшее развитие холистического подхода приводит к появлению организаций, в которых не только подразделения или люди, но и любые материальные ресурсы (станок, транспорт, готовая деталь и т.д.) могут быть условно организованы как автономные бизнес-единицы, имеющие собственные цели и ограничения, возможности и потребности на виртуальном рынке.

Финансовые, материальные, кадровые и другие ресурсы такого предприятия могут постоянно реконфигурироваться, пополняться или наоборот сокращаться, изменяясь и адаптируясь к решаемым задачам и ситуации на рынке, что кардинально меняет требования к программным средствам для поддержки процессов принятия управленческих решений.

В этих условиях традиционные программные системы по управлению ресурсами предприятий (*Enterprise Resource Planning*), разработанные такими компаниями, как SAP, BAAN, Navision и другие, оказываются недостаточно эффективными, поскольку любое изменение корпоративных знаний или модификация схем принятия решений в этих системах представляют собой весьма сложные и трудоемкие процессы и требуют высокой квалификации исполнителей, что делает разработку и эксплуатацию рассматриваемых систем крайне дорогостоящими, а также не позволяет реализовать индивидуальный подход к решению задач каждого конкретного пользователя.

1.2. Мультиагентные системы

Новый подход к решению задачи оперативной обработки информации в процессах принятия решений может быть основан на применении ***мультиагентных технологий***, получивших интенсивное развитие в последнее десятилетие на стыке методов искусственного интеллекта, объектно-ориентированного программирования, параллельных вычислений и телекоммуникаций [1-3].

Мультиагентные системы (МАС) объединяют три технологии: распределенный ИИ (*distributed AI*), распределенные решатели задач (*distributed problem solving*), параллельные вычисления.

В своем развитии МАС прошли две фазы:

1. 1977 г. – начало 90-х гг. – создание ***«смышленых агентов» (smart agents)***. Работы этого времени были преимущественно теоретическими: в этих работах изучались вопросы о том, как агенты могут договариваться (теперь это целый раздел под названием negotiations), координировать свои действия, декомпозировать и распределять задачи и т.д.

2. 90-е гг. – до настоящего времени – создание ***интеллектуальных агентов (intelligent agents)***. Работы этого периода характеризуются практическим применением, а в фокусе внимания исследователей теперь находятся вопросы архитектуры, языки представления сценариев, средства проектирования агентов.

1.2.1. Общая характеристика интеллектуальных агентов

В основе мультиагентной технологии лежит понятие ***«агента»*** – программного объекта, способного воспринимать ситуацию, принимать решения и коммуницировать с подобными себе объектами, динамически устанавливая с ними связи. Отличительными свойствами агента являются [2]:

- **Автономность** – способность функционировать без прямого вмешательства людей или компьютерных средств и при этом осуществлять самоконтроль над своими действиями и внутренними состояниями;
- **Общественное поведение** (*social ability*) – способность «разумно» общаться с другими объектами посредством создания сообщений, посылки их другим агентам или людям, способности интерпретировать входящие сообщения, обрабатывая их определенным образом;
- **Реактивность** (*reactivity*) – способность воспринимать состояние среды (физического мира, человека – через пользовательский интерфейс, совокупности других агентов, сети Internet, или сразу всех этих компонентов внешней среды);
- **Целенаправленная активность** (*pro-activity*) – способность агентов не просто реагировать на стимулы, поступающие из среды, но и функционировать на основе заданных ему целей, проявляя инициативу.

Мир агентов – сообщество (*swarm*) – это группа агентов, участвующих в переговорах друг с другом с целью выполнения поставленной им задачи.

Эти возможности кардинально отличают мультиагентные системы (МАС) от существующих «жестко» организованных систем, обеспечивая им такое важное свойство как способность к самоорганизации. При этом агенты могут действовать от имени и по поручению лиц, принимающих решения, и на основе данных им полномочий в автоматическом режиме вести переговоры, находить варианты решений и согласовывать свои решения друг с другом.

Рассмотрим простой пример: агент-брокер, помогающий человеку или предприятию сделать покупку некоторого товара или оборудования (рис. 1) [3].

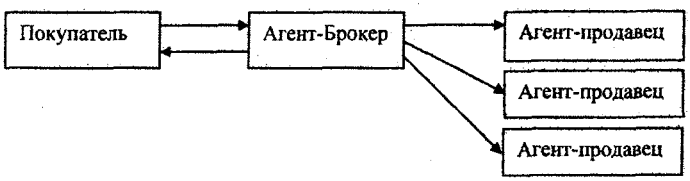


Рис. 1. Схема взаимодействия агентов

Здесь покупатель делает запрос брокеру, брокер связывается с продавцами, поставщиками или непосредственно производителями, осуществляет выбор товара, связывает покупателя с продавцом для оплаты товара. Все эти взаимодействия происходят через переговоры между агентами.

Агенты должны быть способны к принятию решений в условиях неопределенности ситуации, действовать при отсутствии полной информации, хотя бы и в какой-либо узкой области. Как правило, агенты скорее обучены, чем запрограммированы для выполнения конкретной работы. Наиболее продвинутые версии агентов могут учиться на собственном опыте и иметь отличительные черты индивидуальности.

Классификация агентов приведена в табл. 1 [3].

Таблица 1

Классификация агентов

Типы агентов	Простые	Смышленые (Smart)	Интеллектуальные (Intelligent)	Действительно интеллектуальные (Truly)
Автономное выполнение	+		+	+
Коммуникация с другими агентами и пользователями	+	+	+	+
Слежение за окружением	+	+	+	+
Способность использования символов и абстракций		+	+	+

Окончание табл. 1

Типы агентов	Простые	Смышленые (Smart)	Интеллектуальные (Intelligent)	Действительно интеллектуальные (Truly)
Использование знаний предметной области		+	+	
Цели и поведение			+	+
Адаптивное обучение из среды			+	+
Реакция на ошибки (обработка ошибок)				+
Реальное время				+
Естественные языки				+

1.2.2. Сети потребностей и возможностей (ПВ-сети)

В качестве методической основы для создания открытых мультиагентных систем оперативной обработки информации для поддержки процессов принятия решений (OMAC ППР) рассматривается модель ***сети потребностей и возможностей (ПВ-сеть)***. Эта модель базируется на ***холистическом подходе***, в рамках которого предприятие декомпозируется до уровня сети отдельных автономных «физических сущностей» (станки, транспортные средства, детали, материалы и т.д.), каждая из которых получает своих агентов потребностей и возможностей [4,5].

Предприятие можно «расщепить» до уровня отдельных независимых ресурсов (человек, компьютер, готовая деталь и т.д.), каждый из которых сам по себе может быть некоторой автономной бизнес-единицей (холоном). В таком подходе не только группы людей (или даже один человек), но и неживые физические (или даже абстрактные) объекты знаний условно могут являться независимыми «компаниями» (например, компьютер за каждый час времени может получать условную арендную плату, которая может идти на ремонт его частей, осуществление профилактики и т.д.).

Такое предприятие может рассматриваться как набор некоторых ресурсов (рис. 2). Ресурсы призваны перерабатывать разного рода заказы. Заказы и ресурсы при реализации проекта могут быть динамически связаны между собой, образуя в каждый момент времени некоторую внутреннюю сеть предприятия, например, ресурс некоторого исполнителя для реализации некоторого заказа может быть временно связан с ресурсом оборудования, ресурсом помещения, ресурсом времени, ресурсом знаний и т.д.

Возникновение непредусмотренных изменений во внешней среде (появление нового большого проекта или, наоборот, останов уже начатого проекта; появление нового исполнителя или более производительного пакета программ; поломка оборудования; пересмотр целей или критериев работы предприятия и т.д.) должно немедленно вызывать цепочки пересмотра принятых ранее решений по распределению ресурсов предприятия по принятым ранее заказам, а также, возможно, по привлечению новых ресурсов или выведению из работы некоторых других неэффективных ресурсов. Примеров таких изменений,

например, в авиационной отрасли предостаточно: когда меняется конъюктура, собственник предприятия, менеджмент, субподрядчики, переналаживается производство под новый самолет и т.п.

Все заказы и ресурсы предприятия взаимодействуют, вступая в связи между собой, на основе своих возможностей и потребностей. В этом контексте можно считать, что каждый ресурс или заказ раздваивается на противоположности (рис. 2): возможности и потребности. Например, определенный рабочий может иметь возможность изготовить деталь в заданные сроки, но при этом имеет потребность в станке. Если же заказ слишком сложен, у него появится «потребность» найти для себя со-исполнителей (субподрядчиков) с соответствующими знаниями и умениями, способных выполнять данную работу. Причем они должны быть свободны в заданный момент времени, быть согласными на оплату не выше заданной и т.д.

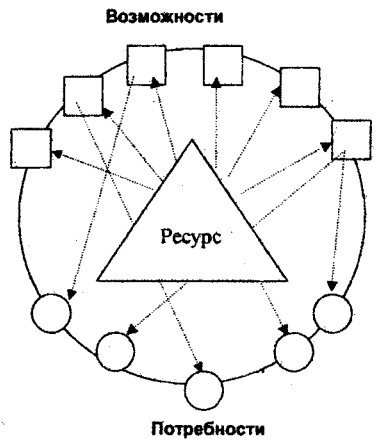


Рис. 2. Возможности и потребности ресурса

В результате, «характеры» заказов и ресурсов также оказываются прямо противоположными: ресурсы более консервативны и должны стремиться быть максимально использованными и просуществовать как можно больший срок, а заказы, наоборот, более активны и стремятся как можно быстрее реализоваться с наилучшими характеристиками, в частности, как можно меньше использовать ресурсы, чтобы меньше потратить денег на расчеты с ними.

Таким образом, заказы и ресурсы, возможности и потребности становятся новыми базовыми элементами устройства предприятия, состоящими в диалектических отношениях между собой. Единство и борьба этих противоположностей реализуется через постоянный поиск соответствия между ними (далее – ***«матчинг» потребностей и возможностей***), обусловленный заданными индивидуальными критериями для каждой из сторон (качество, деньги, время и т.д.). Суть этой операции для каждой из сущностей состоит в том, чтобы обна-

ружить все имеющиеся на внутреннем рынке предприятия альтернативные возможности или потребности и принять решение об установлении связи, которая отвечает улучшению заданного критерия или набора критериев при условии, что и противоположный партнер (ресурс или заказ) согласен на установление такой связи.

Как следствие, два ресурса могут конкурировать за один заказ, или, наоборот, кооперироваться, когда работа не по силам каждому из них в одиночку. В результате установления связи противоположности временно перестают конкурировать и приходят в «единство» – до тех пор, пока заказ не будет выполнен и не появится новая материальная сущность (ресурс) со своими возможностями и потребностями. Однако, если в ходе работы по проекту выясняется необходимость и возможность поменять исполнителя или заменить оборудование – потребности и возможности должны договориться о разрыве одних и установлении других связей.

Предприятие, построенное на основе динамического распределения ресурсов и заказов, будем называть «***сетью потребностей и возможностей***» или, для краткости, «***ПВ-сетью***».

Мультиагентная система поддержки принятия решений может рассматриваться как система, состоящая из агентов возможностей и потребностей, ***соревнующихся или кооперирующихся*** между собой, в зависимости от ситуации, с целью дать возможность системе выполнить поставленную задачу.

При этом главной особенностью предлагаемого подхода становится ***самоорганизация заказов и ресурсов***, при которой вместо централизованного решения задачи и построения полностью оптимального плана действий, каждый заказ и каждый ресурс должен самостоятельно принять решение, какой из возможных вариантов взаимодействия ему подходит, и своевременно изменить принятое ранее решение, если находится вариант для его улучшения.

Под самоорганизацией при этом понимается возможность системы автономно видоизменять уже существующие и/или устанавливать новые связи между ее компонентами с целью повышения значения критерия эффективности ее существования или восстановления после повреждения. В контексте распределения ресурсов любое автономное изменение связи ***возможность-потребность*** рассматривается как шаг в самоорганизации. Такая самоорганизация важна, когда спрос на ресурсы и предложение ресурсов подчиняются весьма изменчивым и непредсказуемым процессам, а сами ресурсы и заказы могут появляться и исчезать в заранее непредвиденные моменты времени. Что, собственно, и происходит в реальной жизни и производственной сфере, моделирование процессов в которой является целью многих исследований и данного пособия, в том числе.

1.2.3. Модель реализации ПВ-сети

При построении ПВ-сети в качестве ***базовых сущностей*** предлагается использовать агентов потребностей и возможностей, получающих способность взаимодействовать и вступать в отношения между собой [4,5].

В качестве ***основных отношений*** при этом выделяются:

- R1 – структурное отношение потребность «a» порождается возможностью «b»;
- R2 – структурное отношение потребность «a» есть часть потребности «b»;
- R3 – отношение соответствия («матчинга») между возможностями и потребностями (потребность «a» может соответствовать возможности «b»);
- R4 – отношение установленной связи (потребность «a» бронирует возможность «b»);
- некоторые другие отношения, спектр которых может дополняться.

Для принятия решений агентам необходимы ***правила***, определяющие возможность достижения поставленных целей. В частности, агент возможности должен узнать о наличии той или иной потребности и сопоставить свои параметры с параметрами этой потребности. Если параметры удовлетворяют условию «матчинга», то агент возможности может вступить в переговоры с агентом потребности, если же нет – должен искать другую потребность (и наоборот). Окончательное решение каждым из агентов при наличии нескольких альтернативных вариантов принимается на основе целей, устанавливаемых каждому из них индивидуально.

Формально будем называть порождающей ПВ-сетью множество N вида:

$$\mathbf{N=\{A, R, P, G\},}$$

где **A** – множество агентов потребностей и возможностей для заданной предметной области, **R** – множество отношений между агентами потребностей и возможностей; **P** – множество правил принятия решений и установления/разрыва связей; **G** – множество целей, заданных агентам.

Реализацией S ПВ-сети **N** будем называть конкретную конфигурацию (***сцену***) ПВ-сети, отражающую состояние агентов потребностей и возможностей открытой системы и отношения между ними в заданный момент времени. В ходе работы система переходит из состояния **S1** в состояние **S2** с помощью правил **P** и на основе целей **G**.

Пример порождающей ПВ-сети, описывающей фрагмент некоторой логистической сети транспортных перевозок, приведен на рис. 3. Здесь выделены агенты заказа (потребности) в перевозке, возможностей транспортных средств (легковое и грузовое авто, поезд), потребностей в топливе и водителе (общие для авто), аренде вагона (для поезда), а также показаны отношения создания потребности и матчинга потребностей и возможностей для бронирования.

На рис. 4 представлен пример реализации данной сети для случая, когда имеется пять перевозчиков, две заправочные станции и три водителя транспортных средств. Из рисунка видно, что в настоящий момент только одно легковое и одно грузовое авто успешно осуществили матчинг, т.е. соответствуют, например, требуемой цене и сроку доставки, и конкурируют за исходный заказ на перевозку. При этом легковое авто 1 (вариант Б) по своим критериям находит и бронирует водителя 3 и топливо 1, а грузовое авто 2 (вариант Г) – также водителя 3, но другое топливо 2. Потребность в перевозке в соответствии со своим критерием в результате выбирает вариант Б (итоговый вариант выделен жирной

линией), который оказался дешевле и обеспечивает доставку в более короткие сроки. В свою очередь, появление нового заказа или нового ресурса может изменить конфигурацию настоящей ПВ-сети.

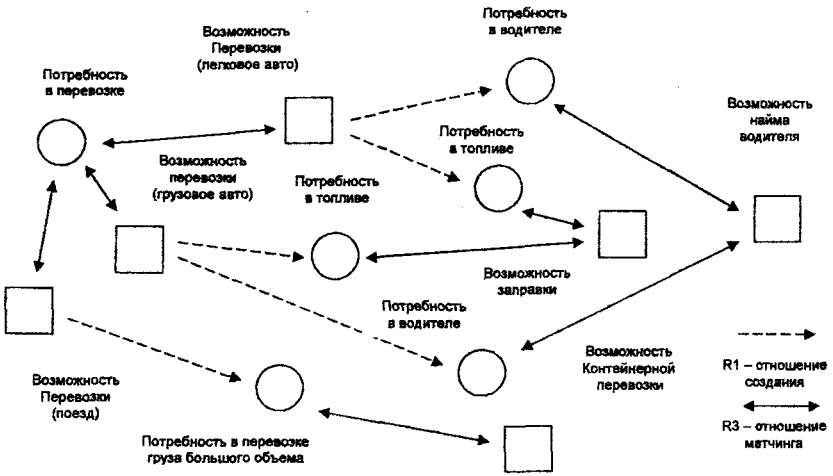


Рис. 3. Структура фрагмента порождающей ПВ-сети для задачи логистики

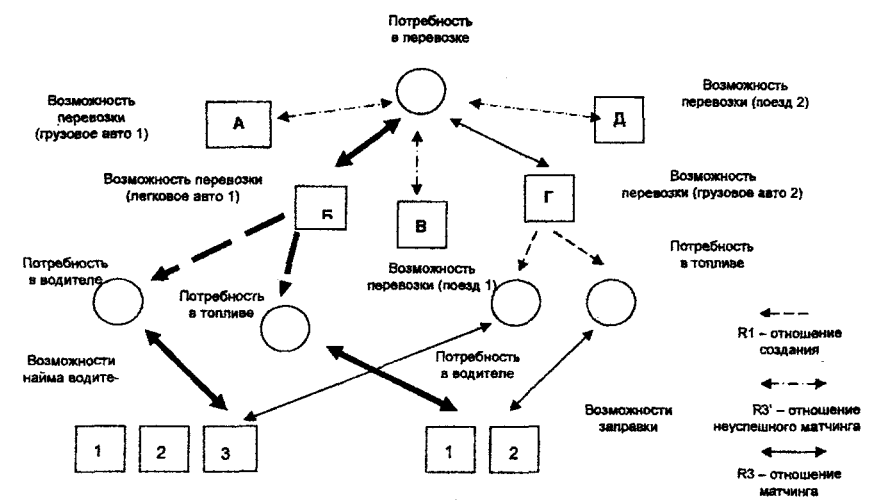


Рис. 4. Структура фрагмента реализации ПВ-сети (сцены) для задачи логистики

1.3. Принципы построения мультиагентных систем

Рассмотрим архитектуру открытых мультиагентных систем поддержки принятия решений (ОМАС ППР) на основе ПВ-сетей.

1.3.1. Основные компоненты архитектуры открытых мультиагентных систем поддержки принятия решений

В архитектуре ОМАС ППР выделены следующие компоненты:

- база знаний предметной области, включающая набор онтологий, описывающих объекты и законы предметной области, представленной в виде ПВ-сети;
- реализация ПВ-сети (сцена);
- исполняющая система;
- библиотеки расширений;
- интерфейсная система (рис. 5) [6,7].

