Министерство образования и науки Республики Казахстан
Костанайский государственный университет имени А. Байтурсынова

Кафедра программного обеспечения

Салыкова О.С.

Фундаментальные основы распределенных интеллектуальных систем

Учебно-методическое пособие

Костанай, 2019

УДК 004.4
ББК 32.973.26-018я73
С 32
Составитель:
Салыкова О.С., к.т.н., доцент кафедры программного обеспечения
Рецензенты:
Жунусов Куат Маратович, к.э.н., доцент кафедры информационных технологий Костанайского инженерно - экономического университета имени М.Дулатова
Иванова Ирина Владимировна , к.п.н., доцент кафедры программного обеспечения Костанайского государственного университета имени А. Байтурсынова
Исмаилов Арман Оразалиевич, к.т.н., доцент кафедры программного обеспечения Костанайского государственного университета имени А. Байтурсынова
Салыкова О.С.

С 21 Фундаментальные основы распределенных интеллектуальных систем**. Учебно-**методическое пособие. – Костанай: КГУ имени А.Байтурсынова, 2019. - 65 с.

ISBN 978-601-7985-46-2

Учебно-методическое пособие охватывает содержание дисциплины «Фундаментальные основы распределенных интеллектуальных систем», которая читается докторантам специальности 6D070400 – Вычислительная техника и программное обеспечение. Учебно-методическое пособие включает в себя теоретический и практический материал для подготовки докторантов при управлении сложными и распределенными инфраструктурами больших предприятий и технологических комплексов, включая управление транспортными потоками наземного транспорта крупных мегаполисов, системами аэропортов и морских портов, управления сложными автономными мобильными роботами и системами специального назначения. Знания, полученные в результате освоения дисциплины, помогут привить навыки в управлении распределенными роботами и специальными автономными мобильными системами.

УДК 004.4

ББК 32.973.26-018я73

С32

Утверждено и рекомендовано к изданию Учебно-методическим советом Костанайского государственного университета имени А.Байтурсынова от 29.04. 2019 г. протокол №3

ISBN 978-601-7985-46-2

СОДЕРЖАНИЕ

Тема 1. Инженерия знаний
Цель: Изучить основные понятия инженерии знаний.

План:
1. Основные понятия.
2. Представление знаний.

1.1Основные понятия

Инженерия знаний – научная дисциплина, связанная с разработкой систем, основанных на знаниях – интеллектуальных систем. Она изучает методы и средства извлечения, представления, структурирования, формализации и использования знаний. Человек в свое практической деятельности постоянно вынужден принимать решения, причем эти решения и соответствующие им действия должны быть правильными, т.е. приводить к желаемому результату.

В основе решений, принимаемых человеком, всегда лежат его опыт и знание. Во второй половине XX века были предприняты небезуспешные попытки создания систем искусственного интеллекта. Такие системы должны не только уметь обрабатывать комбинации количественной и качественной информации, но и каким-либо образом формализовать знания человека об окружающем мире либо свойственные человеку приемы мышления, посредством которых он изучает окружающий мир и подчиняет его своим интересам.

Интеллектуализация систем поддержки принятия решений началась с математической логики и ее приложений, которые позволили анализировать достаточно сложные ситуации при помощи простейших характеристик – «да» и «нет». Позднее появились нечеткие множества и нечеткая логика, позволившие существенно расширить семантику (смысл) качественных характеристик. Были созданы различные системы моделирования знаний, на их основе разработаны базы знаний и механизмы принятия решений, например, в виде правил «если ситуация такая-то действия должны быть такими». Это, в конечном счете, привело к созданию экспертных систем, имитирующих действия и решения наиболее квалифицированных специалистов – экспертов.

В основе экспертных систем лежат знания, получаемые от экспертов, отсюда их синоним – когнитивные системы (лат. cognition – познание) или системы, основанные на знаниях.

Существует другой подход к созданию систем искусственного интеллекта, называемый структурным, к которому относятся искусственные нейронные сети. Таким сети имитируют деятельность человеческого мозга, образуя совокупность заимствованных из нейрофизиологии моделей параллельных вычислительных структур.

Третье направление реализуется в рамках эволюционных методов, к которым можно отнести генетические алгоритмы. В основу эволюционных методов легло заложенное самой природой свойство живых организмов осуществлять правильный выбор, в частности, на основе эволюционного отбора. Генетические алгоритмы являются мощным универсальным средством решения задач глобальной оптимизации, с помощью которого выбираются решения, если не оптимальные, то достаточно близкие к ним. Отдельное направление составляют системы, которые для решения задач используют сочетания различных методов искусственного интеллекта: экспертных систем, искусственных нейронных сетей, генетических алгоритмов и нечеткой математики.

Данные представляют собой элементарные описания предметов, событий, действий, которые классифицированы и сохранены, но не организованы для передачи какого-либо специального смысла. Данные – это конкретные факты, такие как температура воздуха, высота здания, фамилия сотрудника, адрес сайта и т.д. Элементы данных могут быть числовыми, алфавитно-числовыми, цифровыми, звуковыми или образными. Информация – это данные, которые организованы так, что они имеют значение и ценность для получателя. Любая компьютерная информационная система, реализующая информационный процесс, выполняет следующие функции: воспринимает вводимые пользователем информационные запросы (цели решения задачи) и необходимые исходные данные, обрабатывает введенные и хранимые в системе данные в соответствии с известным алгоритмом и формирует требуемую выходную информацию. С точки зрения реализации перечисленных функций информационную систему можно рассматривать как фабрику, производящую информацию, в которой заказом является информационный запрос, сырьем – исходные данные, продуктом – требуемая информация, а инструментом (оборудованием) – знание, с помощью которого данные преобразуются в информацию. Знания состоят из данных или информации, которые организованы и обработаны с целью передачи понимания, накопленного профессионального опыта, практической деятельности, результатов обучения и экспертизы таким образом, что они могут использоваться для решения текущих проблем или выполнения действий. По степени обобщенности описания знания можно разделить на: Поверхностные – знания о видимых взаимосвязях между отдельными событиями и фактами в предметной области. Глубинные – абстракции, аналогии, схемы, отображающие структуру и природу процессов, протекающих в предметной области. Эти знания объясняют явления и могут использоваться для прогнозирования поведения объектов. Пример. Поверхностные знания: если ввести правильный пароль – на экране компьютера появится изображение рабочего стола. Глубинные знания: понимание принципов работы операционной системы и знания на уровне квалифицированного системного администратора. Современные системы

работают, в основном, с поверхностными знаниями. Это связано с тем, что на данный момент нет универсальных методик, позволяющих выявить глубинные структуры знаний и работать с ними. По степени отражения явлений знания делятся на: Жесткие – позволяют получить четкие однозначные рекомендации при задании начальных условий. Мягкие – допускают множественные расплывчатые решения и многовариантные рекомендации.

1.2 Представление знаний

Представление знаний является одной из наиболее важных проблем при создании системы искусственного интеллекта (СИИ). Форма представления знаний оказывает существенное влияние на характеристики и свойства системы.

Представление знаний – это соглашение о том, как описывать реальный мир. Основная цель представления знаний - строить математические модели реального мира и его частей, для которых соответствие между системой понятий проблемного знания может быть установлено на основе совпадения имен переменных модели и имен понятий без предварительных пояснений и установления дополнительных неформальных соответствий.

В области экспертных систем представление знаний интересует нас в основном как средство отыскания методов формального описания больших массивов полезной информации с целью их последующей обработки с помощью символических вычислений. Формальное описание означает упорядочение в рамках какого-либо языка, обладающего достаточно четко формализованным построения выражений и такого же уровня семантикой, синтаксисом увязывающей смысл выражения с его формой.

Символические вычисления означают выполнение нечисловых операций, в которых могут быть сконструированы символы и символьные структуры для представления различных концептов и отношений между ними.

В области искусственного интеллекта ведется интенсивная работа по созданию языков представления (representation languages).

Под этим термином понимаются компьютерные языки, ориентированные на организацию описаний объектов и идей, в противовес статическим В области экспертных систем представление знаний интересует нас в основном как средство отыскания методов формального описания больших массивов полезной информации с целью их последующей обработки с помощью символических вычислений. Формальное описание означает упорядочение в рамках какого-либо языка, обладающего достаточно четко формализованным построения выражений и такого же уровня семантикой, синтаксисом увязывающей смысл выражения с его формой.

На раннем этапе становления экспертных систем проектирование каждой очередной системы начиналось практически с нуля, в том смысле, что проектировщики для представления знаний и управления их применением использовали самые примитивные структуры данных и средства управления, которые содержались в обычных языках программирования. В редких случаях в существующие языки программирования включались специальные языки представлений правил или фреймов. Такие специальные языки, как правило, обладали двумя видами специфических средств: модулями представления знаний (в виде правил или фреймов);интерпретатором, который управлял активизацией этих модулей.

Совокупность модулей образует базу знаний экспертной системы, а интерпретатор является базовым элементом машины логического вывода. Невольно напрашивается мысль, что эти компоненты могут быть повторно используемыми, т.е. служить основой для создания экспертных систем в разных предметных областях. Использование этих программ в качестве базовых компонентов множества конкретных экспертных систем позволило называть их оболочкой системы. Примером такой оболочки может служить система EMYCIN, которая является предметно-независимой версией системы MYCIN, т.е. это система MYCIN, но без специфической медицинской базы знаний системы как "Empty MYCIN" По мнению разработчиков, EMYCIN вполне может служить "скелетом" для создания консультационных программ во многих предметных

областях, поскольку располагает множеством инструментальных программных средств, облегчающих задачу проектировщика конкретной экспертной консультационной системы. Она особенно удобна для решения дедуктивных задач, таких как диагностика заболеваний или неисправностей, для которых характерно большое количество ненадежных входных измерений

(симптомов, результатов лабораторных тестов и т.п.), а пространство решений, содержащее возможные диагнозы, может быть достаточно четко очерчено.

Некоторые программные средства, впервые разработанные для EMYCIN, в дальнейшем стали типовыми для большинства оболочек экспертных систем. Среди таких средств следует отметить следующие.Язык представления правил. В системе EMYCIN такой язык использует систему обозначений, аналогичную языку ALGOL. Этот язык, с одной стороны, более понятен, чем LISP, а с другой—более строг и структурирован, чем тот диалект обычного английского, который использовался в MYCIN. Индексированная схема применения правил, которая позволяет сгруппировать правила, используя в качестве критерия группировки параметры, на которые ссылаются эти правила.

Интерфейс между консультационной программой, созданной на основе EMYCIN, и конечным пользователем. Этот компонент оболочки обрабатывает все сообщения, которыми обмениваются пользователь и программа (например, запросы программы на получение данных, варианты решения, которые формирует программа в ответ на запросы пользователя, ит.п.).

Интерфейс между разработчиком и программой, обеспечивающий ввод и редактирование правил, редактирование знаний, представленных в форме таблиц, тестирование правил и выполнение репрезентативных задач.

Значительная часть интерфейса реализуется отдельным компонентом EMYCIN Эта программа представляет собой "редактор знаний", который

упрощает редактирование и сопровождение больших баз знаний. Редактор проверяет синтаксическую корректность правил, анализирует взаимную непротиворечивость правил в базе знаний и следит за тем, чтобы новое правило не являлось частным случаем существующих. Противоречие возникает, когда два правила с одинаковыми антецедентами имеют противоречивые консеквенты. Одно правило является частью другого в том случае,

когда совокупность условий антецедента одного правила представляет собой подмножество совокупности условий другого правила, а их консеквенты одинаковы. Но в состав TEIRESIASне включены знания о какой-либо конкретной предметной области или о стратегии решения проблем, которая может быть использована в проектируемой экспертной системе.

Такая организация программы TEIRESIAS является, с одной стороны, ее достоинством, а с другой — недостатком. Общность интерфейса, его независимость от назначения проектируемой экспертной системы — достоинства TEIRESIAS. Используемые в ней методы синтаксического анализа могут быть применены к правилам, относящимся к любой предметной области. А тот факт, что эта программа привносит существенные сложности в процесс общения

инженера по знаниям с экспертом, является ее недостатком. Зачастую знания, которыми располагает эксперт, не укладываются в жесткие рамки правил, на соблюдении которых "настаивает" TEIRESIAS. Тем не менее эта программа включает множество новшеств, которые имеет смысл рассмотреть подробнее.

Контрольные вопросы:

1. Дайте определение понятиям: данные, информация, знания.
2. Классификация знаний по степени обобщенности.
3 Классификация знаний по степени отражения явлений.

Тема 2. Распределенные системы управления

Цель: Изучить основные понятия теории систем.

План:
1. Определение распределенной системы.
2. Концепции аппаратных решений.
3. Концепции программных решений

2.1. Основные понятия

Распределенная система – это набор независимых компьютеров, представляющийся их пользователям единой объединенной системой.

В этом определении оговариваются два момента. Первый относится к аппаратуре: все машины автономны. Второй касается программного обеспечения: пользователи думают, что имеют дело с единой системой. Важны оба момента. Позже в этой главе мы к ним вернемся, но сначала рассмотрим некоторые базовые вопросы, касающиеся как аппаратного, так и программного обеспечения.

Возможно, вместо того чтобы рассматривать определения, разумнее будет сосредоточиться на важных характеристиках распределенных систем. Первая из таких характеристик состоит в том, что от пользователей скрыты различия между компьютерами и способы связи между ними. То же самое относится и к внешней организации распределенных систем. Другой важной характеристикой распределенных систем является способ, при помощи которого пользователи и приложения разнообразно работают в распределенных системах, независимо от того, где и когда происходит их взаимодействие.

Распределенные системы должны также относительно легко поддаваться расширению, или масштабированию. Эта характеристика является прямым следствием наличия независимых компьютеров, но в то же время не указывает, каким образом эти компьютеры на самом деле объединяются в единую систему. Распределенные системы обычно существуют постоянно, однако некоторые их части могут временно выходить из строя. Пользователи и приложения не должны уведомляться о том, что эти части заменены или починены или что добавлены новые части для поддержки дополнительных пользователей или приложений.

Для того чтобы поддержать представление различных компьютеров и сетей в виде единой системы, организация распределенных систем часто включает в себя дополнительный уровень программного обеспечения, находящийся между верхним уровнем, на котором находятся пользователи и приложения, и нижним уровнем, состоящим из операционных систем, как показано на рис. 1.

Соответственно, такая распределенная система обычно называется *системой промежуточного уровня (middleware).*

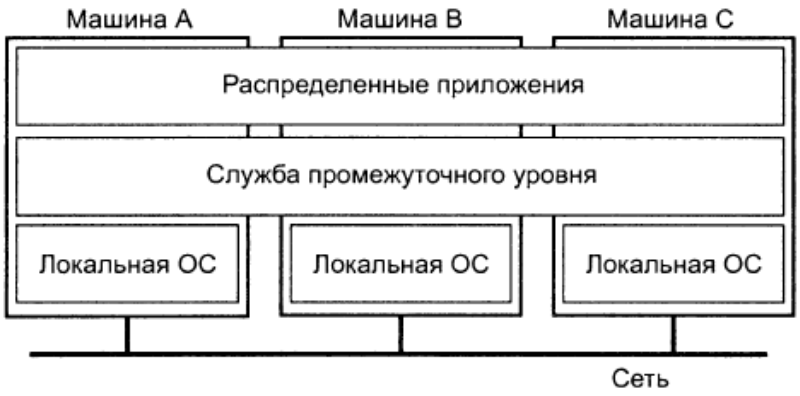

Рисунок 1. Распределенная система организована в виде службы промежуточного уровня

Обсуждение некоторых проблем масштабирования приводит нас к вопросу о том, а как же обычно решаются эти проблемы. Поскольку проблемы масштабируемости в распределенных системах, такие как проблемы производительности, вызываются ограниченной мощностью серверов и сетей, существуют три основные технологии масштабирования: сокрытие времени ожидания связи, распределение и репликация.

Сокрытие времени ожидания связи применяется в случае географического масштабирования. Основная идея проста: постараться по возможности избежать ожидания ответа на запрос от удаленного сервера. Например, если была запрошена служба удаленной машины, альтернативой ожиданию ответа от сервера будет осуществление на запрашивающей стороне других возможных действий.

В сущности, это означает разработку запрашивающего приложения в расчете на использование исключительно *асинхронной связи (asinch?vnous communication).*

Когда будет получен ответ, приложение прервет свою работу и вызовет специальный обработчик для завершения отправленного ранее запроса. Асинхронная связь часто используется в системах пакетной обработки и параллельных приложениях, в которых во время ожидания одной задачей завершения связи предполагается выполнение других более или менее независимых задач. Для осуществления запроса может быть запущен новый управляющий поток выполнения. Хотя он будет блокирован на время ожидания ответа, другие потоки процесса продолжат свое выполнение.

2. Концепции аппаратных решений.

Несмотря на то что все распределенные системы содержат по нескольку процессоров, существуют различные способы их организации в

систему. В особенности это относится к вариантам их соединения и организации взаимного обмена.

За прошедшие годы было предложено множество различных схем классификации компьютерных систем с несколькими процессорами, но ни одна из них не стала действительно популярной и широко распространенной.

Системы, в которых компьютеры используют память совместно, обычно называются *мультипроцессорами {multiprocessors),* а работающие каждый со своей памятью — *мультикомпьютерами (muldcomputers).* Основная разница между ними состоит в том, что мультипроцессоры имеют единое адресное пространство, совместно используемое всеми процессорами. Если один из процессоров записывает, например, значение 44 по адресу 1000, любой другой процессор, который после этого прочтет значение, лежащее по адресу 1000, получит 44. Все машины задействуют одну и ту же память.

В отличие от таких машин в мультикомпьютерах каждая машина использует свою собственную память. После того как один процессор запишет значение 44 по адресу 1000, другой процессор, прочитав значение, лежащее по адресу 1000, получит то значение, которое хранилось там раньше. Запись по этому адресу значения 44 другим процессором никак не скажется на содержимом его памяти.

Типичный пример мультикомпьютера — несколько персональных компьютеров, объединенных в сеть. Каждая из этих категорий может быть подразделена на дополнительные категории на основе архитектуры соединяющей их сети. Две архитектуры обозначены как *шинная (bus)* и *коммутируемая (switched).* Под шиной понимается одиночная сеть, плата, шина, кабель или другая среда, соединяющаявсе машины между собой. Подобную схему использует кабельное телевидение: кабельная компания протягивает вдоль улицы кабель, а всем подпрючикам делаются отводки от основного кабеля к их телевизорам.

Коммутируемые системы, в отличие от шинных, не имеют единой магистрали, такой как у кабельного телевидения. Вместо нее от машины к машине тянутся отдельные каналы, выполненные с применением различных технологий связи. Сообщения передаются по каналам с принятием явного решения о коммутации с конкретным выходным каналом для каждого из них. Так организована глобальная телефонная сеть.

Проведем также разделение распределенных компьютерных систем на *гомогенные (homogeneous)* и *гетерогенные (heterogeneous).* Это разделение применяется исключительно к мультикомпьютерным системам. Для гомогенных мультикомпьютерных систем характерна одна соединяющая компьютеры сеть, использующая единую технологию. Одинаковы также и все процессоры, которые в основном имеют доступ к одинаковым объемам собственной памяти. Гомогенные мультикомпьютерные системы нередко используются в качестве параллельных (работающих с одной задачей), в точности как мультипроцессорные. В отличие от них гетерогенные

мультикомпьютерные системы могут содержать целую гамму независимых компьютеров, соединенных разнообразными сетями. Так, например, распределенная компьютерная система может быть построена из нескольких локальных компьютерных сетей, соединенных коммутируемой магистралью FDDI или ATM.

3. Концепции программных решений.

Аппаратура важна для распределенных систем, однако от программного обеспечения значительно сильнее зависит, как такая система будет выглядеть на самом деле. Распределенные системы очень похожи на традиционные операционные системы. Прежде всего, они работают *KCIK менеджеры ресурсов (resource managers)* существующего аппаратного обеспечения, которые помогают множеству пользователей и приложений совместно использовать такие ресурсы, как процессоры, память, периферийные устройства, сеть и данные всех видов. Во-вторых, что, вероятно, более важно, распределенная система скрывает сложность и гетерогенную природу аппаратного обеспечения, на базе которого она построена, предоставляя виртуальную машину для выполнения приложений.

Чтобы понять природу распределенной системы, рассмотрим сначала операционные системы с точки зрения распределенности. Операционные системы для распределенных компьютеров можно вчерне разделить на две категории — сильно связанные и слабо связанные системы. В сильно связанных системах операционная система в основном старается работать с одним, глобальным представлением ресурсов, которыми она управляет. Слабо связанные системы могут представляться несведущему человеку набором операционных систем, каждая из которых работает на собственном компьютере. Однако эти операционные системы функционируют совместно, делая собственные службы доступными другим.

Это деление на сильно и слабо связанные системы связано с классификацией аппаратного обеспечения, приведенной в предыдущем разделе. Сильно связанные операционные системы обычно называются *распределенными операционными системами (Distributed Operating System, DOS)* и используются для управления мультипроцессорными и гомогенными мультикомпьютерными системами.

Как и у традиционных однопроцессорных операционных систем, основная цель распределенной операционной системы состоит в сокрытии тонкостей управления аппаратным обеспечением, которое одновременно используется множеством процессов. Слабо связанные *сетевые операционные системы (Network Operating Systems, NOS)* используются для управления гетерогенными мультикомпьютерными системами.