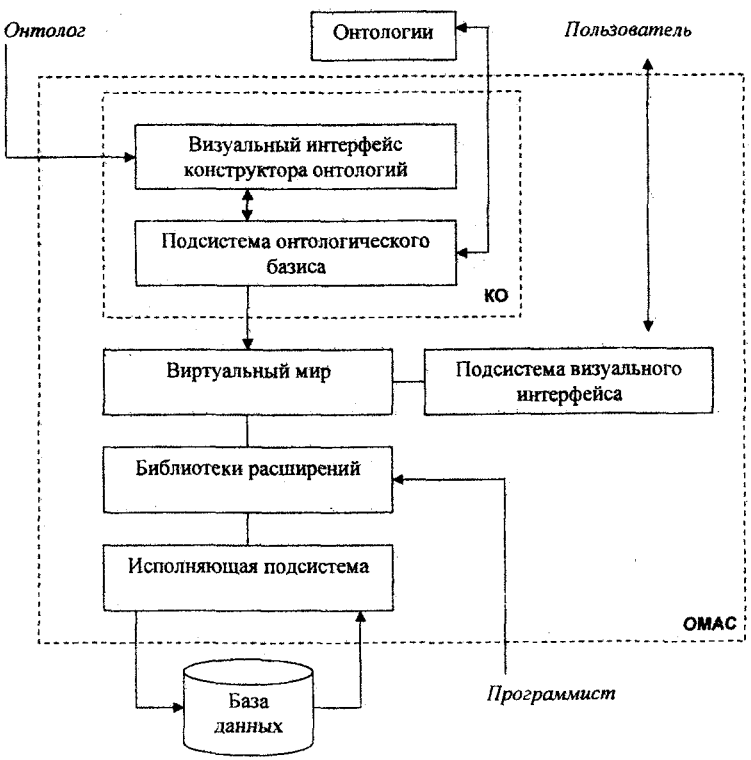


Рис. 5. *Общая архитектура мультиагентной системы*

Исполняющая система [7] представляет собой набор компонент, обеспечивающих работу агентов и их взаимодействие. В основе исполняющей системы лежит параллельная машина, обеспечивающая исполнение сценариев работы агентов с соответствующими механизмами поддержки параллельных процессов и их синхронизации. При этом каждый агент рассматривается как комбинация трех сопрограмм: восприятия, планирования и исполнения сценариев, которые могут легко переключаться между собой в зависимости от изменения ситуации в окружающей среде. Кроме того, здесь имеются компоненты, связанные с поддержанием обмена сообщениями между агентами, взаимодействия с пользователем и т.д.

Для настройки системы на заданную предметную область и работы конечного приложения необходима онтология корпоративной деятельности предприятия, которая представляет собой набор пополняемых баз знаний определенной структуры.

Онтология предприятия [6] описывает «мир» предметной области – модель фрагмента реальной среды предприятия. При этом, например, для агента заказа в онтологии логистики может быть описано, из каких частей состоит заказ, кто может производить или поставлять эти части и как можно их заказать и получить.

Используя интерфейсную компоненту, пользователь может конструировать на экране компьютера различные начальные сцены «мира». Например, в системе логистики пользователь может разместить на экране города и дороги, конвейерные и сборочные цеха производства, транспортные средства и т.д. Здесь также строится виртуальный рынок предприятия, в котором агенты получают возможность торговаться и заключать сделки по определенным правилам.

Библиотеки расширений содержат программные модули, специализированные для предметной области или отдельного приложения, например, специализированные интерфейсы, средства доступа к базам данных пользователя, генерации специализированных отчетов и т.д. Рассмотренная архитектура мультиагентной системы допускает развитие, поэтому при необходимости пользователь может модифицировать не только онтологии, но и любые модули библиотек расширения.

1.3.2. Методы и средства построения онтологий

1.3.2.1. Определение понятия онтология

«Онтология» («ontologia») в философском смысле есть учение о сущем. Для компьютерных систем онтология – это способ представления знаний о фрагменте окружающего мира, или ***знания «как они есть» («knowledge as it is»)***.

Онтологии деятельности предприятий можно конструировать как наборы физических и виртуальных (абстрактных) «миров», описывающих взаимодействие объектов [7].

Рассматриваются следующие «миры деятельности»:

- физический мир, в котором агенты имеют физический смысл, соответствующий предметной области (например, агент груза, агент перевозчика);
- виртуальный мир, в котором каждый агент выступает в абстрактной роли либо заказа, либо ресурса, между агентами ведутся переговоры, заключаются соглашения.

Онтология содержит декларативную и процедурную части. Декларативные знания определяют основные понятия предметной области (***концепты***) и множество связей (***отношений***), установленных между концептами предметной области, т.е. описывают свойства объекта и отношения, в которых он может пребывать с другими объектами. Процедурные знания определяют законы мира, правила и ограничения поведения объектов (сценарии действий).

Предлагается общая для всех миров, как физических, так и виртуальных, мета-онтология («модель мира»), включающая концепты «объект», «свойство», «процесс», «отношение» и «атрибут». Эта базовая «модель мира» предполагает, что объекты обладают свойствами; свойства выражают способность объектов вступать в процессы взаимодействия; процессы изменяют состояния объектов и отношения между ними; свойства характеризуются значениями атрибутов.

Наиболее удобной формой описания и представления знаний в онтологии является ***семантическая сеть***. Семантические сети состоят из узлов и упорядоченных отношений (связей), соединяющих эти узлы. Узлы отображают основные понятия предметной области, а связи определяют взаимоотношения между этими узлами (рис. 6).

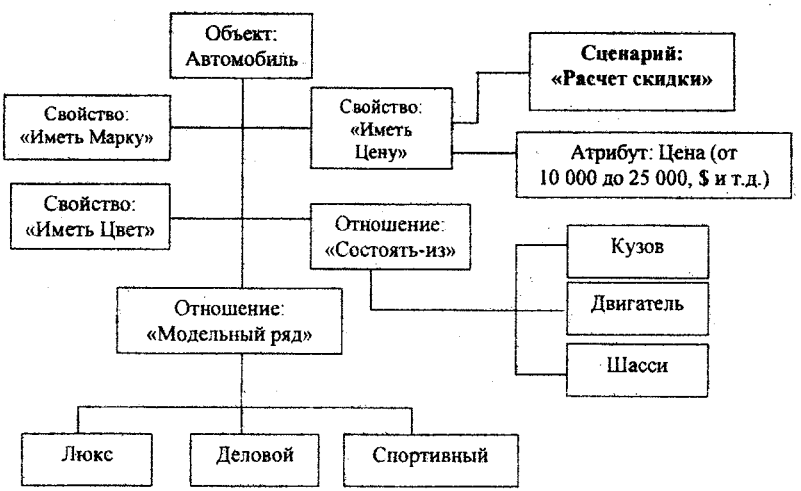


Рис. 6. Фрагмент семантической сети онтологии производства автомобилей

1.3.3. Виртуальный мир ПВ-сетей для поддержки принятия решений

Виртуальный мир для поддержки принятия решений (мир ПВ-сетей) может быть построен в форме виртуального рынка, в котором агенты потребностей и возможностей могут порождать друг друга и устанавливать отношения между собой, проводить матчинг, осуществлять денежные и иные расчеты и т.д.

На этапе разработки виртуального мира каждая сущность предметной области «раздваивается», получая в свое распоряжение агентов потребностей и возможностей. Например, ресурс «Транспорт» получает агента возможности по транспортировке, который будет заниматься поиском вариантов использования ресурса, а также агентов потребностей, обеспечивающих ресурс, например, топливом или ремонтом (рис. 7).

При конструировании виртуального мира задаются условия матчинга, цели и критерии принятия решения для агентов. Например, агент ресурса транспортного средства должен проверять, может ли он перевезти груз указанного веса, обеспечит ли он доставку в заданный срок и за заданные деньги. Определяется также схема потоковых вычислений, которую агенты могут использовать на этапе расчета значений необходимых атрибутов.

При работе в виртуальном мире необходимо управлять активностью агентов (одно– или двусторонняя активность), поддерживать составные потребности (когда один заказ требует исполнения нескольких подзаказов) и разделяемые ресурсы (когда один ресурс может работать на несколько заказов), обеспечивать сокращение числа переговоров и т.д.

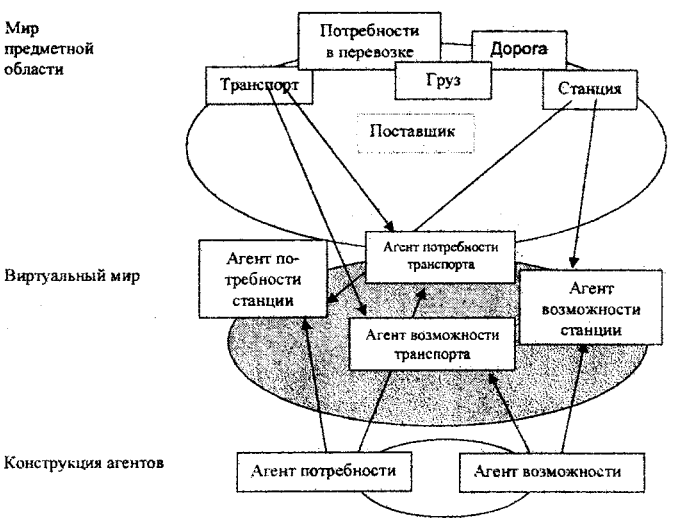


Рис. 7. Структура знаний ОМАС ППР

1.3.4. Специализированные компоненты для работы в ОМАС ППР

В качестве специализированных компонент для работы агентов в ПВ-сетях, обеспечивающих требуемую функциональность агентов и возможность поддержки процессов принятия решений, разработаны компоненты, которые описаны ниже (рис. 8) [6,7].

Матчинговый процессор (матчер) – позволяет агентам коммуницировать и устанавливать связи с другими агентами, осуществляя двустороннюю проверку условий допустимости предлагаемых вариантов решений. При этом каждый матчер отвечает за один контакт, т.е. у одного агента образуется столько матчеров, сколько различных альтернатив он исследует на рынке (рис. 9, 10).

Машина принятия решений (МПР) – позволяет динамически, непосредственно в ходе исполнения, строить таблицы допустимых вариантов решений, прошедших матчинг как стадию предварительной селекции этих решений, сортировать решения в соответствии с заданными пользователем критериями и выбирать наилучший вариант на основе актуального в текущий момент критерия или комбинации критериев.

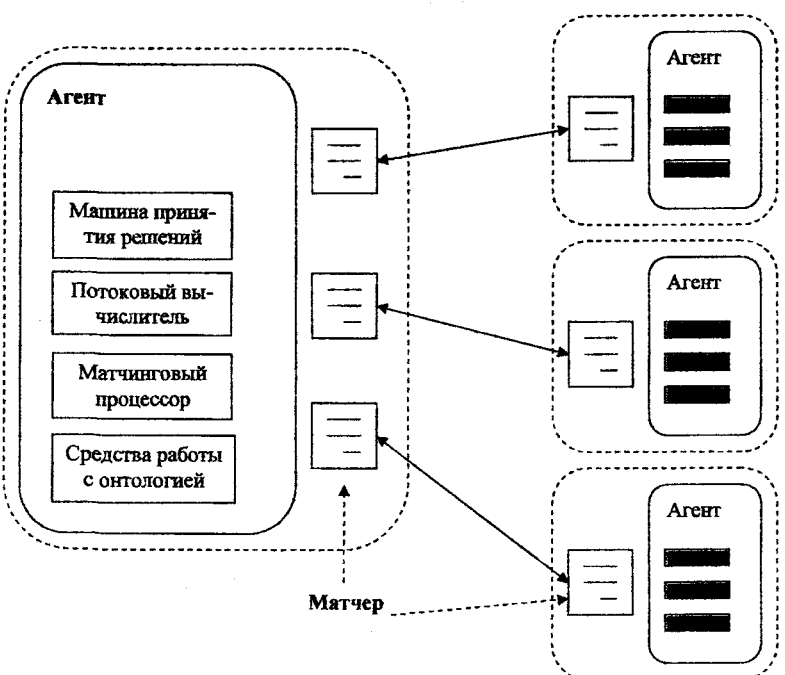


Рис. 8. Специализированные компоненты для работы в МАС

Рис. 9. Алгоритм работы матчера

Рис. 10. Алгоритм проверки условий матчинга

Блок работы с онтологией – позволяет агенту осуществлять поиск атрибутов в семантической сети онтологии и считывать свои атрибуты или необходимые правила для проверки условия матчинга и принятия решений.

Потоковый вычислитель – позволяет проводить расчеты значений атрибутов агентов, используемых в проверках при матчинге и принятии решений. При этом порядок вычисления значений атрибутов может быть заранее не предписан агенту, а определяется в ходе его работы на основе прослеживания связей между понятиями в онтологии.

Реализация рассмотренных компонент позволяет существенно сократить трудоемкость или полностью освободить пользователя от рутинного программирования взаимодействия агентов потребностей и возможностей – достаточно создать агента с указанием его целей и критериев принятия решений и описать условия матчинга, которые он должен проверять, а все остальные действия агент динамически выполняет в процессе работы системы.

1.3.4.1. Алгоритм работы машины принятия решений

Если задается несколько условий матчинга, дополнительно к условиям принятия решений указывается алгоритм сортировки альтернатив. Поддерживается два алгоритма:

- **Метод главного условия принятия решений (*Ordered mode*)** используется по умолчанию. Альтернативные варианты для принятия соответствующего решения указаны в таблице X. Строки таблицы соответствуют вариантам принятия решения, а столбцы – атрибутам, на основании которых принимается решение.

Вариант	Условие 1	Условие 2	…	Условие N
1	X_{11}	X_{12}	…	X_{1N}
2	X_{21}	X_{22}	…	X_{2N}
…	…	…	…	…
K	X_{K1}	X_{K2}	…	X_{KN}

Например, для выполнения проекта необходимо выбрать одного из трех исполнителей в соответствии с двумя критериями: максимальный опыт и минимальная продолжительность выполнения проекта.

Критерий, j / Номер альтернативы, i	Опыт работы, max	Продолжительность выполнения проекта, min
1	5	200
2	10	160
3	8	150
Максимальное значение критерия	10	200

Обозначим X_{ij} – значение критерия j, соответствующее альтернативе i. Найдем максимальное значение каждого из критериев – $X_{j\ max}$. Определим

нормализованные значения критериев Y_{ij}, где $Y_{ij} = X_{ij} / X_{j\ max}$, если направление оптимизации для критерия j – максимум; и $Y_{ij} = 1 - X_{ij} / X_{j\ max}$, если направление оптимизации для критерия j – минимум. Сформируем таблицу нормализованных значений критериев Y.

Нормализованный критерий, j / Номер альтернативы, i	Опыт работы, max	Продолжительность выполнения проекта, min
1	5/10 = 0,5	1-200/200 = 0
2	[illegible]	1-160/200 = 0,2
3	8/10 = 0,8	[illegible]
Максимальное значение критерия	10	200

Затем альтернативы сортируются по следующему правилу. Главным считается то условие, которое указано первым в списке условий принятия решения. Условия проверяются в том порядке, как они указаны в списке условий. Если $Y_{11} > Y_{21}$, то альтернатива 1 лучше альтернативы 2; если $Y_{11} = Y_{21}$, то проверяется следующее по порядку условие (Y_{12} ? Y_{22}) и т.д. Если первым указан критерий максимального опыта работы, то будет выбран исполнитель 2 (альтернатива 2). Если первым указан критерий минимальной продолжительности выполнения проекта, то будет выбран исполнитель 3 (альтернатива 3).

- **Метод многопараметрической средневзвешенной оптимизации (Balanced mode).**

Для условий принятия решений задаются весовые коэффициенты. Веса критериев обозначим вектором (w_1, w_2, ..., w_m). Сформируем таблицу нормализованных значений критериев.

Нормализованный критерий, j / Номер альтернативы, i	Опыт работы, max	Продолжительность выполнения проекта, min	Обобщенная функция цели, max
1	5/10 = 0,5	1 - 200/200 = 0	S1=0,5 · 100 + 0 · 50 = 50
2	10/10 = 1	1 - 160/200 = 0,2	[illegible]
3	8/10 = 0,8	1 - 150/200 = 0,25	S3=0,8 · 100+0,25 · 50 = 92,5
Максимальное значение критерия	10	200	
Вес критерия	100	50	

Затем для каждой альтернативы (каждой i-й строки таблицы) рассчитаем обобщенную функцию цели: $S_i = Y_{i1} \cdot w_1 + Y_{i2} \cdot w_2 + ... + Y_{in} \cdot w_n$. Лучшей считается альтернатива i, для которой обобщенная функция цели принимает максимальное значение. В примере, согласно методу многопараметрической средневзвешенной оптимизации, следует выбрать альтернативу 2.

2. ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА ДЛЯ ПОСТРОЕНИЯ ОМАС ППР

В качестве инструментальных средств для построения открытых мультиагентных систем поддержки принятия решений рассмотрим конструктор онтологий [6] и исполняющую систему [7], разработанные компанией Magenta Development.

2.1. Конструктор онтологий

2.1.1. Структура конструктора онтологий

Определим онтологию как совокупность пополняемых баз знаний, структурированных в виде семантической сети, описывающей виртуальный мир предметной области. Эта семантическая сеть содержит как декларативные, так и процедурные компоненты, и включает такие понятия как ***объекты***, ***свойства***, ***процессы***, ***отношения*** и ***атрибуты***.

Конструктор онтологий (КО) – это программа визуального проектирования семантических сетей, в которой пользователь может в удобной форме описывать предметную область, т.е. создавать и редактировать онтологии, определяя свои концепты и устанавливая связи между ними, а также формируя сценарии действий. Напомним, что концепт онтологии – это объект онтологии, который соответствует некоторой сущности (объекту, атрибуту, процессу) проектируемого мира. Все концепты онтологии принадлежат одному из определенных в КО типов.

В конструкторе онтологий принята следующая организация знаний: все знания расположены в библиотеках онтологий, представляющих собой совокупность онтологий.

Онтология, в свою очередь, состоит из концептов и связей между ними, причем однотипные концепты помещаются в категории. Графически такая схема хранения знаний представлена на рис. 11.

Рис. 11. Схема хранения знаний в конструкторе онтологий

В конструкторе онтологий можно оперировать двумя типами онтологий.

1. Онтология предметной области, т.е. физического мира (*Descriptive ontology*). В дескриптивной онтологии описывается структура и взаимосвязь всех объектов мира, а также законы, действующие в этом мире, и сценарии, по которым живут объекты данного мира. Базовыми категориями концептов дескриптивной онтологии являются:

- Объекты (*Objects*),
- Свойства (*Properties*),
- Атрибуты (*Attributes*),
- Скрипты (*Scripts*),
- Отношения (*Relations*).

2. Онтология мира ресурсов и заказов, т.е. виртуального мира (*Virtual world ontology*). Данный класс онтологий описывает мир в терминах заказов, ресурсов и отношений матчинга между ними. Базовыми категориями концептов онтологии ресурсов и заказов являются:

- Концепты виртуальных агентов заказов (*Demand Agents*),
- Концепты виртуальных агентов ресурсов (*Resource Agents*),
- Виртуальные отношения (*Virtual Relations*).
- К виртуальным отношениям относятся:
- Отношение матчинга (*Matching relation*),
- Отношение создания агентов (*Creation relation*),
- Отношение создания субагентов (*Subagent creation relation*),
- Реверсивное отношение создания субагентов (*Reversible subagent creation relation*).