Хотя управление аппаратным обеспечением и является основной задачей сетевых операционных систем, они отличаются от традиционных.

Это отличие вытекает из того факта, что локальные службы должны быть доступными для удаленных клиентов. В следующих пунктах мы рассмотрим в первом приближении те и другие. Чтобы действительно составить распределенную сргстему, служб сетевой операционной системы недостаточно. Необходимо добавить к ним дополнительные компоненты, чтобы организовать лучшую поддержку прозрачности распределения.

Этими дополнительными компонентами будут средства, известные как *системы промежуточного уровня (middleware),* которые и лежат в основе современных распределенных систем.

Функциональность распределенных операционных систем в основном не отличается от функциональности традиционных операционных систем, предназначенных для компьютеров с одним процессором за исключением того, что она поддерживает функционирование нескольких процессоров.

Контрольные вопросы:

1. Дайте понятие распределенной системы.
2.Сформулируйте основные концепции аппаратных решений.
3. Сформулируйте основные концепции программных решений

Тема 3. Интеллектуальные системы управления

Цель: Изучить основные понятия интеллектуальных систем управления.

План:

1. Принципы интеллектуального управления

1 Принципы интеллектуального управления

На рубеже XXI века тематика интеллектуальных систем и интеллектуального управления претерпевает значительные изменения. Наметились тенденции перехода от игрушечно-модельного подхода к интеллекту к его восприятию как некоторого характеристического свойства систем высокой организационной сложности, свойству достаточно специфическому и выразимому в достаточной степени только на языках контекстно-зависимого уровня. Столь же полезной тенденцией можно считать и постепенное осознание исследователями, что компьютер фон Неймановской архитектуры, конечный автомат по своей сущности, не может быть эффективным инструментом создания интеллектуальных систем, интеллектуального управления ибо является системой контекстно-независимого уровня.

Под "интеллектуальными системами управления" (ИСУ), в общем случае, понимается предельный по сложности класс систем автоматизированного управления (САУ), ориентированных на приобретение, обработку и использование некоторой дополнительной информации, понимаемой как "знание". Такие системы предназначены для работы в условиях неопределенности (невозможности точного математического описания) информации о свойствах и характеристиках системно-сложных объектов и среды их функционирования.

В условиях работы реальных систем с высоким уровнем неопределенности информации для построения систем управления (СУ) неизбежно применение новых информационных технологий, ориентированных на потоки контекстно-зависимой информации, то есть фактическая разработка новых принципов построения интеллектуального управления - теории ИСУ для систем высших уровней системной сложности.

Теория ИСУ опирается на системный подход в том смысле, что она, ориентируясь на системную, а не на описательную сложность, оставляет систему во внешнем мире и признает существование внутренней целевой установки хотя бы на уровне поддержания стабильности своего существования. Внутреннее и внешнее управление интеллектуальны во взаимодействии, в акте взаимной контекстной ситуационной оценки информации.

Управление, как САУ, дополнено экспертными и эвристическими подходами. Управление же, как руководство, считается деятельностью, достаточно обеспеченной математикой и требующей только вычислительных мощностей для решения систем из многих дифференциальных уравнений, или даже просто решения задач линейного или нелинейного программирования.

Автоматизация, как включение человека в процесс принятия решений, устраняла все проблемы в корне: нет функционала - есть "экспертно" полученное решение. Теоретическая несостоятельность и практическая не успешность такого подхода давно стали очевидными.

Рассмотрим некоторые возможные перспективы тематики интеллектуальных систем измерений и интеллектуального управления. Управление и информация требуют использования баз знаний. Интеллектуальное управление возникает там, где информация трактуется как количественно неопределяемая совокупность данных (фактов, знаков, утверждений и тому подобного) и отношений между ними в семантически ясномконтексте их текущей трактовки. Для восприятия управления как осмысленного потока информации необходимо использование базы данных, если контекст и отношения сообщений постоянны и могут быть заданы конечным набором записей и базы знания, если семантика информации достаточно сложна, контекст переменен, цель управления корректируется в процессе управления, что, как минимум, требует реструктуризации внутренних связей базы данных при акте обработки информационного потока. Указанное требование реструктуризации, обеспечивающее практическую возможность активного (актуализированного) отношения к информации, является отличительным моментом возникновения интеллектуального управления.

Ниже даются некоторые положения теории ИСУ.

Интеллект является атрибутом сложной системы и характеристикой ее отношения к внешнему миру.

Интеллект, как атрибут сложной системы, определяется формированием "образа" (изменением структуры внутренних связей в базе знания), влияющего на реакцию на внешние воздействия. Он проявляется только в актах общения со столь же сложными объектами и активизируется в системе в процессе реорганизации внутренних информационных связей.

Основной цикл управления интеллектуальной системы основан на работе со знанием. Классический основной цикл управления не может быть распространен на системы, требующие интеллектуального управления, потому, что управление через "образ" во-первых требует существенного учета конкретного накопленного знания, формально распределенного между руководителем и системой, а во-вторых, более критично к изменению информации в процессе принятия решения. Соответственно, ИСУ,

опирающаяся на принятие решений с использованием знания, имеет совершенно отличный основной цикл управления.

Следующие два пункта рассматривают теоретическую возможность реализации интеллектуальной системы с использованием конечного автомата.

Интеллектуальные свойства системы "объект - управление" имеют дискретное проявление.

Теория ИСУ утверждает, что при ориентации на определение интеллектуальности, данное через базу знания, система управления может обладать интеллектуальными свойствами лишь на некоторых отрезках времени, в течение которых происходят модификации базы знания, что эквивалентно восприятию системой нового контекста. В каждом акте управления при фиксированной текущей структуре базы знания "интеллектуальные" свойства системы не являются строго необходимыми (база знания структурно фиксирована и не отличается в текущий момент времени от базы данных).

Аппарат реструктуризации баз знания базируется на механизмах, аналогичных "функции расстановки". Теория ИСУ базируется на механизмах реструктуризации данных, по реализации аналогичных конечной аппроксимации функции расстановки, которая по существу не является рекурсивной, а может быть и рекурсивно-перечислимой (т.е. не является множеством значений некоторой рекурсивной функции). Исходя из сказанного, для правомерного использования конечного автомата (компьютера) в составе интеллектуальной системы, теория должна рассматривать возможность построения абстрактных конструкций, реализующих невычислимые в обычном смысле объекты.

В настоящее время теория робастного управления (H^∞-теория управления, H^∞-управление) является одной из интенсивно развивающихся ветвей теории управления. Сравнительно молодая (первые работы появились в начале 80-х гг.), она возникла из насущных практических проблем синтеза многомерных линейных систем управления, функционирующих в условиях различного рода возмущений и изменения параметров. Можно подойти к проблеме проектирования управления реальным сложным объектом, функционирующим в условиях неопределенности, другим образом: не пытаться использовать один тип управления — адаптивный или робастный. Очевидно, следует выбирать тот тип, который соответствует состоянию окружающей среды и системы, определенному по имеющейся в распоряжении системы информации. Если же в процессе функционирования системы можно организовать получение информации, целесообразно использовать ее в процессе управления.Но реализация такого комбинированного управления до недавнего времени наталкивалась на непреодолимые трудности при определении алгоритма выбора типа управления. Достигнутые в разработке проблем искусственного интеллекта

успехи делают возможным синтез такого алгоритма. Действительно, поставим задачу: спроектировать систему, использующую адаптивное и робастное управление и осуществляющую выбор типа управления на основе методов искусственного интеллекта. Для этого рассмотрим особенности обоих типов и, учитывая их специфические качества, определим, как можно построить систему комбинированного управления. Одним из основных понятий в теории робастного управления является понятие неопределенности. Неопределенность объекта отражает неточность модели объекта, причем как параметрическую, так и структурную. Рассмотрим подробнее формы задания неопределенности в робастной теории управления с помощью простой системы — с одним входом и одним выходом Сигналы имеют следующую интерпретацию: r — задающий входной сигнал; u — входной сигнал (вход) объекта; d — внешнее возмущение; y — выходной сигнал (выход) объекта, измеряемый.

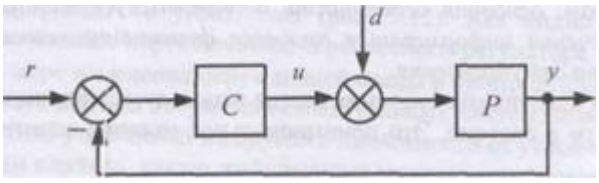

Рисунок 2– Система с одним входом и одним выходом

В H^{∞}-теории управления неопределенность удобно задавать в частотной области. Предположим, что передаточная функциянормального объекта P, и рассмотрим возмущенный объект, передаточная функция которого,

$\tilde{P} = (1 +$

, **1**

где W – фиксированная передаточная функция (весовая функция);

Δ–произвольная устойчивая передаточная функция, удовлетворяющая неравенству $\| \Delta \|_{\infty} \leq 1$.

Такое возмущение Δбудем называть допустимым. Ниже приведем некоторые варианты моделей неопределенности:

(1+ΔW)P; P+ΔW; P/(1+ΔWP);
P/(1+ΔW). **2**

Соответствующие предположения должны быть сделаны для величин Δи W в каждом случае. Неопределенность входных сигналов d отражает различную природу внешних возмущений, действующих на объект и регулятор. Неопределенный объект, таким образом, может рассматриваться как некое множество объектов. Выберем некую характеристику систем с обратной связью, например устойчивость. Регулятор C является робастным

относительно этой характеристики, если ею обладает любой из множества объектов, задаваемых неопределенностью. Таким образом, понятие робастности подразумевает наличие регулятора, множества объектов и фиксацию определенной характеристики системы. В этой работе мы не будем затрагивать всего множества задач, решаемых в рамках H^{∞} теории управления. Коснемся лишь задачи минимальной чувствительности: построения такого регулятора С, который стабилизирует замкнутую систему и минимизирует влияние внешних возмущений на выход у, иначе говоря, минимизирует H^{∞} норму матрицы передаточных функций от внешних возмущений к выходу у. Одной из особенностей решения этой, да и всего множества задач робастного управления является тот факт, что мы заранее в процессе проектирования регулятора закладываем ограничения на входные воздействия и неопределенность объекта в виде неравенств $\|\Delta\|_{\infty} \leq 1, \|d\|_{2} \leq 1 | \wedge \| 2 < c_1$.В процессе функционирования робастной системы информация о неопределенностях в системе не используется для управления. Естественно, это приводит к тому, что робастные системы консервативны и качество переходных процессов порой не удовлетворяет разработчиков этих систем. Подобно робастной адаптивная система управления строится для объектов, информация о которых или о воздействиях на которые недоступна в начале функционирования системы. Чаще всего свойство адаптации достигается посредством формирования в явном или неявном виде математической модели объекта или входного воздействия. Этим отличается как поисковое адаптивное управление, в основе которого поиск и удержание экстремума показателя качества управления, так и бес поисковое, в основе которого компенсация отклонения фактических изменений управляемых координат от желаемых изменений, соответствующих требуемому уровню показателя качества. Далее по уточненной модели происходит подстройка адаптивного регулятора. Таким образом, основная особенность адаптивных систем управления – возможность получения информации в процессе функционирования и использования этой информации для управления. Более того, в адаптивных системах всегда используется априорная информация о неопределенности в системе. Это принципиальное отличие адаптивного подхода от робастного.

Рассмотрим простейшую адаптивную систему управления, обеспечивающую отслеживания входного сигнала в присутствии помехи на входе объекта (рисунок3).

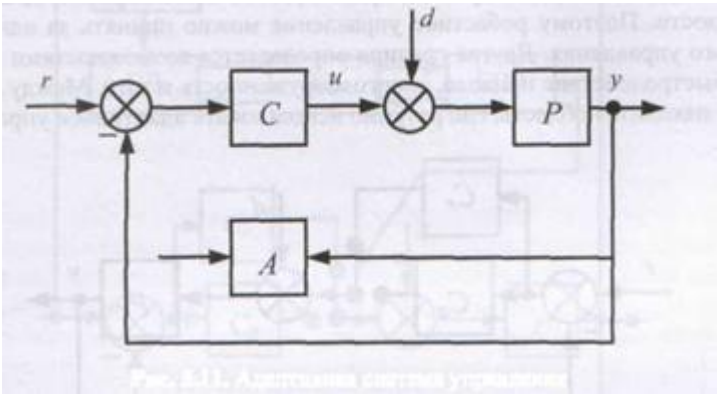

Рисунок. 3–Адаптивная система управления

Формальное отличие от схемы на рисункеЗ— блок адаптации А, который на основании выходного сигнала объекта и сигнала, характеризующего заданное качество, вырабатывает сигнал подстройки коэффициентов адаптивного регулятора. Имея в виду недостатки каждого из регуляторов, целесообразно попытаться использовать их достоинства, предложив комбинированную схему управления объектом. Адаптивная система при помощи блока адаптации вырабатывает некоторую информацию о состоянии внешней среды. В частности, в рассматриваемом случае можно получить информацию о внешнем возмущении **d.** Алгоритм управления $\mathbf{C_a}$ соответствует текущему состоянию внешней среды, согласно заложенному в блоке адаптации критерию. Но адаптивная система требует, чтобы входной сигнал **ru** достаточно широкий частотный диапазон, и накладывает жесткие ограничения на значение и частотный спектр сигнала внешнего возмущения d. Поэтому адаптивные системы могут работать только в узких диапазонах входного сигнала **ru** внешнего возмущения **d.** Вне этих диапазонов адаптивная система имеет низкое качество управления и может даже потерять устойчивость. Рассмотренные выше свойства робастного и адаптивного управления приводят к заключению, что в процессе функционирования системы в одних случаях выгодно использовать робастное управление, в других — адаптивное, т.е. иметь возможность комбинировать управление в зависимости от состояния внешней среды. Комбинированное управление. Основной вопрос при проектировании систем комбинированного управления заключается в том, каким образом, на основании каких знаний (информации) осуществлять выбор того или иного типа управления. Наиболее широкие возможности для этого представляют методы искусственного интеллекта. Их преимущество по сравнению с простыми переключающими алгоритмами состоит в использовании широкого спектра данных и знаний для формирования алгоритма выбора типа управления. Если формально объединить схемы, приведенные на рисунках(2,3) то получим схему комбинированного управления (рисунок4).

Как видно из рисунка, сигнал управления и должен переключаться с робастного регулятора на адаптивный и наоборот — по мере изменения окружающей среды в процессе функционирования системы. Используя методы теории интеллектуальных систем, можно обеспечить переход с одного типа управления на другой в зависимости от условий работы системы.

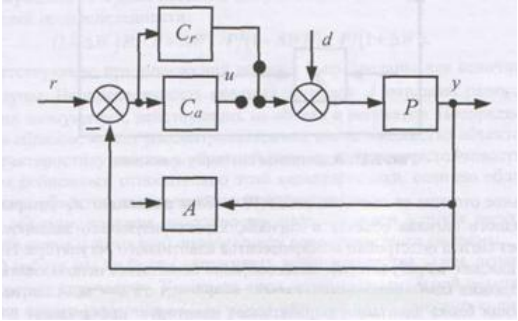

Рисунок 4 – Схема комбинированного управления

Рассмотрим сначала, какую информацию можно использовать для работы интеллектуального блока системы. Как известно, системы с одним входом и одним выходом хорошо описываются в частотной области. Поэтому естественно использовать частотные характеристики для организации процесса принятия решений при выборе типа управления. Как указывалось выше, частотная характеристика системы с робастным управлением соответствует наихудшему сочетанию параметров в области неопределенности. Поэтому робастное управление можно принять за одну из границ выбираемого управления. Другая граница определяется возможностями исследуемой системы (быстродействие привода, энерговооруженность и т.д.). Между этими двумя границами находится область, где разумно использовать адаптивное управление.

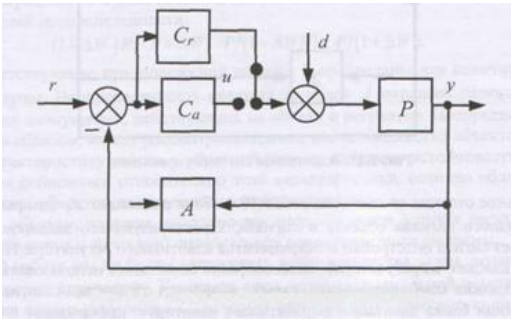

Рисунок 5 – Схема комбинированного управления

Так как адаптивный алгоритм чувствителен к начальному этапу функционирования системы, то на этом этапе целесообразно использовать робастное управление, которое достаточно нечувствительно к скорости изменения внешней помехи. Но его недостатком является большая длительность переходных процессов и большие допустимые значения выходной координаты при действии помехи. По истечении некоторого времени робастное управление имеет смысл переключить на адаптивное. Адаптивное управление позволяет более точно отследить входной сигнал при наличии информации о помехе. Адаптивное управление требовательно к богатству спектра входного сигнала, и, например, при медленно меняющихся сигналах возможны срывы процессов адаптации или сильное их замедление. В такой ситуации необходимо снова переходить на робастное управление, гарантирующее устойчивость работы системы. Из вышеизложенного следует, что для функционирования системы необходимо иметь информацию о частотном спектре полезного сигнала помехи и об отношении сигнал/шум. Кроме того, требуется предварительная информация о частотном спектре, на котором работает адаптивная система, и о частных характеристиках объекта управления на границах области неопределенности. Из этой информации можно сформировать базу данных, в которую информация, индивидуальная для каждого класса объектов, заносится заранее. Информация о частотном спектре полезного сигнала, помех и об отношении сигнал/шум поступает в базу данных по мере функционирования системы и постоянно обновляется. Содержимое базы данных может быть использовано в базе знаний, которая формируется в виде правил. В зависимости от конкретных свойств системы можно установить переключения двух типов управления. Требуемые правила формируются в одной из логических систем, подходящей для рассматриваемого случая. Имея базы данных и знаний, можно разработать механизм принятия решений, который будет обеспечивать правильный выбор типа управления в зависимости от условий функционирования системы.

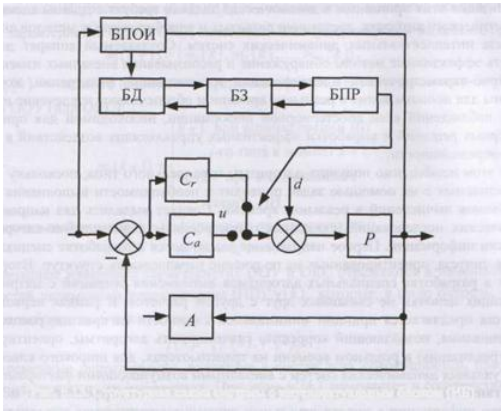

Рисунок 6 – Структурная схема системы с интеллектуальным блоком (ИБ)

Интеллектуальная часть системы работает дискретно, на заданных интервалах времени. На рисунке 13.8 приведена структурная схема системы с интеллектуальным блоком ИБ, обеспечивающим выбор типа управления. На вход блока поступают сигнал ri измеряемый, выходной сигнал объекта y. В блоке предварительной обработки информации БПОИ по временным характеристикам сигналов r(t), y(t) определяются частотные характеристики входного сигнала r(w) и внешнего возмущения d(w), взаимное расположение спектров r(w) и d(w) и характерные значения отношения сигнал/шумr(w)/d(w). Вся эта информация поступает в базу данных БД. Блок принятия решения БПР, используя сформированную базу знаний БЗ и данные БД, вырабатывает решение, в соответствии с которым включается один из типов управления. На следующем интервале процесс повторяется с использованием новых данных.

Контрольные вопросы:

1. В чем заключаются принципы интеллектуальных систем?

Тема **4. Интеллектуальный анализ данных**

Цель: Изучить основные понятия интеллектуальных систем управления.

План:

1. Исследование множества объектов.
2. Майнинг больших данных (Big Data).

1. Исследование множества объектов.

Data Mining - мультидисциплинарная область, возникшая и развивающаяся на базе таких наук как прикладная статистика, распознавание образов, искусственный интеллект, теория баз данных и др., см. рис. 1.

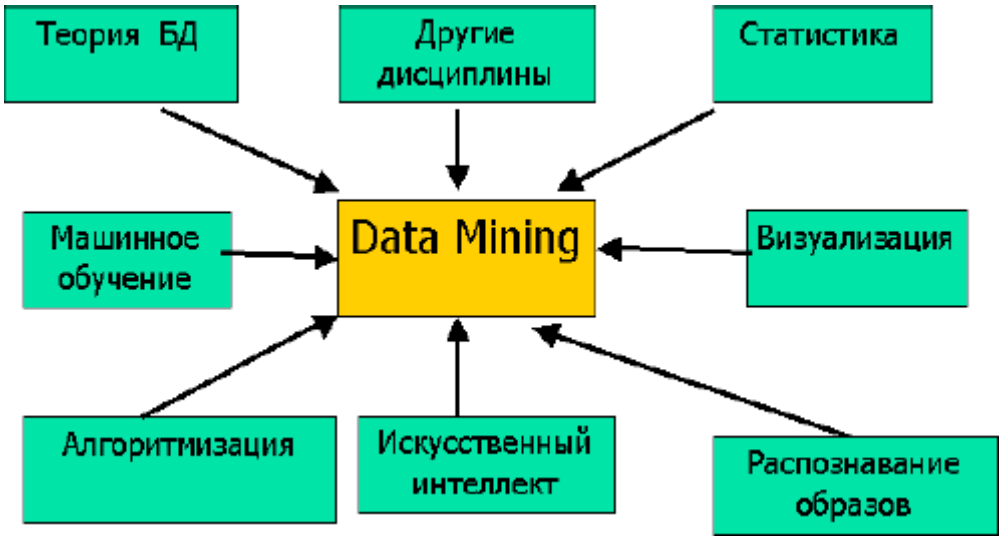

Рисунок – 7. Data Mining как мультидисциплинарная область

Статистика - это наука о методах сбора данных, их обработки и анализа для выявления закономерностей, присущих изучаемому явлению.