Все представленные концепты связаны друг с другом и представляют собой универсальный набор базовых концептов – ***метаонтологию***, которую можно применять для широкого круга приложений.

2.1.2. Назначение конструктора онтологий

Конструктор онтологий предназначен для решения следующих задач:

- Создание и редактирование дескрипторов (описателей) объектов, отношений и скриптов предметной области, объединяемых в семантическую сеть.
- Создание и редактирование дескрипторов (описателей) объектов, отношений и скриптов виртуального мира, обеспечивающего поддержку процессов матчинга агентов и принятия решений.
- Построение потоковой модели вычисления значений атрибутов.
- Отображение семантической сети предметной области и виртуального мира в графическом виде.
- Отображение взаимозависимости между понятиями.
- Сохранение построенных семантических сетей для последующей работы с ними исполняющей системы.

Конструктор онтологий обеспечивает следующие возможности:

- **Проектирование семантической сети**. Очень важной особенностью проектирования является то, что внесение изменений в онтологию возможно также в ходе работы системы без ее перезапуска, что обеспечивает чрезвычайную гибкость системы. Все изменения может осуществлять эксперт в данной предметной области, поскольку с внедрением конструктора онтологий для написания нового сценария достаточно изменить бизнес-правила, открытые для пользователя в онтологии.
- **Связь агентов с онтологией**. Выполнение буферных функций между онтологией и агентом, т.е. обеспечение работы с онтологией и экземплярами объектов в сцене. При этом исполняющая система выполняет передачу сообщений, синхронизацию и т.д., не работая напрямую с онтологией, а получая требуемую информацию от специальных сервисных объектов, таких как матчеры и экземпляры объектов.
- **Процесс динамического установления связей**. Предоставление механизма задания и проверки условий матчинга между заказом и ресурсом, который позволяет формировать список подходящих ресурсов для определенного заказа, т.е. реализует настраиваемый механизм принятия решений для установления динамических связей (соглашений) между агентами.
- **Выбор агентом вариантов из множества альтернатив**. Предоставление разработчику средства принятия решений, ориентированного на агентов, – машины принятия решений. Машина принятия решений по условиям принятия решений сортирует предложенные заказу альтернативы ресурсов, указывая на более предпочтительные варианты.
- **Вычисление значений атрибутов в потоковой схеме**. Предоставление агенту механизма сбора необходимых значений атрибутов, не только принадлежащих данному агенту, но и определяемых скриптами либо другими агентами.

2.1.3. Интерфейс конструктора онтологий

2.1.3.1. Общая структура экрана конструктора онтологий

Вызов конструктора онтологий производится при запуске программы *OntCons.exe*, находящейся в папке *OntConsUniIntf*. Как только будет открыта вновь созданная или уже существующая онтология, появится рабочий экран конструктора онтологий *Magenta Ontology Constructor*, аналогичный тому, что показан на рис. 12.

Экран конструктора онтологий содержит следующие компоненты:

- Верхняя строка, кроме сокращенного названия инструментальной системы – *«MagentA Ontology Constructor»*, содержит имя открытого проекта.
- В строке заголовков меню содержатся команды, которые запускают функции конструктора и поддерживают работу его отдельных компонент.
- Кнопки-пиктограммы панели инструментов обеспечивают вызов важнейших и чаще всего исполняемых команд конструктора.

- В левой части экрана располагается менеджер конструктора онтологий, позволяющий отображать архитектуру онтологий, входящих в библиотеку, в виде иерархической структуры дерева категорий и концептов, а также осуществлять удобную навигацию по компонентам этого дерева. Если узел навигатора развернут, то слева от пиктограммы и названия соответствующего узла указан знак «-», в противном случае – знак «+». Например, на рис. 12 в менеджере конструктора онтологий показана библиотека под названием *DeRisk*, содержащая дескриптивную онтологию *DeRisk descriptive ontology* (соответствующий ей узел навигатора развернут) и онтологию виртуального мира *DeRisk virtual world* (соответствующий ей узел навигатора свернут). В дескриптивной онтологии развернуты категории объектов (*Objects*) и скриптов (*Scripts*).

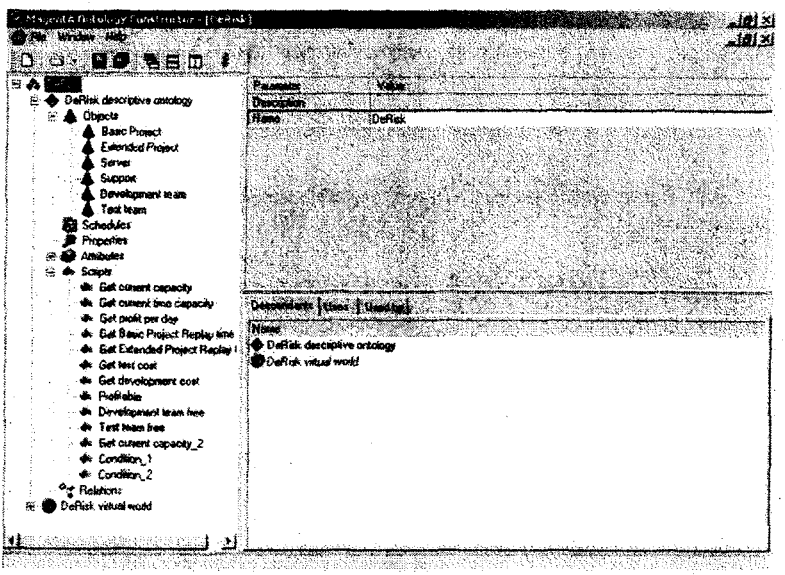


Рис. 12. Общая структура рабочего экрана конструктора онтологий

- В правой верхней части экрана располагается редактор свойств концептов, с помощью которого пользователь может просматривать свойства концептов и задавать определенные значения этим свойствам.
- В правой нижней части экрана для каждого концепта, выбранного в дереве классов, в закладке *Descendants* отображаются потомки этого концепта, находящиеся на следующем уровне, в закладке *Uses* – все концепты онтологии, которые выбранный концепт использует в своей работе, в закладке *Used by* – все концепты, которые используют сам данный концепт. Нахо-

дясь в закладке *Uses* или *Used by*, можно установить режим отображения связей данного концепта с другими концептами в онтологии (рис. 13):

– *Show link by category* – показать связи концепта с концептами других категорий,

– *Show inherited link* – показать связи концепта в иерархии наследования.

Каждый элемент библиотеки онтологий имеет контекстное меню, которое можно вызвать с помощью нажатия правой кнопки мыши на пиктограмме соответствующего элемента.

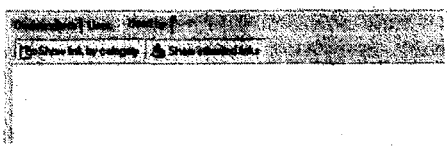


Рис. 13. Режимы отображения связей концептов в онтологии

2.1.3.2. Основные меню интерфейса конструктора онтологий

Вид меню ***Ontology as network*** представлен на рис. 14. К командам этого меню относятся:

Рис. 14. Меню *Ontology as network*

- *Restore* – уменьшить размер окна конструктора онтологий.
- *Minimize* – свернуть окно конструктора онтологий.
- *Close* – закрыть окно конструктора онтологий.
- *Next* – перейти к следующей онтологии из списка ранее редактировавшихся.

Вид меню *File* представлен на рис. 15. Это меню содержит следующие команды:

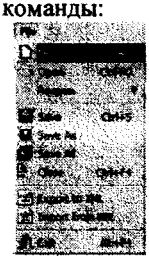


Рис. 15. Меню *File*

- *New* – создать новую онтологию.
- *Open* – открыть ранее созданную онтологию, которая хранится в файле с расширением **.ocl*.
- *ReOpen* – открыть онтологию из списка ранее редактировавшихся онтологий.
- *Save* – сохранить редактируемую (активную) онтологию.
- *Save As* – сохранить онтологию в файле под новым именем.
- *Save All* – сохранить все онтологии.
- *Close* – закрыть файл онтологии.

- *Export to XML* – преобразовать из формата представления онтологии в формат XML.
- *Import from XML* – преобразовать из формата XML в формат представления онтологии.
- *Exit* – завершить работу с конструктором онтологий. Если файл онтологии редактировался и не был сохранен, то появится предложение сохранить измененный файл.

Вид меню ***Tools*** представлен на рис. 16. Это меню содержит следующие команды:

Рис. 16. Меню *Tools*

- *Ontology as network* – показать онтологию в виде семантической сети (см. 2.1.3.5).
- *Options* – установить опции конструктора онтологий. Окно установки опций КО показано на рис. 17.

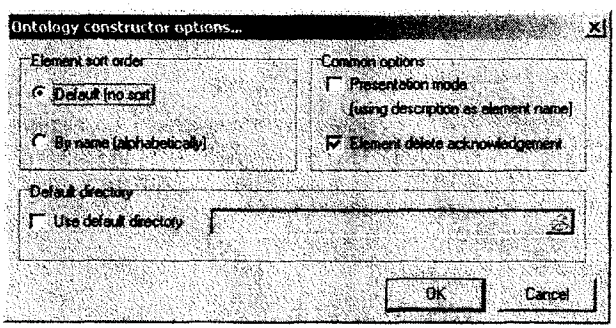


Рис. 17. Опции конструктора онтологий

Возможна установка следующих опций:

- *Element sort order* – режим сортировки элементов в дереве концептов:
  - *Default* – без сортировки (в порядке создания);
  - *By name* – по алфавиту.
- *Default directory* – директория по умолчанию. Если директория задана, то при открытии файла будет показа эта директория, в противном случае – последняя использовавшаяся директория.
- *Presentation mode* – отображение вместо имен концептов их описаний.
- *Element delete acknowledgement* – запрос подтверждения удаления концепта.

Вид меню ***Window*** представлен на рис. 18. Это меню содержит следующие команды:

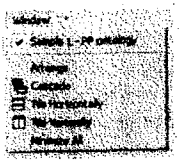


Рис. 18 – Меню *Window*

- *Arrange* – расположение окон на экране:

– *Cascade* – разместить окна с отображаемыми файлами онтологий каскадом, с перекрытием окон;

– *Tile Horizontally* – разместить окна с отображаемыми файлами онтологий друг за другом горизонтально без перекрытий;

– *Tile Vertically* – разместить окна с отображаемыми файлами онтологий друг за другом вертикально без перекрытий.

- *Minimize All* – свернуть все окна.

В верхней части меню *Window* располагается список файлов, открытых конструктором онтологий, где отмечен активный файл (активное окно) и пользователь может выбрать в качестве активного любой другой файл из этого списка, расположив его при перекрытии сверху всех других редактируемых файлов.

Меню ***Help*** содержит всего одну команду:

- *About* – отобразить краткую информацию о конструкторе онтологий (номер версии, дата копирования, авторские права).

2.1.3.3. Панель инструментов конструктора онтологий

Все кнопки-пиктограммы панели инструментов, обеспечивающие вызов важнейших и чаще всего исполняемых команд системы, уже были показаны при обсуждении меню. Они отображаются рядом с соответствующими командами и видны на рисунках, представляющих различные меню системы. Поэтому здесь ограничимся лишь их перечислением и краткой справкой:

 – создать новую онтологию,

 – открыть ранее созданную онтологию,

 – сохранить активный файл онтологии,

 – сохранить все файлы онтологий,

 – разместить окна с отображаемыми файлами онтологий каскадом,

 – разместить окна с отображаемыми файлами онтологий горизонтально,

 – разместить окна с отображаемыми файлами онтологий вертикально.

 – получить краткую справку о конструкторе онтологий.

2.1.3.4. Редактор свойств конструктора онтологий

Редактор свойств концептов предназначен для просмотра и редактирования значений атрибутов библиотеки, онтологий, категорий и концептов онтологий. Например, для изменения названия библиотеки, онтологии или категории, а также имени любого концепта необходимо в окне редактора свойств выбрать в списке параметров поле *Name* и отредактировать его. При редактировании зна-

чения любого атрибута изменения вступают в силу либо после нажатия клавиши *Enter*, либо после того, как поле перестало быть активным. У каждого концепта (как и у онтологии и категории) есть параметр *Description*, который представляет собой строку. В данном поле (необязательно) можно ввести словесное описание концепта.

2.1.3.5. Просмотр онтологии как семантической сети

В конструкторе онтологий имеется возможность просмотра онтологии в виде семантической сети (ориентированного графа, узлами которого являются концепты онтологии, а ребрами – связи между ними – рис. 19). Для этого необходимо выбрать в контекстном меню любого элемента библиотеки онтологий пункт *Ontology as network* (для онтологии мира заказов ресурсов имеется дополнительно еще один режим просмотра, доступный из пункта меню *View Virtual World net*).

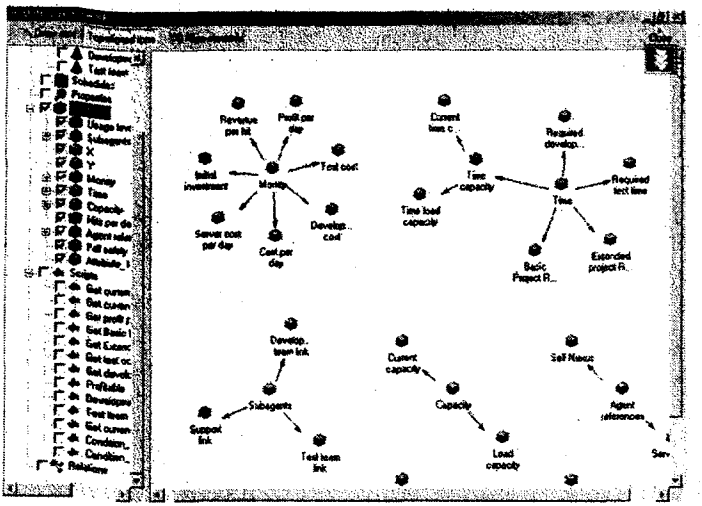


Рис. 19. Просмотр онтологии в виде семантической сети

При просмотре онтологии в виде сети пользователь имеет возможность перемещать концепты (посредством «захвата» мышью), скрывать определенные вершины, выполнять автовыравнивание элементов.

Остальные функции конструктора онтологий рассмотрим в процессе проектирования мультиагентных приложений.

2.2. Исполняющая система

Исполняющая система предназначена для создания сцены виртуального мира в соответствии с онтологией предметной области, спроектированной с использованием конструктора онтологий, а также для моделирования созданной сцены виртуального мира и визуализации протекающих в нем процессов.

2.2.1. Интерфейс исполняющей системы

2.2.1.1. Общая структура экрана исполняющей системы

Вызов исполняющей системы производится при запуске программы UniIntf.exe, находящейся в папке *OntConsUniIntf*. На экране перед пользователем появляется основное окно, аналогичное рис. 20.

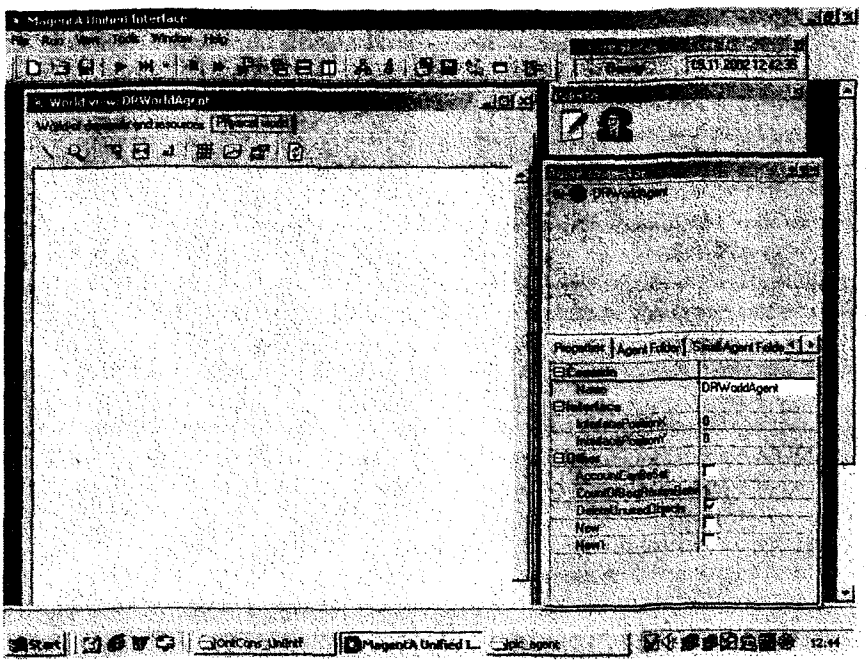


Рис. 20. Основное окно интерфейса исполняющей системы

Экран исполняющей системы содержит следующие компоненты:

- Верхняя строка, кроме сокращенного названия инструментальной системы – *«MagentA Ontology Constructor»*, содержит имя открытого проекта.
- В строке заголовков меню содержатся команды, которые запускают функции исполняющей системы и поддерживают работу ее отдельных компонент.

- Кнопки-пиктограммы панели инструментов обеспечивают вызов важнейших и чаще всего исполняемых команд исполняющей системы.
- В левой части экрана располагается окно *Virtual World*, предназначенное для конструирования сцены.
- В правом верхнем углу экрана располагается окно с отметками о текущем состоянии ядра исполняющей системы (*Core Status*) и модельном (системном) времени (*Core Time*).
- Ниже располагается палитра, содержащая пиктограммы концептов.
- Ниже палитры располагается окно инспектора агентов. Инспектор агентов предназначен для редактирования их свойств.

2.2.1.2. Основные меню интерфейса исполняющей системы

Большинство команд **меню *File*** традиционны (открыть, сохранить, закрыть), но непосредственными объектами здесь являются не файлы, а сцены виртуального мира. Меню ***File*** содержит следующие команды:

- *New scene* (пиктограмма на панели инструментов, горячие клавиши – <Ctrl>+<N>) – создание новой сцены.
- *Open scene* (, F3) – открытие ранее сохраненной сцены (выбор файлов с расширением **.osf*).
- *Save scene* (, F2) – сохранение текущей сцены.
- *Save the scene as…* – сохранение текущей сцены под новым именем.
- *Close scene* (<Ctrl>+<F7>) – закрытие текущей сцены.
- *Batch* – пакетный режим презентаций, который позволяет определять последовательность создания сцен, агентов, запуска сцен и т.д. – зарезервировано.
- *Reopen last scene* – зарезервировано.
- *Exit* (<Alt>+<F4>) – выход из программы.

Меню *Run* содержит команды запуска и останова моделирования созданной или загруженной сцены, а также настройки цикла исполнения:

- *Run* (, F9) – запуск моделирования сцены без остановок, до завершения выполнения всех операций или выполнения команды *Stop*.
- *Stop* (, F7) – остановка моделирования сцены, которая происходит по данной команде после исполнения текущего шага (цикла) моделирования.
- *Next dispatch* (<Ctrl> + <F8>) – зарезервировано.
- *Run To* (<Ctrl> + <F9>) – запуск моделирования сцены до определенного времени.
- *Default Core Time* – установка текущего времени ядра исполняющей системы.
- *Next step* (, F8) – выполнение цикла (шага) моделирования, установленного режимом работы системы (опциями *Step Mode*).
- *Step mode* – режим пошаговой работы системы, где можно активировать одну из следующих опций:

– *Until time validation started* – запустить программу до начала проверки времени в цикле;

– *Until time changing started* – запустить программу до начала смены времени в цикле и остановиться до смены;

– *Until time changing finished* – запустить программу до смены времени в цикле и остановиться сразу после смены;

– *Until next time mark started* – запустить программу до начала следующей временной отметки.