Статистика является совокупностью методов планирования эксперимента, сбора данных, их представления и обобщения, а также анализа и получения выводов на основании этих данных.

Статистика оперирует данными, полученными в результате наблюдений либо экспериментов.

Машинное обучение можно охарактеризовать как процесс получения программой новых знаний. Примером алгоритма машинного обучения являются нейронные сети.

Искусственный интеллект - научное направление, в рамках которого ставятся и решаются задачи аппаратного или программного моделирования

видов человеческой деятельности, традиционно считающихся интеллектуальными.

Искусственным интеллектом называют свойство интеллектуальных систем выполнять творческие функции, которые традиционно считаются прерогативой человека.

Data Mining - это процесс поддержки принятия решений, основанный на поиске в данных скрытых закономерностей.

Технология Data Mining - это процесс обнаружения в «сырых» данных ранее неизвестных, нетривиальных, практически полезных и доступных интерпретации знаний, необходимых для принятия решений в различных сферах человеческой деятельности.

Технология Data Mining предназначена для поиска в больших объемах данных неочевидных, объективных и полезных на практике закономерностей. Неочевидных - это значит, что найденные закономерности не обнаруживаются стандартными методами обработки информации или экспертным путем.

Объективных - это значит, что обнаруженные закономерности будут полностью соответствовать действительности, в отличие от экспертного мнения, которое всегда является субъективным.

Практически полезных - это значит, что выводы имеют конкретное значение, которому можно найти практическое применение.

В основу технологии Data Mining положена концепция шаблонов (patterns), которые представляют собой закономерности, свойственные подвыборкам данных, которые могут быть выражены в форме, понятной человеку.

Цель поиска закономерностей - представление данных в виде, отражающем искомые процессы. Построение моделей прогнозирования также является целью поиска закономерностей.

Средства Data Mining, в отличие от статистических, не требуют наличия строго определенного количества ретроспективных данных. Эта особенность может стать причиной обнаружения недостоверных, ложных моделей и, как результат, принятия на их основе неверных решений. Необходимо осуществлять контроль статистической значимости обнаруженных знаний.

В широком понимании данные представляют собой факты, текст, графики, картинки, звуки, аналоговые или цифровые видео-сегменты. Данные могут быть получены в результате измерений, экспериментов, арифметических и логических операций.

Данные должны быть представлены в форме, пригодной для хранения, передачи и обработки. Иными словами, данные - это необработанный материал, предоставляемый поставщиками данных и используемый потребителями для формирования информации на основе данных.

На рисунке 2 представлена двухмерная таблица, представляющая собой набор данных. По горизонтали таблицы располагаются атрибуты объекта или его признаки. По вертикали таблицы - объекты. Объект описывается как набор атрибутов. Объект может быть представлен, как строка таблицы.

Атрибут - свойство, характеризующее объект. Атрибут также называют переменной, измерением, характеристикой. Он может быть представлен, как поле таблицы.

В результате перехода от общих категорий к конкретным величинам (операционализации понятий), получается набор переменных изучаемого понятия.

Переменная (variable) - свойство или характеристика, общая для всех изучаемых объектов, проявление которой может изменяться от объекта к объекту.

	Атрибуты				
	Код клиента	Возраст	Семейное положение	Доход	Класс
Объекты	1	18	Single	125	1
	2	22	Married	100	1
	3	30	Single	70	1
	4	32	Married	120	1
	5	24	Divorced	95	2
	6	25	Married	60	1
	7	32	Divorced	220	1
	8	19	Single	85	2
	9	22	Married	75	1
	10	40	Single	90	2

Рисунок – 8. Двухмерная таблица «объект-атрибут»

Значение (value) переменной является проявлением признака.

При анализе данных, как правило, нет возможности рассмотреть всю совокупность объектов. Изучение очень больших объемов данных является

дорогостоящим процессом, требующим больших временных затрат, а также неизбежно приводит к ошибкам, связанным с человеческим фактором.

Вполне достаточно рассмотреть некоторую часть всей совокупности, то есть выборку, и получить интересующую информацию на ее основании.

Однако размер выборки должен зависеть от разнообразия объектов, представленных в генеральной совокупности. В выборке должны быть представлены различные комбинации и элементы генеральной совокупности.

Генеральная совокупность (population) - вся совокупность изучаемых объектов, интересующая исследователя.

Выборка (sample) - часть генеральной совокупности, определенным способом отобранная с целью исследования и получения выводов о свойствах и характеристиках генеральной совокупности.

Параметры - числовые характеристики генеральной совокупности. Статистики - числовые характеристики выборки.

Часто исследования основываются на гипотезах. Гипотезы проверяются с помощью данных. Гипотеза это предположение относительно параметров совокупности объектов, которое должно быть проверено на ее части.

Гипотеза - частично обоснованная закономерность знаний, служащая либо для связи между различными эмпирическими фактами, либо для объяснения факта или группы фактов.

Допустим, существует гипотеза, что зависимая переменная (продолжительность жизни) изменяется в зависимости от некоторых причин (качество питания, образ жизни, место проживания и т.д.), которые и являются независимыми переменными.

Однако переменная изначально не является зависимой или независимой. Она становится таковой после формулировки конкретной гипотезы. Зависимая переменная в одной гипотезе может быть независимой в другой.

Измерение - процесс присвоения чисел характеристикам изучаемых объектов согласно определенному правилу.

В процессе подготовки данных измеряется не сам объект, а его характеристики. Измерение осуществляется в некоторой шкале.

Шкала - правило, в соответствии с которым объектам присваиваются числа. Переменные могут являться числовыми данными либо символьными.

Числовые данные, в свою очередь, могут быть дискретными и непрерывными.

Дискретные данные являются значениями признака, общее число которых не более чем счетно.

Непрерывные данные - данные, значения которых находятся в некотором интервале.

Обычно рассматривают следующие шкалы измерений: номинальная, порядковая, интервальная, относительная и дихотомическая.

Номинальная шкала (nominal scale) - шкала, содержащая только категории; данные в ней не могут упорядочиваться, с ними не могут быть произведены никакие арифметические действия.

Для этой шкалы применимы только такие операции: равно\не равно.

Порядковая шкала (ordinal scale) - шкала, в которой числа присваивают объектам для обозначения относительной позиции объектов, но не величины различий между ними.

Шкала измерений дает возможность ранжировать значения переменных. Измерения же в порядковой шкале содержат информацию только о порядке следования величин, но не позволяют сказать "насколько одна величина больше другой", или "насколько она меньше другой". Для этой шкалы применимы только такие операции: равно\не равно, больше\меньше.

Интервальная шкала (interval scale) - шкала, разности между значениями которой могут быть вычислены, однако их отношения не имеют смысла.

Эта шкала позволяет находить разницу между двумя величинами, обладает свойствами номинальной и порядковой шкал, а также позволяет определить количественное изменение признака.

Для этой шкалы применимы только такие операции: равно\не равно, больше\меньше, сложения\вычитания.

Относительная шкала (ratio scale) - шкала, в которой есть определенная точка отсчета и возможны отношения между значениями шкалы.

Для этой шкалы применимы операции: равно\не равно, больше\меньше, сложения\вычитания), умножения\деления.

Дихотомическая шкала (dichotomous scale) - шкала, содержащая только две категории, обычно это «да-нет» (частный случай номинальной шкалы).

Пример использования разных шкал для измерений свойств различных объектов, в данном случае температурных условий, приведен в таблице данных, изображенной на рис.3.

Номер объекта	Профессия (номинальная шкала)	Средний балл (интервальная шкала)	Образование (порядковая шкала)
1	слесарь	22	среднее
2	ученый	55	высшее
3	учитель	47	высшее

Рисунок 9 - Множество измерений свойств различных объектов

Типы наборов данных Данные, состоящие из записей

Наиболее часто встречающиеся данные - данные, состоящие из записей (record data). Это табличные данные, матричные данные, документальные данные, транзакционные или операционные.

Табличные данные состоят из записей, каждая из которых состоит из фиксированного набора атрибутов.

Транзакционные данные представляют собой особый тип данных, где каждая запись, являющаяся транзакцией, включает набор значений.

Пример транзакционной базы данных, содержащей перечень покупок клиентов магазина, приведен на рис.4.

Транза́кция — минимальная логически осмысленная операция, которая имеет смысл и может быть совершена только полностью.

TID	Items
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

Рисунок 10 - Пример транзакционных данных

С помощью карт, например, можно отследить изменения объектов во времени и пространстве, определить характер их распределения на плоскости или в пространстве.

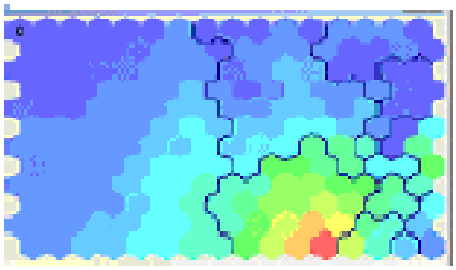

Рисунок 11 - Пример данных типа «Карта Кохонена»

Преимуществом графического представления данных является большая простота их восприятия, чем, например, табличных данных. Пример карты Кохонена (модель нейронных сетей), представлен на рис.6.

Одна из основных особенностей данных современного мира состоит в том, что их становится очень много.

Возможны четыре аспекта работы с данными: определение данных, вычисление (измерение), манипулирование (форматное преобразование) и обработка (сбор, передача и др.).

При манипулировании данными используется структура данных типа "файл". Файлы могут иметь различные форматы.

Наиболее распространенным форматом хранения данных для Data Mining выступают базы данных.

База данных (Database) - это особым образом организованные и хранимые в электронном виде данные.

Данные организованы неким конкретным способом, способным облегчить их поиск и доступ к ним для одного или нескольких приложений. Также такая организация данных предусматривает наличие минимальной избыточности данных.

Базы данных являются одной из разновидностей информационных технологий, а также формой хранения данных.

Целью создания баз данных является построение такой системы данных, которая бы не зависела от программного обеспечения, применяемых технических средств и физического расположения данных в ЭВМ. Построение такой системы данных должно обеспечивать непротиворечивую и целостную информацию. При проектировании базы данных предполагается многоцелевое ее использование. База данных в простейшем случае представляется в виде системы двумерных таблиц.

Схема данных - описание логической структуры данных, специфицированное на языке описания данных и обрабатываемое СУБД (системой управления базами данных).