Перечисленные режимы *Step Mode* позволяют более точно позиционировать во времени важные события, которые происходят в мультиагентной системе, например, начало выполнения заказа, завершение переговоров и т.д.

- *Core Options* – опции ядра, которые используются, в основном, при отладке системы:

– *Enable Breakpoints* – разрешить точки прерывания для отладки;

– *Raise Core Exceptions* – разрешить показ ошибок ядра.

Меню *View* обеспечивает возможность работы с группой важнейших компонентов рассматриваемого интерфейса:

- *Show Palette* (F12) – показать палитру концептов.
- *Show Agent Inspector* ( , F11) – показать Инспектор агентов (см. 2.2.2.2).
- *Show Log* ( ,Ctrl+L) – показать системный лог, окно сообщений агентов (см. 2.2.2.3).
- *Confirmations* – установка требований подтверждения при выполнении следующих операций:

– *Destroy Agent* – подтверждение при выполнении операции удаления агента;

– *Close Scene* – подтверждение при выполнении операции закрытия сцены;

– *Exit* – подтверждение при выполнении выхода из исполняющей системы.

- *Options* – системные настройки интерфейса, к которым относятся:

– *Clear Log* – очистить системный лог (окно сообщений агентов).

– *Log smart update* – если этот пункт выбран, то обновление системного лога происходит только после окончания переговоров агентов, в противном случае обновление лога происходит в реальном времени в процессе моделирования.

– *Enable Log* – если этот пункт выбран, то сообщения добавляются в лог в процессе моделирования, в противном случае – нет.

Меню *Tools* включает следующие команды:

- *Manage Extensions* ( , <Shift> + <Ctrl> + <E>) – управление расширениями.

- *Show Exceptions Log* – показать лог сообщений, формируемых расширениями ядра.
- *Agent Views* – настройка окон просмотра базовых свойств агентов, состоящая из подпунктов:

– *Restore on load* – установка данной опции обеспечивает восстановление окна просмотра базовых свойств при загрузке сцены.

– *Close all* – закрывает все окна просмотра базовых свойств.

- *Usage load statistic...* (<Ctrl> + <S>) – статистика изменения значения атрибута Usage level в сцене.
- *Ontology Palette* – показать окно палитры пиктограмм агентов в сцене.
- *Make report* (<Alt> + <M>) – сформировать отчет по сцене в формате **.html*, который включает значения атрибутов агентов и расписания.
- *Scene information* (<Ctrl> + <I>) – полная информация о сцене, включая все расписания, матрицу матчинга и пиктограммы агентов виртуального мира.
- *Save ontology scene* – сохранить онтологическую сцену.
- *Load ontology scene* – загрузить онтологическую сцену на выполнение.
- *Clear results* – очистить результаты предыдущего сеанса моделирования, при этом разрушаются все связи между агентами, ранее установленные в процессе матчинга.

Меню Window – это традиционное меню, в котором содержатся команды по размещению открытых окон с компонентами системы на экране:

- *Arrange* – выстроить иконки окон.
- *Minimize All* – минимизировать все окна.
- *Cascade* () – расположить окна каскадом.
- *Title Horizontally* () – расположить окна по горизонтали.
- *Title Vertically* () – расположить окна по вертикали.

Меню Help содержит следующие команды:

- *Help* – вызов справки, который пока зарезервирован для конкретных приложений.
- *About* () – показать окно с краткой информацией о данной системе и ее разработчиках.

2.2.1.3. Панель инструментов интерфейса исполняющей системы

Панель инструментов из верхней части окна интерфейса представляет собой набор пиктограмм–кнопок, обеспечивающих вызов важнейших команд системы (рис. 21).

Рис. 21. Стандартная панель инструментов унифицированного интерфейса

К стандартным пиктограммам унифицированного интерфейса пользователя мультиагентной исполняющей системы относятся:

 – создание новой сцены (*Create a new scene*).

 – открытие ранее созданной и сохраненной сцены (*Open a scene*).

 – сохранение текущей сцены (*Save the scene*).

 – выполнение очередного шага, а точнее цикла моделирования, признаки завершения которого устанавливается посредством изменения режима работы системы (*Make next step*).

 – установка текущего режима смены времени системы (признака завершения цикла моделирования) (*Step Mode*).

 – остановка выполнения работы системы. Остановка происходит после исполнения текущего шага (*Stop execution*).

 – выполнение моделирования системы без остановок, до завершения выполнения всех операций (*Non–stop run*).

 – запустить сцену на выполнение до определенного значения времени (*Run To*).

 – расположить все открытые окна каскадом (*Cascade windows*).

 – расположить все открытые окна горизонтально (*Tile horizontally*).

 – расположить все открытые окна вертикально (*Tile vertically*).

 – управление расширениями (*Manage extension*).

 – показать диалоговое окно, с информацией о данной системе (*About this program*).

 – загрузить онтологическую сцену (*Load ontology scene*).

 – показать мир запросов/ресурсов (*Show demand resource world*).

 – показать палитру концептов (*View ontology palette*).

 – показать инспектор агентов (*Show Agent Inspector*).

 – показать центральный лог (*Show Log*).

2.2.2. Интерфейс физического и виртуального мира

2.2.2.1. Окна физического и виртуального мира

Для конструирования сцены предназначено окно *Virtual World*, в котором имеются две закладки:

- ***Physical world*** – открывает окно физического мира, в котором собственно и происходит создание агентов и конструирование сцены.
- ***World of demands and resources*** – открывает окно виртуального мира, в котором можно наблюдать процесс матчинга между агентами.
- **Стандартная панель инструментов окна физического мира** представляет собой набор пиктограмм-кнопок, обеспечивающих вызов важнейших команд (рис. 22).

Рис. 22. Стандартная панель инструментов окна физического мира

К стандартным пиктограммам панели инструментов окна физического мира относятся:

 – режим создания соединительных линий между агентами (*Lines creation mode*).

 – ручное масштабирование карты физического мира (*Zooming mode).*

 – режимы автоматического масштабирования карты (*Fit to map, Fit to view).*

 – 100-процентный масштаб карты физического мира (*100% zoom*).

 – отобразить на карте физического мира сетку (*View grid lines*).

 – загрузить фон (*Load background picture*).

 – установить свойства карты физического мира (*Set map properties*).

 – обновить карту физического мира (*Refresh* (F5)).

По умолчанию, при вызове исполняющей системы открыто окно физического мира.

Стандартная панель инструментов окна ***виртуального мира*** представляет собой набор пиктограмм–кнопок, обеспечивающих возможность редактирования агентов виртуального мира.

Чтобы создать агента, соответствующего концепту ранее созданной онтологии, необходимо выбрать в палитре агентов пиктограмму этого концепта и «перетащить» ее на поле физического мира. У выбранного агента можно редактировать значения онтологических атрибутов в инспекторе агентов (рис. 23).

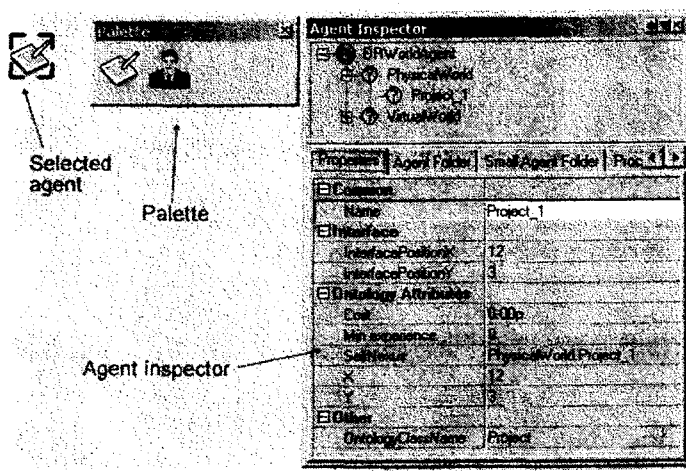


Рис. 23. Создание агента и инспектор агента

2.2.2.2. Инспектор агентов

Инспектор агентов, вызов которого осуществляется командой *View → Show Agent Inspector* (, <F11>), является одним из базовых расширений унифицированного интерфейса и служит для обеспечения доступа к агентам создаваемой сцены и их параметрам. Именно с помощью инспектора агентов можно просматривать иерархическое дерево, связывающее агентов, изменять их варьируемые характеристики (свойства, атрибуты) и анализировать параметры, представляющие результаты моделирования.

Иерархическое дерево отношений между объектами сцены

Окно **Инспектор агентов** состоит из двух основных частей (рис. 24). Верхняя часть окна (обведена) – представляет собой иерархическое дерево, отображающее основные отношения между агентами проектируемой сцены. При этом возможны два вида представления окна Инспектор агентов: полное иерархическое дерево взаимосвязи агентов и отдельный узел дерева взаимосвязи агентов. Для переключения между этими двумя видами представления окна используется кнопка , расположенная в его правом верхнем углу.

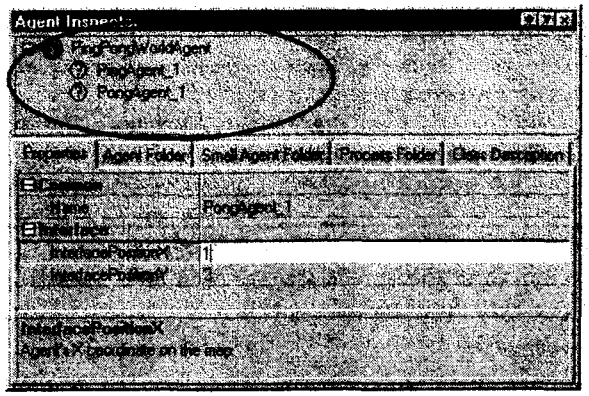


Рис. 24. Инспектор агентов с полным иерархическим деревом

После выбора необходимого агента в дереве и щелчка правой кнопкой мыши, на экране появляется всплывающее меню, представленное на рис. 25.

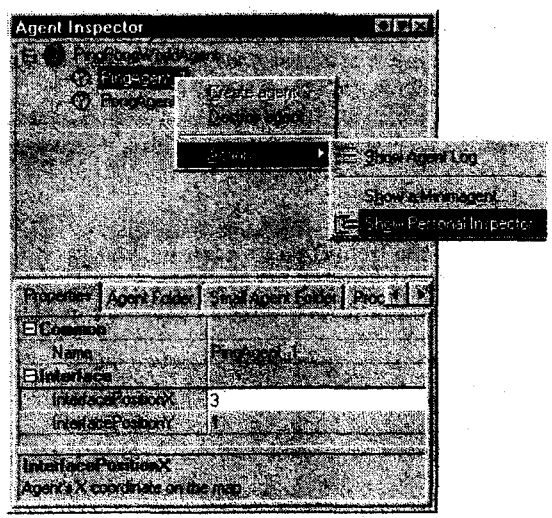


Рис. 25. Всплывающее меню в инспекторе агентов

С помощью этого меню можно осуществлять следующие действия над агентами:

- *Create agent* – создание нового агента, для которого текущий выбранный

агент будет владельцем. (В простых примерах данная команда неактивна).

- *Destroy agent* – удаление агента, вместе с которым будут удалены все процессы этого агента, а также все агенты (в том числе и малые), для которых этот агент является владельцем.
- *Show Agent Log* – показать персональный лог для выбранного агента.
- *Show a Minimagent* – зарезервирована.
- *Show Personal Inspector* – создать персонального инспектора агента. Основное отличие от обычного инспектора агентов состоит в том, что отсутствует окно с иерархическим деревом.

Вторая часть окна инспектора агентов состоит из закладок, отражающих характеристики выбранного агента.

Закладка свойств объекта

Закладка ***Properties*** – содержит список свойств выделенного объекта.

Свойства могут быть подразделены на некоторые подгруппы, такие как ⊞Common или ⊟Interface, имена которых отображаются в окне жирным шрифтом. Для того чтобы показать/скрыть свойства конкретной подгруппы, необходимо щелкнуть левой кнопкой мыши на белый квадратик слева от названия подгруппы.

При выборе того или иного свойства в самой нижней строке окна Инспектора агентов отображается краткая информация об этом свойстве.

Для изменения значения свойства следует выбрать его и ввести в редактируемое поле данного свойства его новое значение.

Кроме важнейшей закладки *Properties*, окно инспектора агентов имеет ряд закладок для отображения списков дочерних агентов и активных процессов выбранного агента.

Закладка Agent Folder

Закладка ***Agent Folder*** отображает список подчиненных объектов-агентов, для которых объект, выделенный в дереве Инспектора, является владельцем (рис. 26). Этот список соответствует списку в подуровне иерархического дерева.

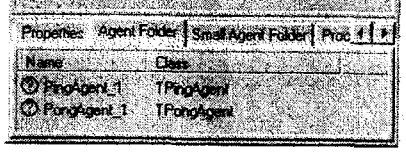


Рис. 26. Закладка *Agent Folder* инспектора агентов

2.2.2.3. Системный лог

Системный лог (окно сообщений) является одним из базовых расширений унифицированного интерфейса и обеспечивает визуализацию переговоров между агентами, которые они ведут в процессе решения поставленной задачи. Это не только инструмент отладки, который предоставляется проектировщику системы, но и важнейший компонент, анализируя который, пользователь может выполнять трассировку алгоритмов взаимодействия агентов.

Системный лог представлен двумя разновидностями: *Central Log* и персональный лог для одного агента. В *Central Log* отображаются переговоры всех агентов системы (рис. 27), а в персональном логе – только переговоры, связанные с конкретным выбранным агентом. Персональный лог начинает заполняться только после его создания, выполненного пользователем. Это основное и единственное отличие логов, поэтому описание *Cenral Log* применимо и к персональному логу. Для вызова персонального лога используется команда *Show Agent Log* из контекстного меню в Инспекторе агентов или функция *Create a new log* во всплывающем меню *Central Log*.

Итак, окно *Central Log*, аналогичное окну, представленному на рис. 27, появляется на экране после выполнения команды *View* → *Show Log* (, *<Ctrl>* + *<L>*).

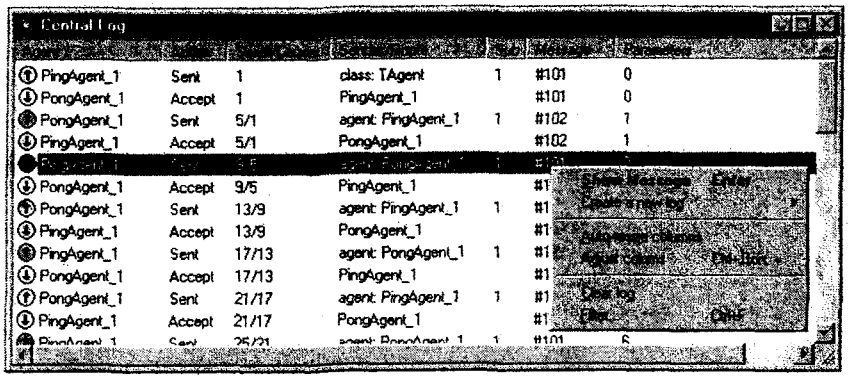


Рис. 27. Окно расширения *Central Log*

Поля окна Central Log

- Agent – полное имя агента, выполнившего данное действие. Слева от имени агента располагается вспомогательная пиктограмма, обозначающая следующие действия:

  – отправка сообщения агентом.

  – получение сообщения агентом.

  – отказ агента получать сообщение.

  – системный комментарий.

- Action – действие, выполненное агентом, может иметь один из следующих типов:
  - *Sent* – отправка сообщения агентом.
  - *Accept* – получение сообщения агентом.
  - *Reject* – отказ агента получать сообщение.
- *Serial* – порядковый номер сообщения.
- *Causer* – порядковый номер сообщения, вызвавшего появление данного сообщения.
- *Sender/Route* – в случае отправки сообщения — агент или тип агентов, которому было послано сообщение; в случае получения сообщения – агент, отославший данное сообщение.
- *Sub* – порядковый номер процесса у агента, который выполнил отправку сообщения.
- *Message* – тип отправляемого/получаемого сообщения. Каждое сообщение имеет уникальный номер, указанный в скобках.
- *Parameters* – параметры, передаваемые сообщением. Для более удобного просмотра параметров следует вызвать окно просмотра параметров сообщения, выделив интересующую строку сообщений с помощью двойного щелчка мыши или нажатия на клавишу <Enter>.

Описание всплывающего меню

Всплывающее контекстное меню, связанное с окном *Central Log*, показано на рис. 27 вместе с окном лога. Это меню содержит следующие пункты:

- *Show Message* – вызвать диалоговое окно просмотра параметров сообщения.
- *Create a new log ⇒ by Agent* – создание персонального лога для агента.
- *Auto–resize columns* – если этот пункт выбран, то ширина колонок подстраивается под их содержимое.
- *Adjust columns* – выровнять колонки по размеру самого большого значения каждой колонки (<Ctrl> + <Num>).
- *Clear log* – очистить лог.
- *Filter* – вызов окна настройки фильтра сообщений (<Ctrl> + <F>), показанного на рис. 28.

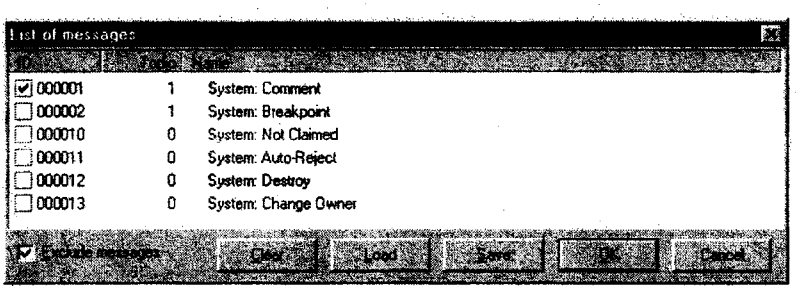


Рис. 28. Окно настройки фильтра сообщений

Фильтр сообщений следует использовать для вывода только определенных типов сообщений. Для того чтобы сообщения некоторых типов не отображались в логе, следует пометить их галочкой, если опция *Exclude messages* включена. Если опция *Exclude messages* выключена, отображаться в логе будут только помеченные сообщения. Для очистки, загрузки, сохранения фильтра используются соответствующие функции *Clear*, *Load*, *Save*. Каждый фильтр сохраняется в отдельном файле с расширением **.llf* и для работы с ним используется стандартный диалог *Windows*.