Схема пользователя - зафиксированный для конкретного пользователя один вариант порядка полей таблицы.

Системы управления базами данных (СУБД).

Система управления базой данных - это программное обеспечение, контролирующее организацию, хранение, целостность, внесение изменений, чтение и безопасность информации в базе данных.

СУБД (Database Management System, DBMS) - представляет собой оболочку, с помощью которой при организации структуры таблиц и заполнения их данными получается та или иная база данных.

Система управления реляционными базами данных (Relational Database Management System) - это СУБД, основанная на реляционной модели данных.

В реляционной модели данных любое представление данных сводится к совокупности реляционных таблиц (двумерных таблиц особого типа). Системы управления реляционными базами данных используются для построения хранилищ данных (предметно-ориентированных

информационных баз данных). СУБД имеет программные, технические и организационные составляющие.

Программные средства включают систему управления, обеспечивающую ввод-вывод, обработку и хранение информации, создание, модификацию и тестирование базы данных.

Внутренними языками программирования СУБД являются языки четвертого поколения (C, C++, Pascal, Object Pascal). С помощью языков БД создаются приложения, базы данных и интерфейс пользователя, включающий экранные формы, меню, отчеты.

Метаданные хранилища обычно размещаются в репозитории (обычно данные в репозитории хранятся в виде файлов, доступных для дальнейшего распространения по сети). Это позволяет использовать метаданные совместно с различными инструментами и процессам при проектировании, установке, эксплуатации и администрировании хранилища.

4.2.Майнинг больших данных (Big Data).

Процесс Data Mining состоит из определенных этапов, включающих элементы сравнения, типизации, классификации, обобщения, абстрагирования, повторения. Процесс Data Mining неразрывно связан с процессом принятия решений. Процесс Data Mining строит модель, а в процессе принятия решений эта модель эксплуатируется.

Традиционный процесс Data Mining включает следующие этапы:

- анализ предметной области;
- постановка задачи;
- подготовка данных;
- построение моделей;
- проверка и оценка моделей;
- выбор модели;
- применение модели;
- коррекция и обновление модели.

Этап 1. Анализ предметной области.

Исследование - это процесс познания определенной предметной области, объекта или явления с определенной целью.

Процесс исследования заключается в наблюдении свойств объектов с целью выявления и оценки важных, с точки зрения субъекта-исследователя, закономерных отношений между показателями данных свойств.

Предметная область - это мысленно ограниченная область реальной действительности, подлежащая описанию или моделированию и исследованию.

Предметная область состоит из объектов, различаемых по свойствам и находящихся в определенных отношениях между собой или взаимодействующих каким-либо образом.

Предметная область - это часть реального мира, она содержит как

существенные, так и не значащие данные, с точки зрения проводимого исследования. Существенность данных зависит от выбора предметной области.

В процессе изучения предметной области должна быть создана ее модель. Знания из различных источников должны быть формализованы при помощи каких-либо средств. Это могут быть текстовые описания предметной области или специализированные графические нотации. Существует большое количество методик описания предметной области: например, методика структурного анализа SADT и основанная на нем IDEF0, диаграммы потоков данных Гейна-Сарсона, методика объектно-ориентированного анализа UML и другие.

Модель предметной области описывает процессы, происходящие в предметной области, и данные, которые в этих процессах используются. От того, насколько верно смоделирована предметная область, зависит успех дальнейшей разработки приложения Data Mining.

Этап 2. Постановка задачи.

Постановка задачи Data Mining включает следующие шаги:

- формулировка задачи;
- формализация задачи.

Постановка задачи включает также описание статического и динамического поведения исследуемых объектов.

Описание статики подразумевает описание объектов и их свойств. При описании динамики описывается поведение объектов и те причины, которые влияют на их поведение. Динамика поведения объектов часто описывается вместе со статикой.

Иногда этапы анализа предметной области и постановки задачи объединяют в один этап.

Этап 3. Подготовка данных.

Цель этапа: разработка базы данных для Data Mining.

Подготовка данных является важнейшим этапом, от качества выполнения которого зависит возможность получения качественных результатов всего процесса Data Mining. Рассмотрим шаги этого этапа.

1. Определение и анализ требований к данным.

На этом шаге осуществляется так называемое моделирование данных, т.е. определение и анализ требований к данным, которые необходимы для осуществления Data Mining. При этом изучаются вопросы распределения данных (географическое, организационное, функциональное); вопросы доступа к данным, которые необходимы для анализа, необходимость во внешних и/или внутренних источниках данных; а также аналитические характеристики системы (измерения данных, основные виды выходных документов, последовательность преобразования информации и др.).

2. Сбор данных.

Наличие в организации хранилища данных делает анализ проще и эффективней, его использование, с точки зрения вложений, обходится дешевле, чем использование отдельных баз данных или витрин данных. Однако далеко не всегда имеются хранилища данных. В этом случае источником для исходных данных являются оперативные, справочные и архивные БД, т.е. данные из существующих информационных систем.

Также для Data Mining может потребоваться информация из информационных систем внешних источников, бумажных носителей, а также знания экспертов или результаты опросов.

На этом шаге осуществляется кодирование некоторых данных.

При определении необходимого количества данных следует учитывать, являются ли данные упорядоченными или нет.

Если данные упорядочены и имеют дело с временными рядами, желательно знать, включает ли такой набор данных сезонную/цикличную компоненту. В случае присутствия в наборе данных сезонной/цикличной компоненты, необходимо иметь данные как минимум за один сезон/цикл.

Если данные не упорядочены, то есть события из набора данных не связаны по времени, в ходе сбора данных следует соблюдать следующие правила.

- Недостаточное количество записей в наборе данных может стать причиной построения некорректной модели. С точки зрения статистики, точность модели увеличивается с увеличением количества исследуемых данных.
- Возможно, некоторые данные являются устаревшими или описывают какую-то нетипичную ситуацию, и их нужно исключить из базы данных.
- Алгоритмы, используемые для построения моделей на сверхбольших базах данных, должны быть масштабируемыми.
- При использовании многих алгоритмов необходимо определенное (желательное) соотношение входных переменных и количества наблюдений. Количество записей (примеров) в наборе данных должно быть значительно больше количества факторов (переменных).
- Набор данных должен быть репрезентативным и представлять как можно больше возможных ситуаций. Пропорции представления различных примеров в наборе данных должны соответствовать реальной ситуации.

3. Предварительная обработка данных.

Анализировать можно как качественные, так и некачественные данные. Результат будет достигнут и в том, и в другом случае. Для обеспечения качественного анализа необходимо проведение предварительной обработки данных, которая является необходимым этапом процесса Data Mining.

Данные, полученные в результате сбора, должны соответствовать определенным критериям качества. Таким образом, можно выделить важный

подэтап процесса Data Mining - оценивание качества данных.

Качество данных (Data quality) - это критерий, определяющий полноту,

точность, своевременность и возможность интерпретации данных.

Данные могут быть высокого качества и низкого качества, последние - это так называемые грязные или "плохие" данные.

Данные высокого качества - это полные, точные, своевременные данные, которые поддаются интерпретации. Такие данные обеспечивают получение качественного результата: знаний, которые смогут поддерживать процесс принятия решений.

Данные низкого качества, или грязные данные - это отсутствующие, неточные или бесполезные данные с точки зрения практического применения (например, представленные в неверном формате, не соответствующем стандарту).

Наиболее распространенные виды грязных данных:

- пропущенные значения;
- дубликаты данных;
- шумы и выбросы.

Пропущенные значения (Missing Values).

Некоторые значения данных могут быть пропущены в связи с тем, что:

- данные вообще не были собраны (например, при анкетировании скрыт возраст);
- некоторые атрибуты могут быть неприменимы для некоторых объектов (например, атрибут "годовой доход" неприменим к ребенку).

Обработка пропущенных данных.

1. Исключить объекты с пропущенными значениями из анализа.
2. Рассчитать новые значения для пропущенных данных.
3. Игнорировать пропущенные значения в процессе анализа.
4. Заменить пропущенные значения на возможные значения.

Дублирование данных (Duplicate Data).

Набор данных может включать продублированные данные, т.е. дубликаты. Дубликатами называются записи с одинаковыми значениями всех атрибутов. Наличие дубликатов в наборе данных может являться способом повышения значимости некоторых записей. Такая необходимость иногда возникает для особого выделения определенных записей из набора данных. Однако в большинстве случаев, продублированные данные являются результатом ошибок при подготовке данных.

Существует два варианта обработки дубликатов. При первом варианте удаляется вся группа записей, содержащая дубликаты. Этот вариант используется в том случае, если наличие дубликатов вызывает недоверие к информации, полностью ее обесценивает. Второй вариант состоит в замене группы дубликатов на одну уникальную запись.

Шумы и выбросы.

Выбросы - резко отличающиеся объекты или наблюдения в наборе данных. Шумы и выбросы являются достаточно общей проблемой в анализе данных. Выбросы могут как представлять собой отдельные наблюдения, так и быть объединенными в некие группы. Задача аналитика - не только их обнаружить, но и оценить степень их влияния на результаты дальнейшего анализа. Если выбросы являются информативной частью анализируемого набора данных, используют робастные методы и процедуры.

Достаточно распространена практика проведения двухэтапного анализа

- с выбросами и с их отсутствием - и сравнение полученных результатов.

Различные методы Data Mining имеют разную чувствительность к выбросам, этот факт необходимо учитывать при выборе метода анализа данных. Также некоторые инструменты Data Mining имеют встроенные процедуры очистки от шумов и выбросов.

Визуализация данных позволяет представить данные, в том числе и выбросы, в графическом виде.

Очевидно, что результаты Data Mining на основе грязных данных не могут считаться надежными и полезными, очевидно необходима очистка данных.

Очистка данных (data cleaning, data cleansing или scrubbing) занимается выявлением и удалением ошибок и несоответствий в данных с целью улучшения качества данных.

Специальные средства очистки обычно имеют дело с конкретными областями - в основном это имена и адреса - или же с исключением дубликатов. Преобразования обеспечиваются либо в форме библиотеки правил, либо пользователем в интерактивном режиме. Преобразования данных могут быть автоматически получены с помощью средств согласования схемы.

Метод очистки данных должен удовлетворять ряду критериев.

1. Метод должен выявлять и удалять все основные ошибки и несоответствия, как в отдельных источниках данных, так и при интеграции нескольких источников.
2. Метод должен поддерживаться определенными инструментами, чтобы сократить объемы ручной проверки и программирования, и быть гибким в плане работы с дополнительными источниками.
3. Очистка данных не должна производиться в отрыве от связанных со схемой преобразования данных, выполняемых на основе сложных метаданных.
4. Функции маппирования для очистки и других преобразований данных должны быть определены декларативным образом и подходить для использования в других источниках данных и в обработке запросов.

5. Инфраструктура технологического процесса должна особенно интенсивно поддерживаться для Хранилищ данных, обеспечивая эффективное и надежное выполнение всех этапов преобразования для множества источников и больших наборов данных.

Этапы очистки данных

В целом, очистка данных включает следующие этапы

Этап № 1. Анализ данных.

Подробный анализ данных необходим для выявления подлежащих удалению видов ошибок и несоответствий. Здесь можно использовать как ручную проверку данных или их шаблонов, так и специальные программы для получения метаданных о свойствах данных и определения проблем качества.

Этап № 2. Определение порядка и правил преобразования данных.

В зависимости от числа источников данных, степени их неоднородности и загрязненности, данные могут требовать достаточно обширного преобразования и очистки. Иногда для отображения источников общей модели данных используется трансляция схемы; для Хранилищ данных обычно используется реляционное представление. Первые шаги по очистке могут уточнить или изменить описание проблем отдельных источников данных, а также подготовить данные для интеграции. Дальнейшие шаги должны быть направлены на интеграцию схемы/данных и устранение проблем множественных элементов, например, дубликатов. Для Хранилищ в процессе работы по определению ETL (*Extract, Transform, Load*) должны быть определены методы контроля и поток данных, подлежащий преобразованию и очистке.

Преобразования данных, связанные со схемой, так же как и этапы очистки, должны, насколько возможно, определяться с помощью декларативного запроса и языка маппирования, обеспечивая, таким образом, автоматическую генерацию кода преобразования. К тому же, в процессе преобразования должна существовать возможность запуска написанного пользователем кода очистки и специальных средств.

Этапы преобразования могут требовать обратной связи с пользователем по тем элементам данных, для которых отсутствует встроенная логика очистки.

Этап № 3. Подтверждение.

На этом этапе определяется правильность и эффективность процесса и определений преобразования. Это осуществляется путем тестирования и оценивания, например, на примере или на копии данных источника, - чтобы выяснить, необходимо ли как-то улучшить эти определения. При анализе, проектировании и подтверждении может потребоваться множество итераций, например, в связи с тем, что некоторые ошибки становятся заметны только после проведения определенных преобразований.

Этап № 4. Преобразования.

На этом этапе осуществляется выполнение преобразований либо в

процессе ETL для загрузки и обновления Хранилища данных, либо при ответе на запросы по множеству источников.

Этап № 5. Противоток очищенных данных.

После того как ошибки отдельного источника удалены, загрязненные данные в исходных источниках должны замениться на очищенные, для того чтобы улучшенные данные попали также в унаследованные приложения и в дальнейшем при извлечении не требовали дополнительной очистки. Для Хранилищ очищенные данные находятся в области хранения данных.

Такой процесс преобразования требует больших объемов метаданных (схем, характеристик данных уровня схемы, определений технологического процесса и др.). Для согласованности, гибкости и упрощения использования в других случаях, эти метаданные должны храниться в репозитории (хранилище) на основе СУБД. Для поддержки качества данных подробная информация о процессе преобразования должна записываться как в репозиторий, так и в трансформированные элементы данных, в особенности информация о полноте и свежести исходных данных и происхождения информации о первоисточнике трансформированных объектов и произведенных с ними изменениях.

Контрольные вопросы:

1. Дайте определение понятию Data Mining.
2. Что представляет технология Data Mining?
3. Какие этапы включает традиционный процесс Data Mining?

Тема 5. Мультиагентные системы

Цель: Изучить основные понятия мультиагентных систем.

План:
1. Мультиагентный подход;
2. Агенты и мультиагентные системы.
3. Распределенные интеллектуальные системы на основе агентов
4. Агенты и МАС

5.1. Мультиагентный подход;

В основе мультиагентного подхода лежит понятие мобильного программного агента, который реализован и функционирует как самостоятельная специализированная компьютерная программа или элемент искусственного интеллекта.

Изначально, до появления соответствующих информационных технологий, "агент" был человеком, которому делегировалась часть полномочий — как в выполнении конкретных функций, так и в принятии решений. В первых (не компьютерных) мультиагентных системах агенты представляли сотрудников компаний, от имени и по поручению которых они взаимодействовали между собой при выполнении определенной задачи — например, представители покупателя и продавца в торговой сети или в других видах бизнеса. Такие системы наследовали многие черты "бюрократической" организации, включая централизацию управления, статичную структуру и узкоспециализированную агентную функциональность. В частности, базовый агент (резидент) получал задачу, декомпозировал её и распределял подзадачи между другими агентами, после чего получал результат и принимал решение — при этом, как правило, большинство агентов занимались исключительно сбором и поставкой информации.

На смену таким системам, копирующим централизованную иерархию, быстро пришли распределенные системы, в которых знания и ресурсы распределялись между достаточно "самостоятельными" агентами, но сохранялся общий орган командного управления, принимающий решения в критических или конфликтных ситуациях. Дальнейшим шагом в этом направлении стала парадигма полностью децентрализованных систем, в которых управление происходит только за счет локальных взаимодействий между агентами. При этом узкая функциональная ориентация агента на решение какой-то одной отдельной части общей задачи постепенно стала уступать место универсальной целостности (автономности). Примерами таких децентрализованных организаций отчасти могут служить колонии насекомых, например, пчел или муравьев.

Суть мультиагентных технологий заключается в принципиально новом методе решения задач. В отличие от классического способа, когда проводится поиск некоторого четко определенного (детерминированного) алгоритма, позволяющего найти наилучшее решение проблемы, в мультиагентных технологиях решение получается автоматически в результате взаимодействия множества самостоятельных целенаправленных программных модулей — так называемых программных агентов.

Зачастую классические методы решения задач либо неприменимы к реальной жизни (нетрудно представить себе, что значит попытаться решить задачу управления предприятием в непредсказуемой динамичной обстановке современного бизнеса, даже с по мощью высшей математики и самых продвинутых экономических моделей), либо они требуют огромных объемов расчетов (для которых не хватит мощности всех современных компьютеров), либо они вовсе отсутствуют.

Значит ли это, что ситуация, когда точный алгоритм решения отсутствует, безнадежна?

Нет — отвечают мультиагентные технологии. В конце концов, людям в повседневной жизни постоянно приходится в условиях дефицита времени и средств решать задачи, не имеющие точного формального решения — и они решаются часто не самым худшим образом.

На рисунке 1 показаны в сравнении две схемы построения программного обеспечения: традиционная и на базе мультиагентной системы. В МАС каждой сущности ставится в соответствие программный агент, который представляет ее интересы.

Традиционное и мультиагентное построение программного продукта

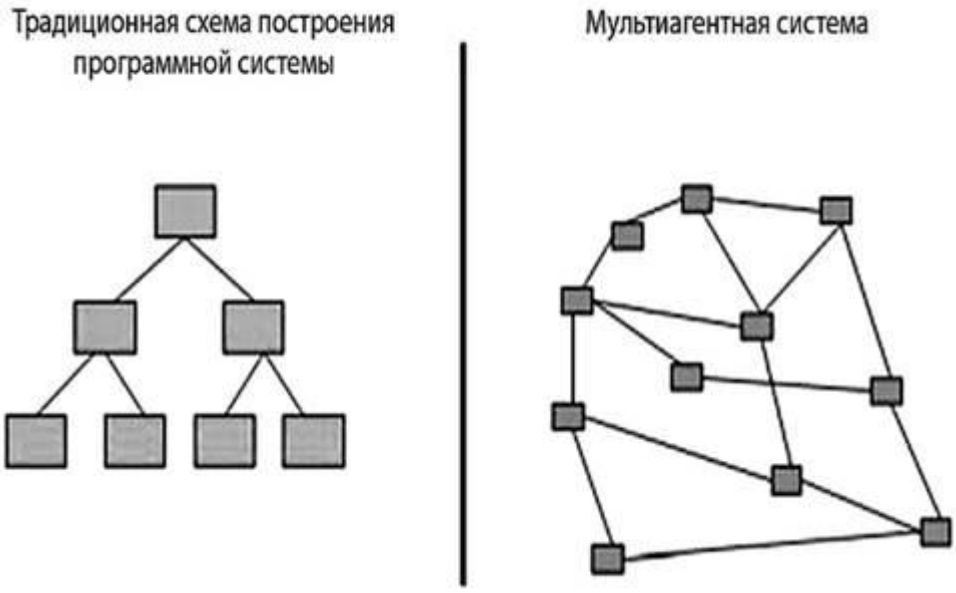

Рисунок 12- Традиционное и мультиагентное построение программного продукта

Человеку присущ интеллект — это его отличает от компьютера, действующего строго по заложенной в него про грамме. А это то, что позволяет человеку ориентироваться в сложной обстановке, иметь дело с нечетко поставленными задачами, адаптироваться к меняющимся условиям. Неопределенность присутствует чаще всего, когда существует набор альтернатив, и невозможно предсказать, какой из вариантов окажется лучшим по прошествии достаточно длительного времени.

При составлении расписаний движения грузовиков, например, это ситуация, когда существует выбор между несколькими грузовиками, перевозящими грузы, несколькими дорогами, которые могут быть использованы для достижения разных точек назначения, и многими водителями, которые могут управлять грузовиками. Каждый из ресурсов (грузовик, дорога и водитель) имеют различные свойства. Неопределенность возрастает в ситуациях, когда возможны непредсказуемые события, такие как изменения в условиях поставок или спроса, аварии или сбои ресурса, задержки, отмены заказов и тому подобное

А есть ли интеллект, скажем, у колонии муравьев или роя пчел? С одной стороны, каждый отдельно взятый муравей или пчела, очевидно, им не обладают. С другой — колония в целом проявляет удивительные образцы поведения, которое во многом может считаться интеллектуальным. Такие ситуации называются проявлением эмерджентного интеллекта, или неожиданных свойств, которыми обладает система, но не обладает ни один входящий в нее отдельный элемент. Возникающий при этом эффект "интеллектуального резонанса" часто так и называют "интеллект роя". Действительно, интеллект и физическая сила одной пчелы не так велики, но рой пчел, согласованно действующий, может победить медведя и даже человека.

Агенты очень похожи на членов команды, которые могут соревноваться друг с другом или сотрудничать в процессе принятия решения. Ключевая особенность эмерджентного интеллекта — динамика и непредсказуемость процесса принятия решений. На практике это означает, что решение достигается за счет сотен и тысяч взаимодействий, которые почти невозможно отследить. Но это и не требуется, поскольку агентам дают цели, которые они должны достигать, но не предопределяют сценарии исполнения задач по достижению этих целей.