Краткое описание системных сообщений

Системные сообщения – это сообщения, которые посылает исполняющая система мультиагентного приложения. Они представлены следующими типами:

- *System: Comment* – системный комментарий.
- *System: BreakPoint* – системное сообщение, симулирующее нажатие кнопки Stop.
- *System: NotClaimed* – сообщение системы агенту о том, что посланное им сообщение не нашло ни одного адресата.
- *System: Auto-Reject* – сообщение, которое отсылает агент, получивший какое-либо сообщение, но не ответивший на него и не отключивший опцию Auto-Reject.
- *System: Destroy* – системное сообщение, при получении которого агент самоуничтожается.
- *System Change Owner* – сообщение о смене владельца агента.

2.3. Контрольные вопросы

1. Приведите примеры предметных областей, где требуется динамическое распределение ресурсов.
2. В чем состоит проблема динамического распределения ресурсов?
3. В чем заключаются особенности функционирования сетевых организаций?
4. Перечислите основные черты холистического подхода к управлению современными предприятиями.
5. Какие программные системы по управлению ресурсами предприятий вам известны? Каковы их недостатки в условиях управления открытыми системами?
6. Как вы можете определить понятие мультиагентной системы? Приведите пример какой-либо предметной области, которую можно адекватно описать с использованием мультиагентного подхода.
7. Назовите этапы развития мультиагентных технологий оперативной обработки информации для поддержки процессов принятия решений.
8. Что такое «агент»? Каковы отличительные свойства интеллектуальных агентов?
9. Каковы основные отличия мультиагентных систем от жестко организованных систем?
10. Дайте определение сети потребностей и возможостей (ПВ-сети). Опишите модель ПВ-сети.

11. Что такое матчинг потребностей и возможностей? Как он осуществляется?

12. Дайте понятие самоорганизации в ПВ-сети.

13. Какие базовые сущности используются для реализации ПВ-сети?

14. Как вы можете определить понятие порождающей ПВ-сети? Что такое реализация ПВ-сети? Как она выполняется?

15. Какие методы взаимодействия агентов в ПВ-сети вам известны? Сформулируйте отличительные особенности каждого из них.

16. Приведите пример ПВ-сети в области логистики.

17. Опишите основные компоненты архитектуры ОМАС ППР.

18. Что такое «онтология»? Из каких частей она состоит? Дайте определения следующих понятий: дескриптивная онтология, онтология мира заказов и ресурсов.

19. Дайте определения следующих понятий: концепт, категория концептов.

20. Дайте определение метаонтологии ОМАС ППР. Каковы функции метаонтологии?

21. Каким образом семантические сети используются для представления метаонтологии?

22. Какие специализированные компоненты используются для работы в ПВ-сетях?

23. Поясните алгоритм работы матчера.

24. Поясните алгоритм проверки условий матчинга.

25. В чем заключаются особенности понятий физического и виртуального миров?

26. Какие инструментальные средства используются для построения ОМАС ППР?

27. Каковы основные функции конструктора онтологий?

28. Опишите структуру библиотеки онтологий. Какие основные категории концептов в ней присутствуют?

29. Для чего предназначено дерево категорий и концептов менеджера конструктора онтологий?

30. Как создать концепт дескриптивной онтологии и задать его свойства (на примере концептов «объект» и «атрибут»)? Как связать концепты между собой?

31. Как создать концепт онтологии мира заказов/ресурсов и задать его свойства (на примере концептов «заказ» и «ресурс»)?

32. Каковы функции физического и виртуального миров в исполняющей системе?

33. Как сконструировать онтологическую сцену с помощью инструментов, предоставляемых исполняющей системой?

34. Как создать агента концепта мира заказов/ресурсов, присутствующего в онтологической сцене?

35. Что такое Инспектор агентов, каковы его функции?

36. Для чего предназначен системный лог и логи агентов? Какую информацию можно получить с их помощью?

3. ЛАБОРАТОРНЫЙ ПРАКТИКУМ

Основная цель лабораторного практикума:

- Изучение мультиагентного подхода к решению задач, связанных с динамическим распределением ресурсов в условиях неопределенности спроса и предложения.
- Освоение принципов построения баз знаний (онтологий) открытых мультиагентных систем для различных предметных областей.
- Приобретение навыков построения мультиагентных систем «распределенного интеллекта», агенты которых способны к переговорам и принятию согласованных решений.

Основные задачи лабораторного практикума:

- Изучение базовых понятий, определяющих особенности мультиагентного подхода к проектированию программного обеспечения.
- Изучение принципов построения мультиагентных систем, предназначенных для изучения динамики функционирования сложноорганизованных систем.
- Освоение онтологического подхода к описанию знаний о структуре и алгоритмах функционирования сложноорганизованных систем.
- Практическое освоение инструментальных средств разработки мультиагентных приложений.
- Освоение приемов конструирования онтологий в различных предметных областях с помощью конструктора онтологий.
- Практическое освоение технологии моделирования процессов взаимодействия между объектами в среде открытой мультиагентной исполняющей системы:
  - работа с онтологией, включая обновление онтологии «на лету», в ходе моделирования;
  - выполнение потоковых вычислений, т.е. расчетов необходимых параметров «на лету», непосредственно в процессе моделирования,
  - выполнение матчинга (предварительного выбора возможных вариантов решений);
  - принятие решений на основе возможных вариантов;
  - пересмотр решений в ходе работы.
- Освоение более сложных технологий принятия решений (простые ресурсы и заказы с односторонней активностью – двусторонняя активность – разделяемые ресурсы и заказы).
- Анализ различных ситуаций, возникающих при решении задач динамического распределения ресурсов.

Лабораторный практикум предусматривает освоение инструментальных средств разработки мультиагентных приложений компании Magenta Development [6,7].

Лабораторный практикум включает серию работ, посвященных изучению различных аспектов применения онтологий и мультиагентного подхода для поддержки процессов принятия решений в различных областях производственной сферы.

4. ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ ОНТОЛОГИИ В РАЗЛИЧНЫХ ОБЛАСТЯХ ПРОИЗВОДСТВЕННОЙ СФЕРЫ

В главе рассмотрены примеры использования онтологического подхода и мультиагентных технологий в различных производственных сферах. Приведены примеры выполнения студентами СГАУ индивидуальных заданий в рамках изучения дисциплины «Онтология производственной сферы».

В задачи данного курса, помимо получения теоретических знаний по искусственному интеллекту и реферативному знакомству с онтологией, как разделом философии, входят также:

– знакомство студентов с онтологическими системами, которые могут быть использованы при описании ряда производственных сфер;

– получение практических навыков работы с такими системами при решении задач, типичных для аэрокосмического профиля (выбор самолета, двигателя и воздушная логистика) [8-12].

Следует подчеркнуть, что умение структурировать изучаемую предметную область и составлять онтологические описания производственной сферы предприятия особенно важны для выпускника университета. Фактически студент должен быть готов к работе инженером по знаниям или инженером-онтологом. Для этого студент должен научиться главному – анализировать информационное пространство производственной сферы для дальнейшего синтеза эффективных решений.

Студенту предлагается самостоятельно выбрать ту или иную производственную сферу, в которой ему предстоит разобраться. Предметная область не имеет принципиального значения, отсюда и широкий диапазон тем, предлагаемых студентам. Помимо тем, приведенных ниже, приветствуются любые инициативы и предложения.

Индивидуальное задание выполняется каждым студентом самостоятельно, в исключительных случаях – бригадой до 2-3-х студентов. В последнем случае в отчете указывается конкретный вклад каждого студента в общий результат: какой раздел он выполнил самостоятельно.

Название индивидуальной зачетной работы в первой части у всех одинаковое: «ОНТОЛОГИЯ ПРОИЗВОДСТВЕННОЙ СФЕРЫ: ...». Отличие заключается лишь в предметной части – выбранной производственной сфере, что и определит особенность каждого проекта.

Примерные темы могут быть выбраны из таких областей как производственная, складская, транспортная логистика, планирование работы персонала и пр. Изучаемой или исследуемой предметной областью могут быть различные предприятия, учреждения, такие как:

- университет,
- деканат,
- кафедра,
- группа,

- ...
- издательство,
- рекламное агентство,
- туристическое агентство,
- риэлторское агентство,
- кадровое агентство,
- банк...
- производство автомобилей,
- производство компьютеров,
- производство подшипников,
- производство телефонов
- производство ...
- перевозки ж/д грузовые,
- перевозки ж/д пассажирские,
- перевозки авто пассажирские,
- перевозки авто грузовые,
- перевозки речные грузовые,
- перевозки речные пассажирские,
- перевозки ...

После выбора конкретного предприятия студенту необходимо:

– в максимально упрощенном виде представить схему или фрагмент схемы работы конкретного реального предприятия (цеха, участка, станка);

– выделить существующие объекты, определить имеющиеся ресурсы и имеющиеся заказы;

– определить атрибуты этих объектов, в том числе простейшие формулы для расчета их характеристик;

– определить существующие связи между объектами на предприятии;

– определить критерии и условия функционирования предприятия;

– с помощью конструктора онтологии создать онтологию предприятия;

– сформировав варианты реальных исходных данных, составить сцену и запустить матчинг;

– проанализировать полученные результаты;

– составить подробный отчет;

...

– получить удовольствие от проделанной работы,

– получить высокую оценку преподавателя и ЗАЧЕТ, а в перспективе – интересную работу!

Таков несложный, на первый взгляд, алгоритм выполнения индивидуального задания.

4.1. Использование онтологии в банковской сфере: «Ипотечное кредитование»

4.1.1. Постановка задачи

Приведенный пример был составлен и решен в докризисный период – весной 2008 года и на тот период был весьма актуален. Задача была сформулирована следующим образом: физическому лицу, желающему улучшить свои жилишные условия, но не имеющему достаточных средств, необходимо получить заемные средства в одной из кредитных организаций Самары.

На основании заданных возможностей физического лица по зарплате, сроку кредита, сумме кредита, первоначально имеющейся сумме денег, необходимо определить наиболее подходящий банк с минимальной суммой выплат по кредиту в месяц.

4.1.2. Решение задачи

Вначале определяются объекты или сущности, участвующие в процессе взаимодействия. В данном примере это два объекта: *Клиент*, выступающий как проект, и *Банк* – кредитное учреждение, выступающее как источник необходимых ресурсов.

После определения объектов необходимо определить их атрибуты.

Атрибуты Клиента:

– сумма кредита, необходимая Клиенту;

– срок, на который Клиент желает взять кредит;

– сумма средств, которая имеется у Клиента и которую он может внести в качестве первоначального взноса;

– общий доход Клиента в месяц.

Атрибуты Банка:

– процент или доля первоначального взноса, который необходимо внести клиенту,

– годовая процентная ставка по кредиту.

Для создания онтологии получения ипотечного кредита в Самарских банках определены следующие атрибуты концептов онтологии Клиент и Банк:

– Sum_ipoteki – кредит, необходимый Клиенту, руб.;

– Time – срок кредита, лет;

– Dohod – общий доход Клиента в месяц, руб.;

– Percent_perv_vznosa – процент первоначального взноса, который необходимо внести Клиенту, в долях;

– Percent – годовая процентная ставка по Кредиту (индивидуально для каждого банка), в долях;

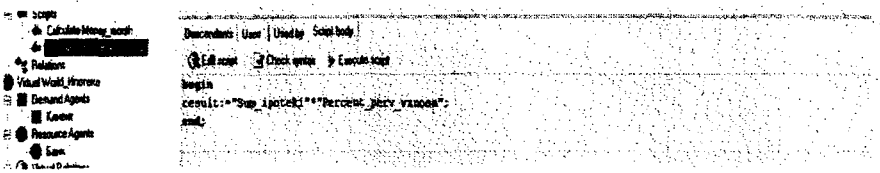
– Pervonach_sum_client – первоначальный взнос Клиента, руб.;

– Pervonach_vznos – минимальная начальная сумма взноса по кредиту (индивидуально для каждого банка), руб.

Значение атрибута Pervonach_vznos определяется по формуле:

$$\text{Pervonach\_vznos} = \text{Sum\_ipoteki} \cdot \text{Persent\_perv\_vznosa}.$$

Расчет по этой формуле осуществляется с помощью скрипта *Calculate Pervonach_vznos*, определенного в онтологии.

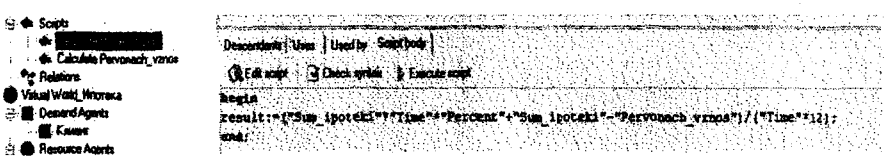


Скрипт *Calculate Pervonach_vznos* рассчитывает минимальный первоначальный взнос для каждого банка, в зависимости от суммы кредита.

Атрибут *Money_month* – сумма выплат по кредиту в месяц. Этот атрибут также является расчетной величиной, зависит от суммы и срока кредита, годовой процентной ставки, а также от первоначального взноса Клиента. Сумма выплат по кредиту в месяц (атрибут *Money_month*) может быть вычислена по следующей формуле:

$$\text{Money\_month} = (\text{Sum\_ipoteki} \cdot \text{Time} \cdot \text{Percent} + \text{Sum\_ipoteki} - \text{Pervonach\_vznos}) / (\text{Time} \cdot 12)$$

Расчет по этой формуле осуществляется с помощью скрипта *Calculate Money_month*, определенного в онтологии.



Скрипт *Calculate Money_month* рассчитывает сумму необходимых выплат по кредиту в месяц.

Дерево концептов онтологии предметной области «Ипотечное кредитование» приведено на рис. 29.

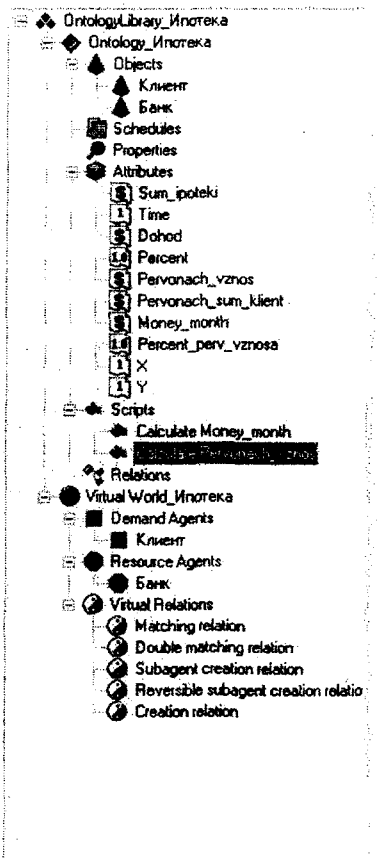


Рис. 29. Дерево концептов онтологии предметной области «Ипотечное кредитование»

Условия матчинга фактически определяются условиями получения Клиентом ипотечного кредита в Банке:

Ежемесячный доход Клиента должен быть больше или равен ежемесячным выплатам процентов и тела кредита Банку:

$$Dohod \geq Money\_month.$$

Минимальная потребная для Банка начальная сумма взноса по кредиту должна быть больше или равна первоначального взноса Клиента

$$Nach\_sum\_client \geq Nach\_sum\_bank.$$

Условия матчинга онтологии «Ипотечное кредитование» приведены на рис. 30.

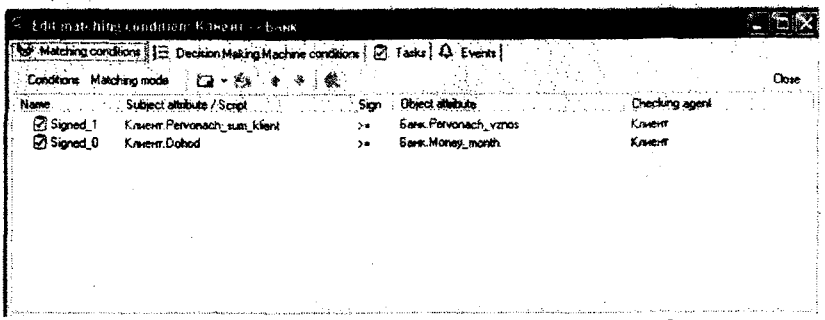


Рис. 30. Условия матчинга онтологии предметной области «Ипотечное кредитование»

Условия принятия решения определяют, каким образом будет заключаться соглашение между Клиентом и Банком в процессе матчинга. Определено два условия: минимальные требования к Клиенту со стороны Банка по первоначальному взносу и минимальным ежемесячным платежам (рис. 31).

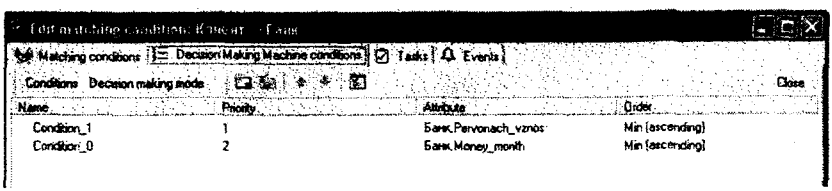


Рис. 31. Условия принятия решения в матчинге «Клиент – Банк»

На основе созданной онтологии и имеющейся информации о потребности конкретного Клиента и условиях выдачи кредитов в Самарских банках с помощью исполняющей системы может быть построена **сцена**.

Исходные данные для построения сцены

Клиент желает приобрести квартиру стоимостью 2 500 000 рублей со сроком выплаты 10 лет. Имеющиеся наличные средства у Клиента составляют сумму в размере 375 000 руб. Общий доход клиента в месяц составляет 50 000 руб.

Ресурсы (банковские условия или параметры банков):

Наименование Банка	**Банк Москвы**	**Альфа Банк**	**МДМ-Банк**	**Банк Возрождение**
Обозначение банка	Банк_1	Банк_2	Банк_3	Банк_4
Percent	0.11	0.12	0.1325	0.1025
Percent_perv_vznosa	0.15	0.15	0.15	0.20

Необходимо выбрать банк, который предоставляет наименьшую сумму выплат по кредиту в месяц.

Результаты матчинга

Наилучшим решением для Клиента является Банк_1, а Банк_4 вообще не готов удовлетворить потребности Клиента (рис. 32-33).

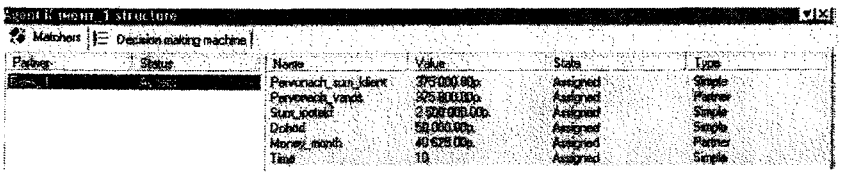


Рис. 32. Структура агента Клиент_1

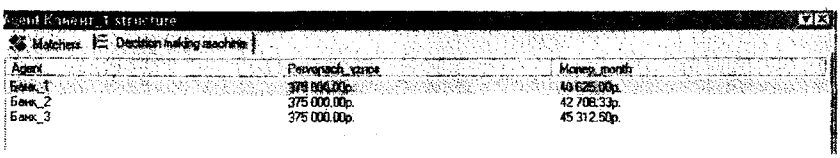


Рис. 33. Таблица принятия решения агента Клиент_1

Результаты матчинга в графическом виде (рис. 34) наглядно иллюстрируют, с какими Банками состоялись успешные переговоры и с каким Банком Клиенту выгоднее сотрудничать – получить ипотечный кредит.

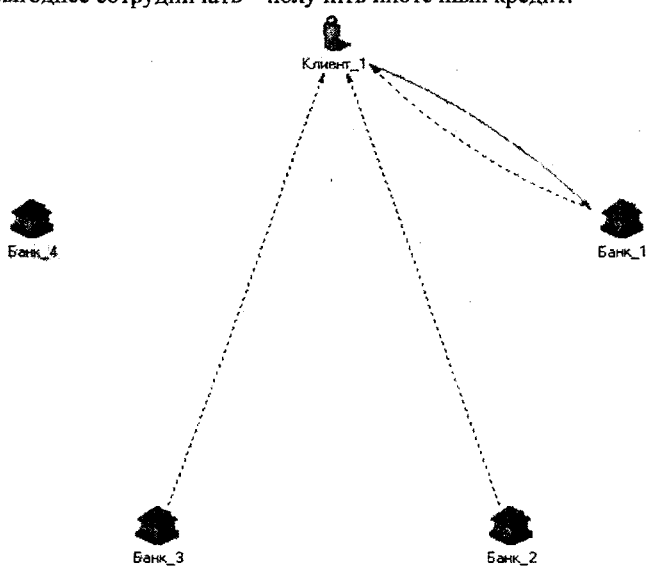


Рис. 34. Результаты матчинга

4.2. Использование онтологии в кадровой службе: «Подбор персонала»

4.2.1. Постановка задачи

Цель работы: необходимо выбрать подходящих кандидатов на вакантные должности.

Основные характеристики должностей, а также требования к кандидатам могут быть описаны в виде онтологии. При этом кандидата следует рассматривать как проект или заказ, а вакансии – как ресурсы. На основании результатов процесса поиска взаимного соответствия между заказом и ресурсами (матчинга) принимаются или пересматриваются решения о бронировании или освобождении ресурсов (т.е. устанавливаются связи между заказом и адекватными ему ресурсами). Тем самым выполняется выбор кандидатов, отвечающих требованиям компании. На основании исходных данных по желаемой вакансии, опыту работы, результатам тестирования, требуемой заработной плате кандидатов необходимо выбрать подходящего кандидата на конкретную должность и рассчитать по этим исходным данным заработную плату, которую компания может предложить претенденту.

4.2.2. Решение задачи

Была рассмотрена конкретная организация НПЦ «Экология». В 2007 году в этой организации производился поиск кадров на две должности (по одному человеку на место).

№	Вакансия
1	Офис-менеджер
2	Координатор

Характеристиками кандидатов являются их стаж, результаты тестирования и их предпочтения по поводу должности и заработной платы. Причем опыт работы на должность офис-менеджера не обязателен, но приветствуется.

Имеется пять кандидатов на вакантные должности:

Кандидат №	Должность	Тест	Опыт работы	з/п, руб
1	1	50	0	6000
2	1	60	0	5000
3	2	45	2	6000
4	1	30	1	10000
5	2	50	2	10000

На данном предприятии принято условно рассчитывать заработную плату, исходя из ставки = 3000 руб. Расчет зарплаты осуществляется на основе данных тестирования и опыта работы кандидата (см. таблицу).

Вакансия	Определение количества ставок по каждой вакансии
1	1 ставка – результат тестирования до 30 баллов 3 ставки – от 30 до 50 (не включительно) 4 ставки – более 50 баллов
2	2 ставки – результат тестирования до 50 баллов, нет опыта работы 4 ставки – от 50 до 100 баллов, нет опыта работы 3 ставки – результат тестирования до 50 баллов, опыт работы до 2-х лет 5 ставок – от 50 до 100 баллов, опыт работы до 2-х лет 4 ставки – результат тестирования до 50 баллов, опыт работы свыше 2-х лет 6 ставок – от 50 до 100, опыт работы свыше 2-х лет

Концепт «объект» – это сущность, которая присутствует в мире, описанном в онтологии.

Необходимо создать два концепта «объект»:

Cand – (кандидат) с атрибутами:

_ Name_cand (имя кандидата);

_Exp_cand (опыт работы);

_ Test (результаты теста);

_ Name_dolg (желаемая должность);

_ Salary_want (заработная плата, заявленная кандидатом);

Salary_real (расчетная заработная плата).

Примечание: имена атрибутов, которые являются исходными данными для расчета, начинаются с нижнего подчеркивания.

Vac – (вакансия) с атрибутами:

_ Name_vac (название вакансии);

_Stavka (ставка).

На рис. 35 представлено дерево концептов дескриптивной онтологии и онтологии виртуального мира предметной области «Подбор персонала».

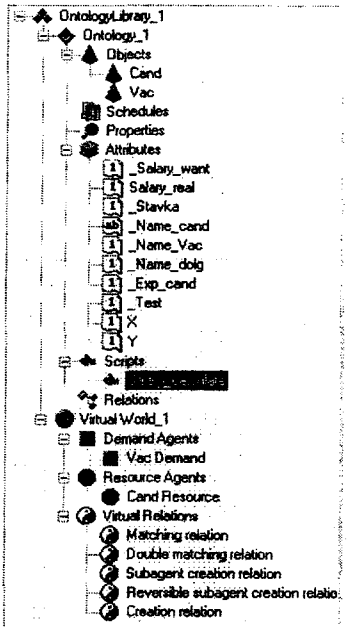


Рис. 35. Дерево концептов онтологии предметной области «Подбор персонала»

Онтология предметной области «Подбор персонала» в виде семантической сети представлена на рис. 36.

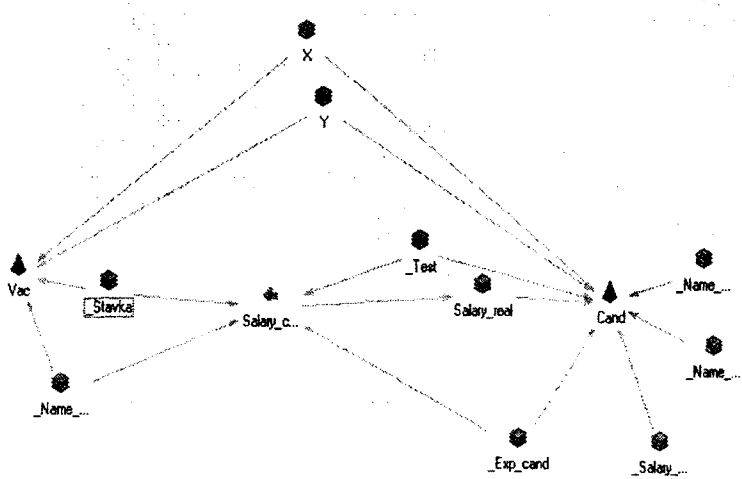


Рис. 36. Представление дерева концептов онтологии предметной области «Подбор персонала» в виде семантической сети

Вычисление количества ставок производится с помощью скрипта *Salary_calculate* (рис. 37).

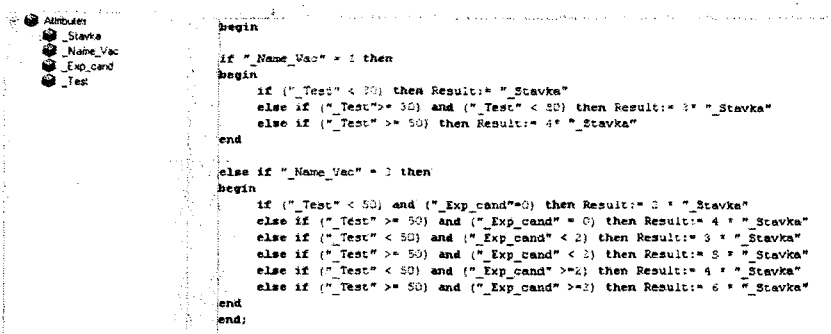


Рис. 37. Тело скрипта *Salary_calculate*

В задаче выбора кандидата для определенной должности должны удовлетворяться следующие два условия (рис. 38), являющиеся условиями матчинга:

1. Должность, выбранная кандидатом, должна совпадать с одной из вакансий;

2. Заработная плата, заявленная кандидатом, должна быть меньше расчетной.

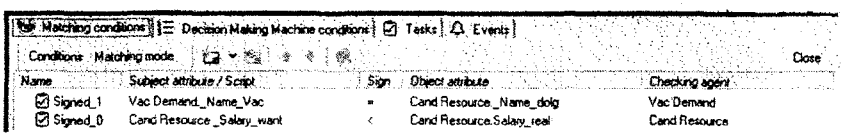


Рис. 38. Условия матчинга онтологии предметной области «Подбор персонала»

Условия принятия решения (рис. 39) определяются тем, что из всех кандидатов, прошедших условия отбора, выбирается один, по следующим признакам:

– максимальный опыт работы;

– максимальный результат тестирования.

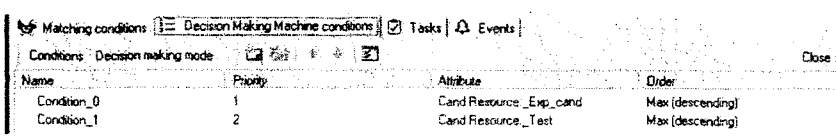


Рис. 39. Условия принятия решения онтологии предметной области «Подбор персонала»

В результате матчинга необходимо получить следующие результаты:

– должны быть выбраны кандидаты, соответствующие условиям;

– должен быть определен наиболее подходящий кандидат;

– должен быть произведен расчет заработной платы подходящим кандидатам.

Для того чтобы решить задачу подбора персонала, в исполняющей системе создается сцена, в которой представлены пять агентов кандидатов и два агента вакансии. С помощью инспектора агентов устанавливаются указанные в таблице значения атрибутов для агентов кандидатов и вакансий.

Процесс матчинга начинается с того, что агенты вакансии проверяют значения атрибутов агентов кандидатов и выбирают кандидатов, у которых значения атрибутов соответствуют ограничениям по должности и предпочтительной зарплате.

Выявленные соответствия заданным условиям показаны на рис. 40. На должность «1» претендовали кандидаты 1, 2, 4. Кандидат 4 не подходит на должность офис-менеджера, так как заявленная зарплата на 7000 руб превышает расчетную. Кандидаты 1 и 2 удовлетворяют обоим условиям матчинга (пунктирные стрелки), но резервируется кандидат 2, т.к. он имеет больший тестовый балл (сплошная линия). На должность «2» претендовали кандидаты 3 и 5. Они удовлетворяют обоим условиям матчинга (пунктирные стрелки), но резервируется кандидат 5, т.к. он, даже при одинаковом опыте работы с кандидатом 3, имеет больший тестовый балл (сплошная линия).

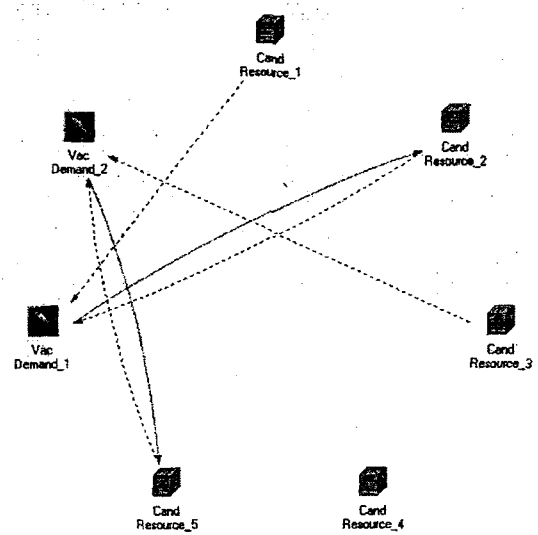


Рис. 40. Результаты матчинга

На рис. 41-46 приведены структура агентов вакансий и кандидатов, а также таблицы принятия решения агентами.

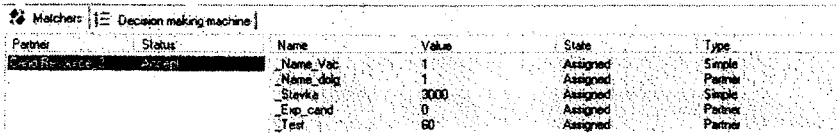


Рис. 41. Структура агента *Vac Demand_1*

Рис. 42. Таблица принятия решения агента *Vac Demand_1*

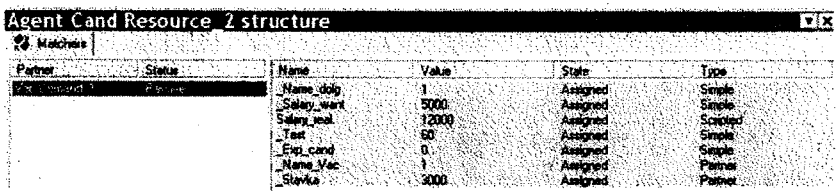


Рис. 43. Структура агента *Cand Resource_2*

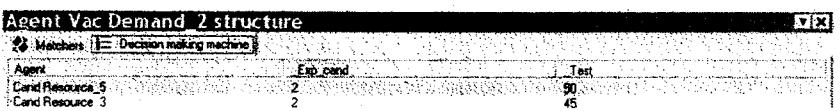


Рис. 44. Таблица принятия решения агента *Cand Resource_2*

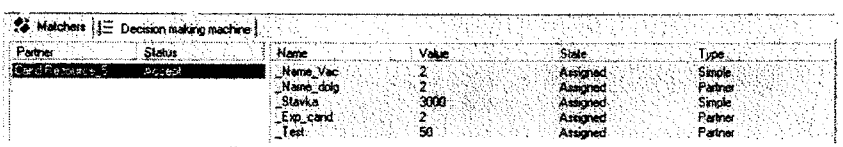


Рис. 45. Структура агента *Cand Resource_5*

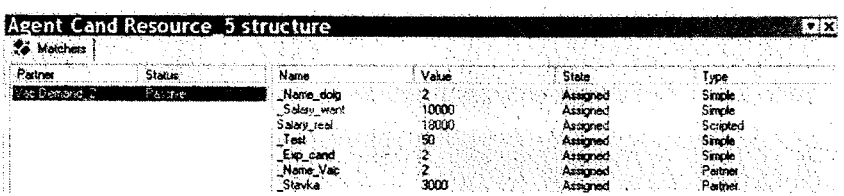


Рис. 46. Таблица принятия решения агента *Cand Resource_5*

Отчет по результатам моделирования сцены представлен на рис. 47.

Cand

Name	Salary_want	Salary_real	Name_cand	Name_dolg	Exp_cand	Test
Cand_1	6000	0	Cand_1	1	0	50
Cand_2	5000	0	Cand_2	1	0	60
Cand_3	6000	0	Cand_3	2	2	45
Cand_4	10000	0	Cand_4	1	1	30
Cand_5	10000	0	Cand_5	2	2	50

Vac

Name	Stavka	Name_Vac
Vac_1	3000	1
Vac_2	3000	2

Рис. 47. Отчет по сцене

4.3. Использование онтологии в университете: «Приемная кампания»

4.3.1. Постановка задачи

Университету необходимо осуществить плановый набор студентов на три факультета. Для каждого факультета определен перечень предметов, по которым проводятся экзамены.

- Физико-математический факультет (№ 1): русский язык, математика, физика.
- Химико-биологический факультет (№ 2): русский язык, химия, биология.
- Историко-филологический факультет (№ 3): русский язык, история.

Прием студентов осуществляется на основании среднего значения баллов по указанным предметам. На каждый факультет может быть зачислено ограниченное количество человек.

При зачислении, в первую очередь, учитывается личное желание студента поступить на тот или иной факультет. Затем рассчитывается средний балл по тем предметам, которые необходимы для поступления на данный факультет. Далее полученный результат сравнивается со средними баллами остальных студентов, желающих учиться на этом же факультете. И, наконец, студенты зачисляются на факультет в порядке убывания их средних баллов до тех пор, пока не закончатся вакантные места.

4.3.2. Решение задачи

В данной задаче определены следующие концепты «объект»:
– Student (объект-заказ),
– MATEMATIKOFIZICHESKII (объект-ресурс),
– XIMIKOBIOLOGICHESKII (объект-ресурс),
– ISTORIKOFILOLOGICHESKII (объект-ресурс).

Концепт *Student* (студент) имеет следующие параметры:
– F_I_O (фамилия, имя и отчество студента),
– Ball_rus (балл по русскому языку),
– Ball_matem (балл по математике),
– Ball_ximia (балл по химии),
– Ball_fizika (балл по физике),
– Ball_biologia (балл по биологии),
– Number_Fak_Studenta (номер факультета, на который хочет поступить студент),
– CR_Ball_M (средний балл, набранный студентом, для физико– математического факультета),
– CR_Ball_X (средний балл, набранный студентом, для химико-биологического факультета),
– CR_Ball_I (средний балл, набранный студентом, для историко-филологического факультета).

Концепт *MATEMATIKOFIZICHESKII* (физико-математический факультет) имеет следующие атрибуты:
– Number_Fakylteta (номер факультета),
– Name_Fakylteta (название факультета),
– Prox_ball_M (проходной балл на факультет),
– Kol-vo_mest_M (количество мест на факультете),
– Sender_names_M (список поступивших студентов),
– USER_M (количество поступивших студентов),
– U_M (процент загруженности факультета).

Концепт *XIMIKOBIOLOGICHESKII* (химико-биологический факультет) имеет следующие атрибуты:
– Number_Fakylteta (номер факультета),
– Name_Fakylteta (название факультета),
– Prox_ball_ X (проходной балл на факультет),
– Kol-vo_mest_X (количество мест на факультете),
– Sender_names_X (список поступивших студентов),
– USER_X (количество поступивших студентов),
– U_X (процент загруженности факультета).

Концепт *ISTORIKOFILOLOGICHESKII* (историко-филологический факультет) имеет следующие атрибуты:
– Number_Fakylteta (номер факультета),
– Name_Fakylteta (название факультета),
– Prox_ball_I (проходной балл на факультет),
– Kol-vo_mest_I (количество мест на факультете),
– Sender_names_I (список поступивших студентов),
– USER_I (количество поступивших студентов),
– U_I (процент загруженности факультета).

Для расчета среднего балла студента определены скрипты.

Средний балл студента химико-биологического факультета рассчитывается с помощью скрипта *Calc_Cr_Ball_X* (рис. 48).

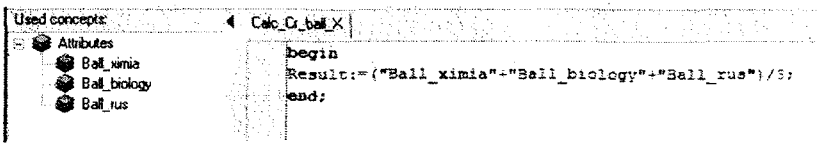


Рис. 48. Атрибуты и тело скрипта *Calc_Cr_Ball_X*

Средний балл студента физико-математического факультета рассчитывается с помощью скрипта *Calc_Cr_Ball_M* (рис. 49).

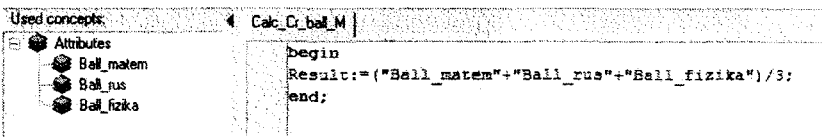


Рис. 49. Атрибуты и тело скрипта *Calc_Cr_Ball_M*

Средний балл студента историко-филологического факультета рассчитывается с помощью скрипта *Calc_Cr_Ball_I* (рис. 50).

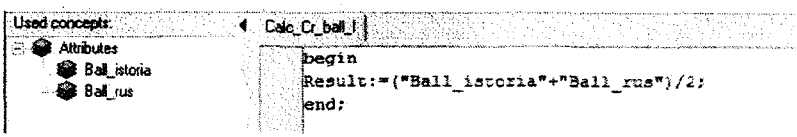


Рис. 50. Атрибуты и тело скрипта *Calc_Cr_Ball_I*

Дерево концептов дескриптивной онтологии *Ontology_Universitet* и онтологии виртуального мира *Virtual World_Universitet* представлено на рис. 51.

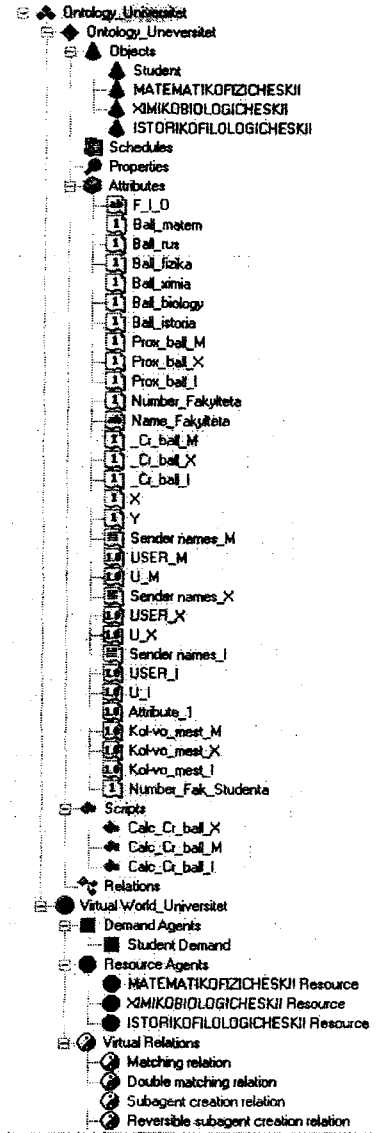


Рис. 51. Дерево онтологии предметной области «Приемная кампания»

Онтология в виде семантической сети приведена на рис. 52.

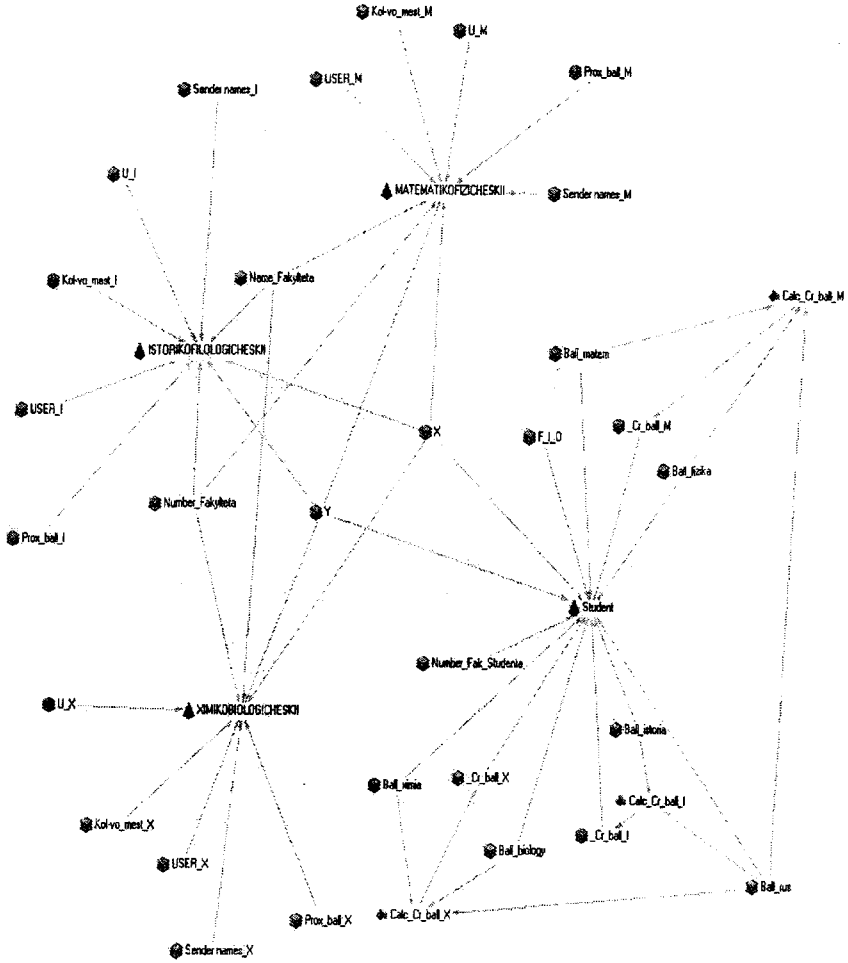


Рис. 52. Онтология предметной области «Приемная кампания» в виде семантической сети»

Условия матчинга, условия принятия решения и обработчики событий для матчинга «Студент – MATEMATIKOFIZICHESKII Resource» приведены на рис. 53.

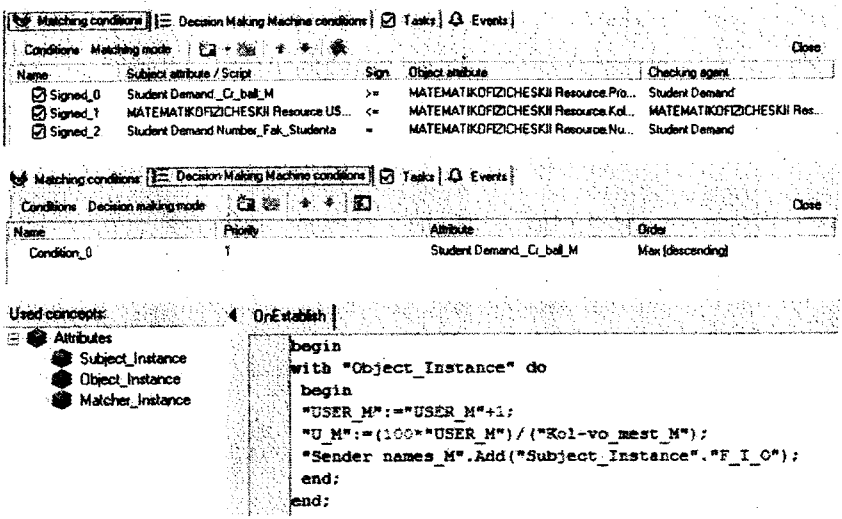


Рис. 53. Условия матчинга, условия принятия решения и обработчики событий в матчинге «Студент – Физико-математический факультет»

Условия матчинга, условия принятия решения и обработчики событий для матчинга «Студент – XIMIKOBIOLOGICHESKII Resource» приведены на рис. 54.

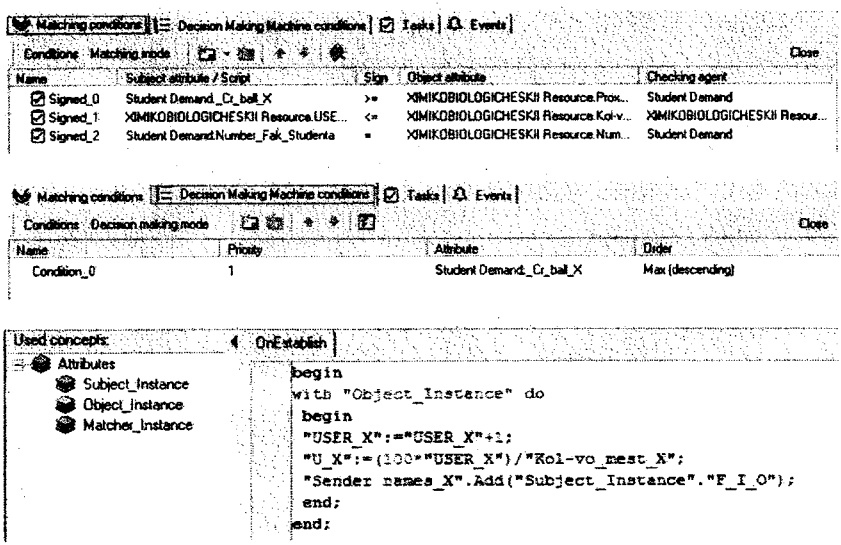


Рис. 54. Условия матчинга, условия принятия решения и обработчики событий в матчинге «Студент – Химико-биологический факультет»

Условия матчинга, условия принятия решения и обработчики событий для матчинга «Студент – ISTORIKOFILOLOGICHESKII Recource» приведены на рис. 55.

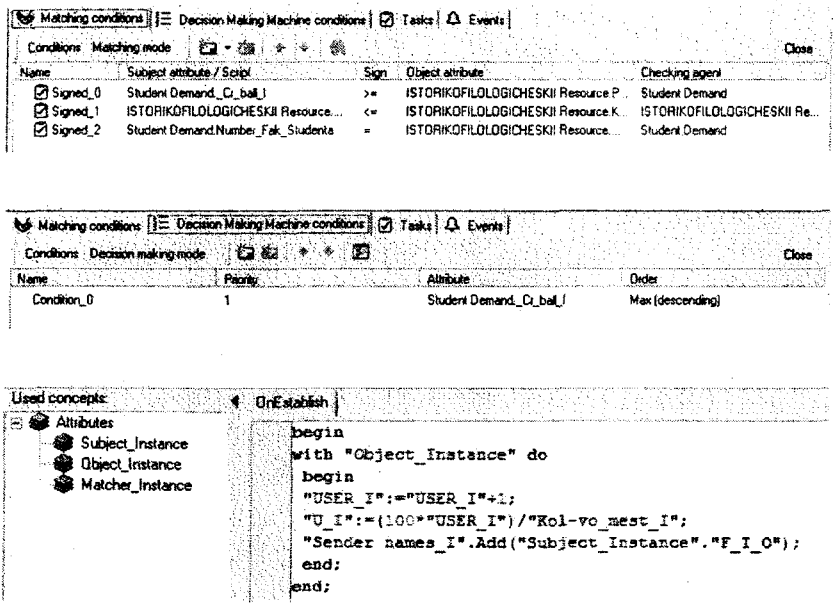


Рис. 55. Условия матчинга, условия принятия решения и обработчики событий в матчинге «Студент – Историко-филологический факультет»

В окне физического мира созданы 15 объектов заказа (агенты студентов) и 3 агента ресурса (агенты факультетов). Онтологическая сцена предметной области «Приемная кампания» показана на рис. 56.

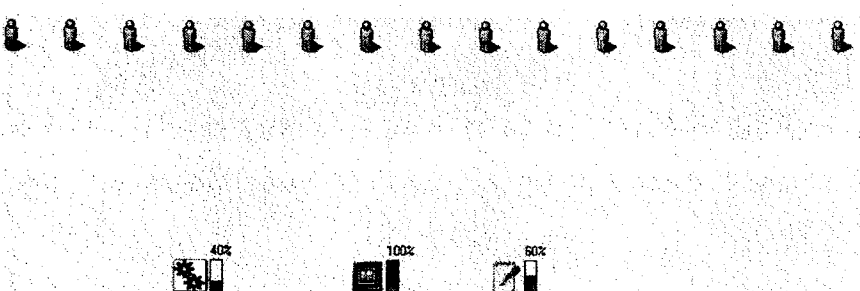


Рис. 56. Онтологическая сцена предметной области «Приемная кампания»

Значения атрибутов объектов *Student* приведены в табл. 2, а атрибутов факультетов – в табл. 3.

Таблица 2

Значения атрибутов объектов *Student*

№	F_I_O	Number_ Fak	Ball_rus	Ball_mat	Ball_xim	Ball_fiz	Ball_istor	Ball_biol
1	Student_1	1	75	62	89	78	45	25
2	Student_2	1	62	85	35	86	75	65
3	Student_3	1	35	45	65	25	75	65
4	Student_4	1	85	76	19	68	23	21
5	Student_5	1	69	78	62	63	48	65
6	Student_6	2	82	86	81	85	65	57
7	Student_7	2	89	52	25	74	67	65
8	Student_8	2	87	0	64	0	0	62
9	Student_9	2	65	84	86	37	36	74
10	Student_10	2	75	48	65	84	96	56
11	Student_11	3	32	84	45	25	75	86
12	Student_12	3	14	12	19	13	18	12
13	Student_13	3	45	95	35	62	45	70
14	Student_14	3	56	44	50	62	78	45
15	Student_15	3	43	44	39	85	65	75

Таблица 3

Значения атрибутов объектов «*Факультет*»

Name_Fak	Number_Fak	Prox_ball_Fak	Kol-vo_mest_Fak
MATEMATIKOFIZICHESKII	1	60	10
XIMIKOBIOLOGICHESKII	2	50	5
ISTORIKOFILOLOGICHESKII	3	50	5

Процесс матчинга начинается с того, что агенты заказа проверяют значения атрибутов агентов ресурсов и выбирают ресурс, у которого значения атрибутов соответствуют ограничениям по заданным выше параметрам.

Результаты матчинга показаны на рис. 57.

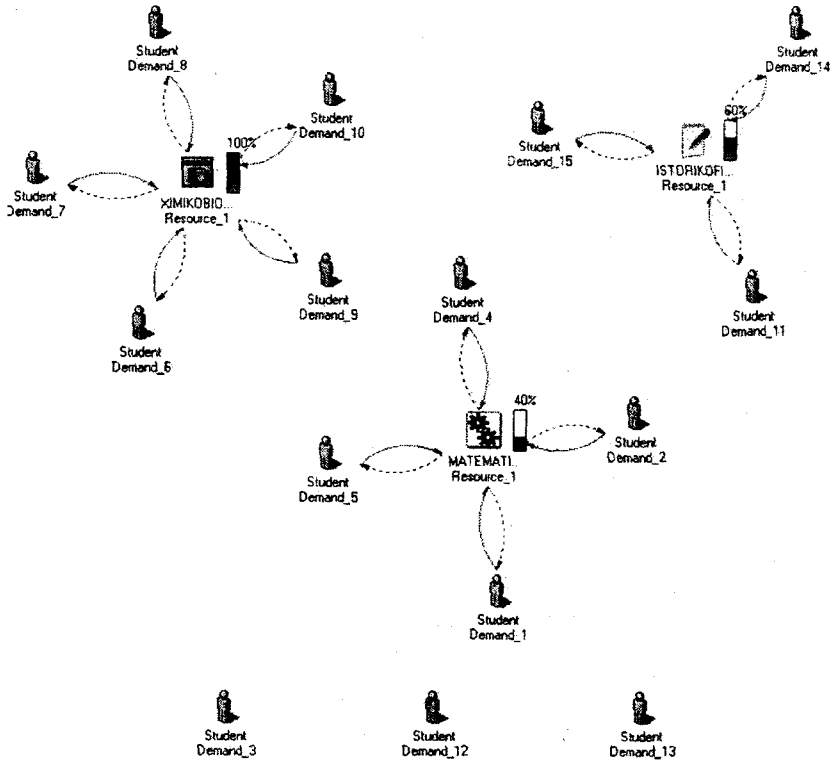


Рис. 57. Результаты матчинга

Результаты матчинга:

- на физико-математический факультет поступило 4 студента,
- на химико-биологический факультет поступило 5 студентов,
- на историко-филологический факультет поступило 3 студента,
- 3 студента по своим баллам не поступили ни на один из факультетов.

4.4. Использование онтологии в туристической фирме: «Выбор тура»

4.4.1. Постановка задачи

Необходимо на туристическом рынке города Тольятти подобрать тур, отвечающий следующим требованиям клиента:

– дата заезда (число, месяц, год);
– длительность тура;
– стоимость тура (руб.).

Для этого необходимо выбрать в базе данных туристические агентства, удовлетворяющие всем условиям клиента. Среди этих турагентств выбрать турагентство, обеспечивающее минимальную стоимость тура.

4.4.2. Проектирование дескриптивной онтологии

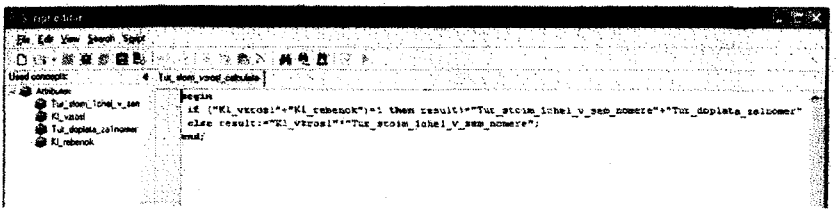
Создается библиотека онтологий – *turagenstva Togliatti*. В ней создается дескриптивная онтология – *Ontology_Turagenstva*. После создания дескриптивной онтологии необходимо создать два концепта «объект»:

- *Klient* – клиент туристического агентства, с атрибутами:
  – Kl_zaezd – дата заезда клиента,
  – Kl_summa – сумма денег, имеющаяся у клиента,
  – Kl_name – имя клиента,
  – Kl_dn – число дополнительных дней,
  – Kl_vzrosl – количество взрослых,
  – Kl_rebenok – количество детей,
  – Kl_period – длительность тура.
- *Turagenstvo* – туристическое агентство с атрибутами:
  – Tur_name – название турагентства и тура,
  – Name_gorod – название города тура,
  – Tur_proezd – включен ли проезд в стоимость тура,
  – Tur_transport – если проезд входит в стоимость тура, то на каком транспорте добираться,
  – Tur_pitanie – входит ли питание в стоимость тура,
  – Tur_doplata_za1nomer – доплата за одноместное проживание,
  – Tur_gost_zvezd – количество звезд у гостиницы тура,
  – Tur_period – продолжительность тура,
  – Tur_stoim_1chel_vsem_nomere – стоимость тура для одного человека в семейном номере,
  – Tur_stoim_1_rebenok – стоимость тура для ребенка младше 14 лет,
  – Tur_stoim_rebenok – стоимость тура для всех детей,
  – Tur_stoim_vzrosl – стоимость тура для всех взрослых,
  – Tur_stoim_dn_v1mest – стоимость дополнительного дня проживания в гостинице в одноместном номере,
  – Tur_stoim_dn_v_sem –стоимость дополнительного дня проживания в гостинице в семейном номере,
  – Tur_stoim_dopl_dn –общая стоимость дополнительных дней,
  – Tur_summa – общая стоимость тура.

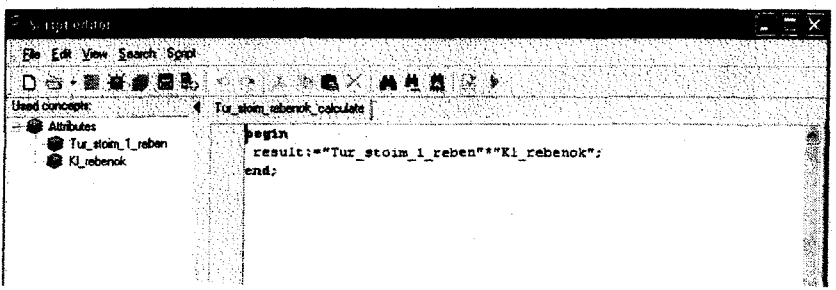
Далее создаются вышеперечисленные концепты «атрибут» и связи между соответствующими концептами «объект» и «атрибут».

Создание концепта «скрипт» необходимо для вычисления стоимости тура по заданным параметрам. Создаются 4 скрипта:

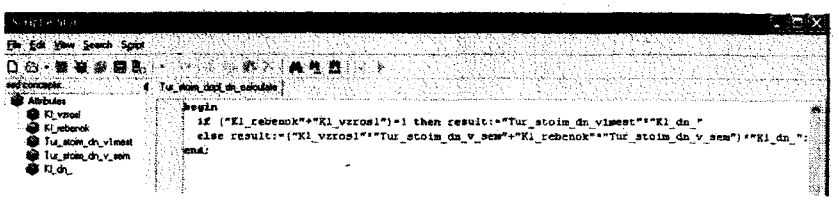
- *Tur_stoim_vzrosl_calculate* – общая стоимость тура взрослых;



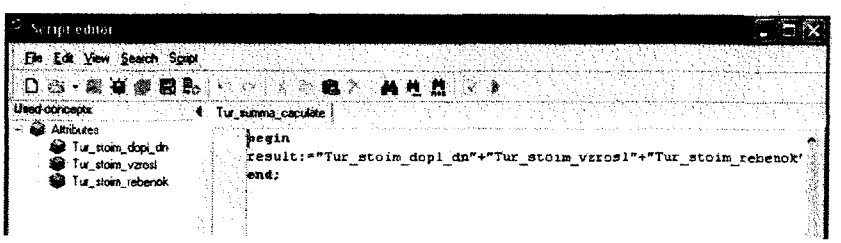
- *Tur_stoim_rebenok_calculate* – общая стоимость тура для детей;



- *Tur_stoim_dopl_dn_calculate* – общая стоимость дополнительных дней;



- *Tur_summa_calculate* – общая стоимость тура;



4.4.3. Проектирование онтологии мира заказов и ресурсов

Создается онтология мира заказов и ресурсов *Virtual World_Turagenstva*, в ней создаются концепт «агент заказа» – *Klient Demand* и концепт «агент ресурса» – *Turagenstvo Resourse*. Далее необходимо создать отношения матчинга. Поиск туристических агентств необходимо выполнять по значениям атрибутов стоимости тура, его продолжительности и дате заезда.

Условия матчинга для поиска турагентства по продолжительности тура, дате начала заезда и стоимости тура приведены на рис. 58.

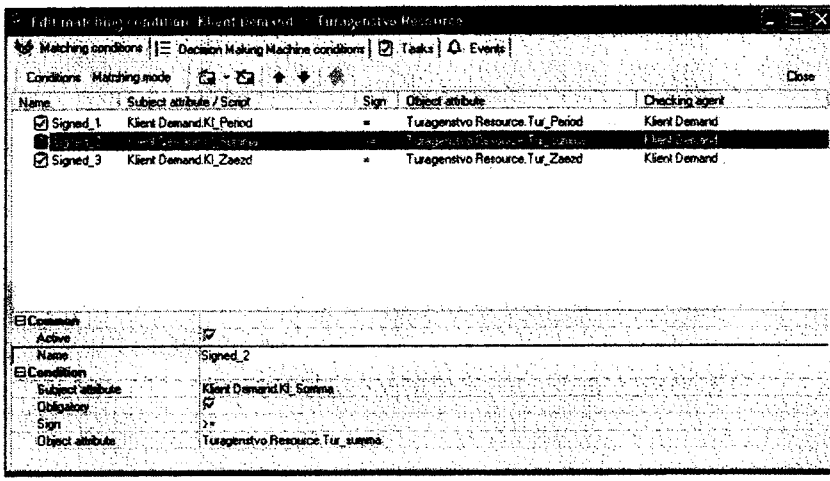


Рис. 58. Условия матчинга для поиска турагентства

Условия принятия решения предназначены для работы машины принятия решений и позволяют агенту выбрать одно из множества возможных предложений (матчингов) от других партнеров. В данном случае выбор происходит по условию минимальной общей стоимости тура (рис. 59).

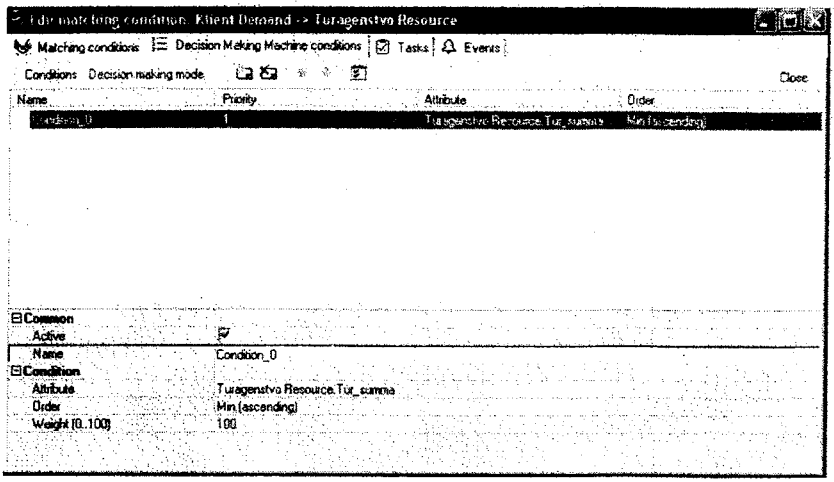


Рис. 59. Условия принятия решения для поиска турагентства

4.4.4. Создание онтологической сцены

В окне физического мира создается два агента клиента и двадцать агентов туристических агентств. С помощью инспектора агентов устанавливаются значения атрибутов для агентов (рис. 60).

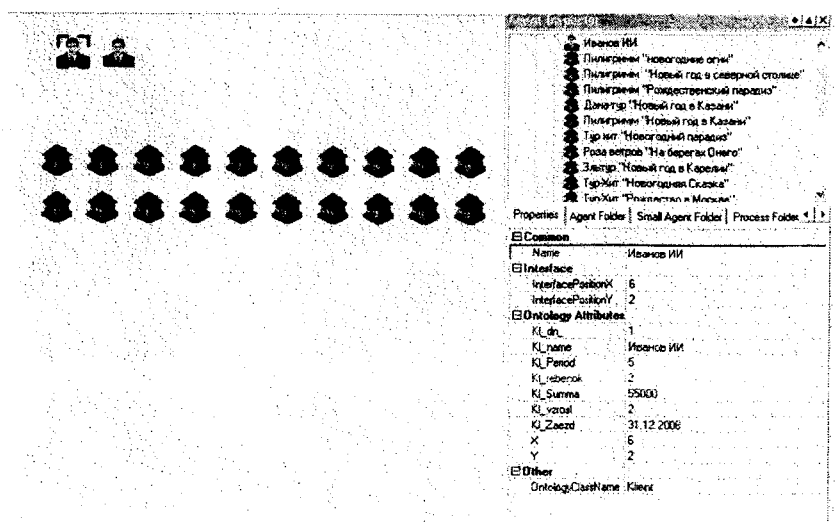


Рис. 60. Создание онтологической сцены предметной области «Выбор тура»

В окне виртуального мира можно увидеть результаты матчинга (пунктирная линия означает, что два объекта согласились на матчинг, сплошная, что заказ зарезервировал ресурс) (рис. 61).

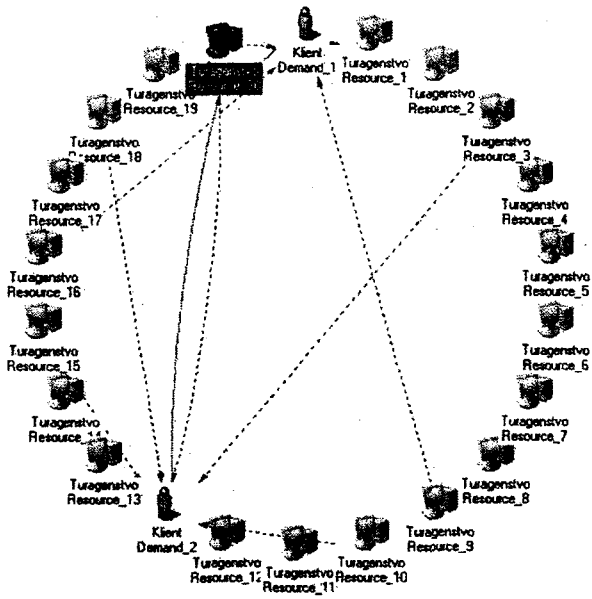


Рис. 61. Результаты матчинга

В результате матчинга была выполнена следующие операции резервирования:

- Klient Demand_1 – Turagenstvo Resource_19,
- Klient Demand_2 – Turagenstvo Resource_20.

Структура агента *Klient Demand_1* показана на рис. 62.

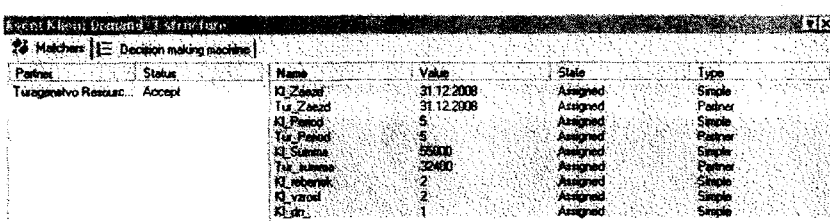


Рис. 62. Структура агента Klient Demand_1

Таблица принятия решений агента *Klient Demand_1* показана на рис. 63.

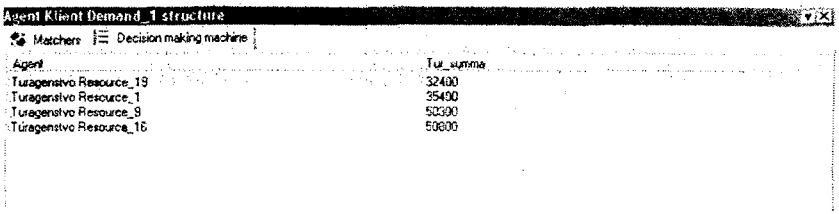


Рис. 63. Таблица принятия решений агента Klient Demand_1

Структура агента ресурса *Turagenstvo Resource_19* показана на рис. 64.

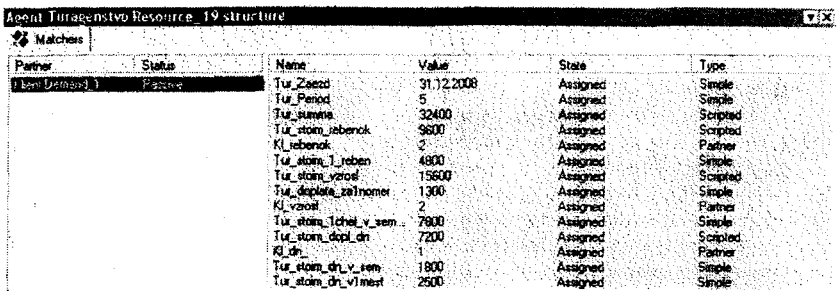


Рис. 64. Структура агента ресурса Turagenstvo Resource_19

ЗАКЛЮЧЕНИЕ

Учебное пособие «Основы построения мультиагентных систем, использующих онтологию» посвящено рассмотрению мультиагентного подхода к использованию онтологии в различных областях производственной сферы и подкрепляется комплексом лабораторнях работ по решению задач выбора параметров самолета, а также логистики воздушного флота.

ГЛОССАРИЙ

Agent (Агент) – небольшая компьютерная программа, способная решать задачи во взаимодействии с другими агентами. Агент предназначен для представления объекта предметной области, который может быть неживым (грузовик), живым (водитель грузовика) или спецификацией (заказ) и для ведения переговоров от имени этого объекта с агентами других объектов с целью достижения наилучшего решения задачи.

Agent actions (Действия агента) – элементарные шаги, которые агент может предпринимать для достижения цели.

Agent Negotiation Log (Лог переговоров агента) – хранилище записей, определяющих все переговоры агента в процессе принятия решения. Это средство предоставляет возможность тщательного анализа истории функционирования мультиагентной системы.

Agent Personality («Личность» агента) реализует PCX-cycle – жизненный цикл агента. Представляет собой главную активность агента, управляющую всеми остальными его активностями.

Agent sensors (Органы восприятия агента) позволяют агенту получать информацию о сцене, сообщения от других агентов.

Agent swarm (Сообщество агентов) – группа агентов, участвующих в переговорах друг с другом с целью выполнения поставленной перед ними задачи.

Constraints (Ограничения) – ограничения предметной области, которым агент должен следовать при построении своих планов. Задают правила, которым агент обязан следовать всегда – например, физические ограничения на выполнение некоторых операций.

Criteria (Критерии) – количественные оценки степени достижения цели, которые агент использует для оценки своей позиции на пути к достижению цели.

Decision Making Table (DMT) (Таблица принятия решений) – структура, в которой хранятся варианты принятия решений для их последующего сравнения и выбора наилучшего решения в смысле целей и критериев.

Demand (Заказ) – входящий запрос о товаре или услуге, который должен быть обработан компанией, предоставляющей ресурсы.

Demand Agent (Агент заказа) – агент, представляющий некоторое требование (заказ, задачу или проект).

Event (Событие) – определяет некоторый свершившийся факт, в том числе, изменение существующего состояния системы.

Goal (Цель) – примитив логики принятия решения, задающий направление деятельности агента.

Matching (Матчинг) – один из этапов работы агентов, на котором происходит поиск взаимного соответствия между агентами, на основании которого принимаются или пересматриваются решения о бронировании или освобождении ресурсов.

Multi Agent Engine (MAE) – модуль, определяющий структуру агента, способы взаимодействия агентов и протоколы ведения переговоров. Он включает библиотеку Java программ для поддержки разработки приложений, а также для поддержки работы виртуального рынка и сообщества агентов во время исполнения. Библиотека не зависит от приложений (например, в планировании расписаний и электронной коммерции можно использовать один и тот же MAE).

Negotiations protocol (Протокол переговоров) – совокупность определений (соглашений, правил), регламентирующих формат и процедуры обмена информацией между двумя или несколькими независимыми агентами.

Resource (Ресурс) – средства, которые компания может использовать для выполнения заказа.

Resource Agent (Агент ресурса) – агент, представляющий ресурс (например, оператор, грузовик, механизированное средство, топливо, маршрут, ячейку хранения).

Scenario (Сценарий) – спланированная агентом последовательность действий.

Virtual Market (VM), Virtual Market Engine (VME) (Виртуальный рынок) представляет собой библиотеку для создания мультиагентных планировщиков. В основе мультиагентных планировщиков лежат переговоры о размещении между различными агентами возможностей (ресурсов) и потребностей (заказов). При этом принятие решений происходит на основе микроэкономики виртуального рынка, т.е. модели перераспределения прибыли и/или издержек и учета их в критериях агентов таким образом, чтобы достичь целей агентов на пространстве виртуального рынка.

Ontology (Онтология) – это формализованное описание знаний о предметной области. Онтология состоит из классов объектов (заказ, расписание, водитель, грузовик, паллета, склад), классов отношений между объектами (грузовик управляется водителем, паллета хранится на складе), атрибутов объектов (загрузка грузовика, допустимые часы работы водителя) и скриптов, определяющих правила и ограничения предметной области и поведения агентов.

Attribute (Атрибут) – это характеристика концепта, которая используется для его уточнения. Атрибуты бывают качественные (цвет, марка товара) и количественные (масса, грузоподъемность, скорость, стоимость). Атрибут характеризуется символьным именем, типом, единицей измерения, списком возможных значений, значением по умолчанию.

Concept (Концепт) – понятие, термин, сущность, категория предметной области, совокупность которых используется для описания фрагментов окружающего мира. Существуют следующие виды концептов: классы объектов и классы отношений. Класс объектов – это понятие, описывающее некоторую совокупность объектов одного рода. Например, для предметной области «Транспортная логистика» могут быть определены концепты «Перевозчик», «Груз», «Заказчик». Класс отношения – понятие, определяющее семантическую связь между классами в онтологии. Примером класса отношения является класс отношения «Перевозить», который связывает классы «Перевозчик» и «Груз».

Instance (Экземпляр) – это элемент класса объектов или класса отношения. Экземпляр класса объектов – это элемент класса объектов, для которого определены значения атрибутов. Например, конкретный грузовик с именем Truck1, скоростью 50 км/ч и грузоподъемностью 10 т представляет собой экземпляр класса Грузовик. Экземпляр класса отношений – это конкретное отношение между двумя или более объектами, которое определяет взаимосвязи между экземплярами классов объектов, называемыми сторонами отношения. Например, если в онтологии указано отношение «Перевозить» между классами «Транспортное средство» и «Груз», то в сцене возможна установка отношений между экземплярами класса «Транспортное средство» и между экземплярами класса «Груз».

Scene (Сцена) – это модель реальной ситуации, обрабатываемой системой. Онтологическая сцена состоит из экземпляров объектов и отношений между ними. Экземпляры объектов в сцене представляются агентами.

Semantic Net (Семантическая сеть) – это графическое представление знаний в виде взаимосвязанных узлов и дуг. Компьютерные реализации семантических сетей, главным образом, были разработаны для применения в области искусственного интеллекта и машинного перевода, но ранние версии использовались в философии, психологии, лингвистике.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Тарасов, В.Б.* Агенты, многоагентные системы, виртуальные сообщества: стратегическое направление в информатике и искусственном интеллекте / *В.Б. Тарасов* // Новости искусственного интеллекта, № 2, 1998, с. 5-63.

2. *Городецкий, В.И.* Многоагентные системы (обзор) / *В.И. Городецкий, М.С. Грушинский, А.В. Хабалов* // Новости искусственного интеллекта, №2, 1998, с. 64-116.

3. *Гаврилова, Т.А.* Базы знаний интеллектуальных систем: Учебник для вузов / *Т.А. Гаврилова, В.Ф. Хорошевский.* – СпБ: Питер, 2001. – 384 с.

4. *Скобелев, П.О.* Открытые мультиагентные системы для поддержки процессов принятия решений при управлении предприятиями. // Труды Самарского научного центра РАН, 2001, с. 215-227.

5. *Скобелев, П.О.* Открытые мультиагентные системы для холонических предприятий / *П.О. Скобелев* // Новости искусственного интеллекта. № 3, 2001.

6. Инструментальные средства разработки мультиагентных приложений (модель RC-десктопно-ориентированная версия с поддержкой асинхронного режима). Конструктор онтологий. Руководство пользователя. – Самара: Magenta Technology, 2005. – 40 с.

7. Инструментальные средства разработки мультиагентных приложений (модель RC-десктопно-ориентированная версия с поддержкой асинхронного режима). Исполняющая система. Руководство пользователя. – Самара: Magenta Technology, 2005. – 60 с.