Эти сценарии формируются и исполняются агентами самостоятельно. На каждом шаге агенты рассматривают входы системы и реагируют на непредсказуемые события (задержки, сбои, изменения). Реакция может быть самостоятельной, или осуществляться во взаимодействии с оператором. Таким образом, эмерджентный интеллект —

это не есть какой-либо один новый и специально сконструированный уникальный блок-решатель, добавленный к системе. Напротив, это нечто (результат самоорганизации), что возникает как бы "из воздуха" (за счет множества скрытых или явных условий, сложившихся в ситуации), спонтанно и в заранее не предвиденный момент времени, и так же неожиданно исчезает, но в процессе своего существования определяющим образом руководит работой всей системы. Тут мы имеем дело с возникновением порядка из хаоса, с одним из тех явлений, которые изучали и описывали такие выдающиеся ученые, как Александр Богданов (теория организации), Илья Пригожий (самоорганизация в физических системах), Марвин Минский (психология и теория мышления), Артур Кестлер (биология).

5.2. Агенты и мультиагентные системы

В начале XXI века группа ведущих мировых ученых составила список приоритетных задач кибернетики на ближайшие 50 лет Среди них:

- динамически реконфигурируемое интеллектуальное управление сложными системами;
- асинхронная теория управления;
- управление расапределенными объектами через Интернет;
- перепрограммирование системы управления бактериями;
- создание футбольной команды роботов, которая выиграет у победителя кубка мира среди людей.

Мультиагентная система (МАС) кардинально отличаются от традиционных "жестко" организованных систем, и, в перспективе, способны помочь в решении этих задач.

Начало построения моделей и применения искусственных мультиагентных систем на практике было положено в 1960-х годах. В качестве основы были взяты достижения таких областей деятельности человека, как системы искусственного интеллекта (Artificial Intelligence), параллельные вычисления (Parallel Computing), распределенное решение задач (Distributed Problem Solving). Многоагентные системы имеют реальную возможность интегрировать в себе самые передовые достижения перечисленных областей, демонстрируя принципиально новые качества. Сейчас МАС — одно из наиболее динамично развивающихся и перспективных направлений в области искусственного интеллекта.

Открытый характер современного информационного общества и глобальной рыночной экономики приводит к ускорению научно-технического прогресса и обострению конкуренции на рынках. Это заставляет предприятия искать новые методы и средства организации и управления, направленные на более качественное и эффективное удовлетворение индивидуальных запросов потребителей. Большинство

современных систем характеризуются отсутствием средств своевременной идентификации новых потребностей и возможностей в среде, позволяющих предприятию оперативно принимать эффективные решения по реконфигурации производственных, кадровых, финансовых и других ресурсов.

Типичными примерами событий, вызывающих необходимость заново идентифицировать потребности и возможности, являются: появление нового выгодного заказа, для исполнения которого недостаточно собственных ресурсов предприятия, выход из строя части имеющихся ресурсов, а так же изменение критериев принятия решений. Чем выше неопределенность, чем более распределенный характер имеют процессы принятия решения и чем чаще случаются незапланированные события, тем ниже эффективность существующих систем, не способных самостоятельно принимать решения и автоматически перестраиваться под изменения в среде.

Необходимость модификации схемы принятия решений в традиционных системах оказывается сложной и трудоемкой задачей, которая требует высокой квалификации исполнителей. Это делает разработку и эксплуатацию таких систем крайне дорогостоящими. Соответственно, еще одной актуальной проблемой современности становится рост объемов ин формации и степени сложности описания систем.

Для решения подобных проблем применяются мультиагентные технологии, в основе которых лежит понятие "агента", которое в последнее время было адаптировано ко многим областям как прикладного и системного программирования, так и к исследованиям в областях искусственного интеллекта и распределенных интеллектуальных систем. Причем в каждом конкретном случае понятию придается несколько разное значение.

Первоначально идея создания интеллектуального по средника (агента) "возникла в связи с желанием упростить стиль общения конечного пользователя с компьютерными программами, поскольку доминирующий, в основном, и ныне стиль взаимодействия пользователя с компьютером предполагает, что пользователь запускает задачу явным образом и управляет ее решением. Но это совершенно не подходит для неискушенного пользователя". Иначе говоря, сначала идея интеллектуального посредника возникла как попытка интеллектуализации пользовательского интерфейса.

Развитие методов искусственного интеллекта позволило сделать новый шаг к изменению стиля взаимодействия пользователя с компьютером. Возникла идея создания так называемых "автономных агентов", которые породили уже новый стиль взаимодействия пользователя с программой. Вместо взаимодействия, инициируемого пользователем путем команд и прямых манипуляций, пользователь вовлекается в совместный процесс решения. При этом, как пользователь, так и компьютерный посредник, оба принимают участие в запуске задачи, управлении событиями и решении

задачи. Для такого стиля используется метафора персональный ассистент, который сотрудничает с пользователем в той же рабочей среде.

Словари дают следующее толкование слова агент: "некто или нечто, прикладывающее усилия для достижения эффекта". Такое самое общее определение указывает на первый признак агента — агенты совершают действия. Часто утверждается, что агенты не просто совершают действия, но они действуют автономно и рационально. Под автономностью обычно понимают, что агент действует без прямого вмешательства человека или другой управляющей сущности. Под рациональностью понимают стремление агента оптимизировать значение некоторой оценочной функции. Мера рационально стинеявно указывает на то, что агент имеет цели (желания — англ. desires), которых агент "хочет" достичь, и представления о внешнем мире (убеждения — англ. beliefs), на которые агент опирается при выборе действия (реализации намерений — англ. intentions: множество избранных, совместимых и достижимых желаний).

Еще одним важным свойством агента является то, что он помещен во внешнюю среду, с которой он способен взаимодействовать. Обычно, среда не контролируется агентом, он лишь способен влиять на нее. Разделение намерений и желаний необходимо, так как агент может иметь несовместимые желания или желания могут быть недостижимы. Поскольку агент ограничен в ресурсах и не может достичь всех желаний одновременно, естественно выбирать наиболее значимые цели — намерения. Итак, агент — разумная сущность, помещенная во внешнюю среду, способная взаимодействовать с ней, совершая автономные рациональные действия для достижения целей, т. е. Интеллектуальный агент — это агент, обладающая следующими свойствами:

- реактивность (англ. reactivity) — агент ощущает внешнюю среду и реагирует на изменения в ней, совершая действия, направленные на достижение целей;
- проактивность (англ. pro-activeness) — агент показывает управляемое целями поведение, проявляя инициативу, совершая действия направленные на достижение целей;
- социальность (англ. social ability) — агент взаимодействует с другими сущностями внешней среды (другими агентами, людьми и т. д.) для достижения целей.

При разработке системы каждое из первых двух свойств достигается достаточно легко. Наибольшую сложность представляет совмещение в системе обоих свойств в нужных пропорциях. Будет не слишком эффективно, если агент жестко следует сценарию достижения цели, не реагируя на изменения во внешней среде и не обладая способностью заметить необходимости корректировки плана. Но также не эффективно будет и поведение, ограниченное лишь реакцией на поступающие из вне стимулы, без какого-либо планирования целенаправленных действий.

На самом деле описанная проблема настолько сложна, что даже далеко не все люди способны эффективно ее решать. Очень часто можно увидеть чело века, который кидается на каждую подвернувшуюся возможность, но никогда не доводит ничего до конца, т. к. не концентрируется на этой возможности достаточное время, чтобы полноценно ее реализовать. Но также часто встречаются люди, которые, однажды поставив цель и сформировав план, будут пытаться принципиально ему следовать, не замечая изменений в ситуации, требующих пересмотра целей или планов.

Достичь свойства социальности тоже нелегко. Социальность — это не просто обмен данными. Помимо коммуникации, социальное поведение должно включать кооперацию с другими сущностями, заключающуюся в разделении целей между отдельными сущностями, совместном планировании и координации действий, направленных на достижение общих целей. Социальное поведение, как минимум, предполагает наличие у агента представлений о целях других сущностей и том, как они планируют этих целей достичь.

Сложность формулирования содержательных практически значимых задач и невозможность априорного точного задания всех условий функционирования выдвигают адаптивные постановки проблем, отдельно выделяя такую особенность агентов, как адаптивность — способность автоматически приспосабливаться к неопределенным и изменяющимся условиям в динамической среде.

Таким образом, предшественниками программных агентов можно считать сложные адаптивные системы, которые умеют подстраиваться под ситуацию или обстоятельства и принципиальным образом менять свое поведение или характеристики, чтобы обеспечить решение стоящих перед ними задач. Однако в случаях, когда агент функционирует в сложной, постоянно изменяющейся среде, взаимодействуя при этом с другими агентами, такая мультиагентная система значительно сложнее просто адаптивной системы, так как она быстрее обучается и может действовать эффективнее за счет перераспределения функций или задач между агентами.

Сложные системы часто рассматривают как среду действия агентов. С понятием сложных систем связаны следующие фундаментальные идеи, которые непосредственно влияют на функционирование МАС:

- в сложных системах существуют автономные объекты, которые взаимодействуют друг с другом при выполнении своих определенных задач;

□□агенты должны иметь возможность реагировать на изменяющиеся условия среды, в которой они функционируют и, возможно, изменять свое поведение на основе полученной информации;

□□сложные системы характеризуются возникающими структурами — логически связанными схемами, которые формируется в результате взаимодействия между агентами;

сложные системы с возникающими структурами часто существуют на грани порядка и хаоса;

при создании сложных систем на базе агентов имеет смысл рассматривать биологические аналогии, такие как: паразитизм, симбиоз, репродукцию, генетику, митоз и естественный отбор (например, компания British Telecom при формировании сети направления звонков использует модель деятельности колонии муравьев).

Концепция агентов, разработанная в рамках мультиагентных технологий и мультиагентных систем, предполагает наличие активного поведения агентов, т.е. способности компьютерной программы самостоятельно реагировать на внешние события и выбирать соответствующие действия. Сегодня агентные технологии предлагают различные типы агентов, модели их поведения и свойства, семейство архитектур и библиотеки компонентов, ориентированные на современные требования.

В настоящее время не существует устоявшегося определения агента. Ниже перечислены некоторые из них:

"Агент — это аппаратная или программная сущность, способная действовать в интересах достижения целей, поставленных пользователем";

"Под агентом можно понимать самостоятельную программную систему, состоящую из программ-объектов, имеющую возможность принимать воздействие из внешнего мира, определять свою реакцию на это воздействие и в соответствии с этим формировать ответное действие. Такие агенты способны действовать, "рассуждать" и обмениваться данными друг с другом в сети для формирования индивидуальных или коллективных решений";

По определению Кристиана Доннегара (директора по технологии компании Living Systems, занимающейся созданием систем совместной коммерции на основе технологии агентов): "агенты — программные объекты, которые выполняют определенные упреждающие и корректирующие действия в соответствии с заданиями, делегированными человеком";

Алан Кэй, который начал первым развивать теорию агентов, определил агент как "программу, которая после получения задания способна поставить себя на место пользователя и действовать по адаптивному сценарию. Если же агент попадает в тупик, он может задать пользователю вопрос, чтобы определить, каким образом ему необходимо действовать дальше";

Программные интеллектуальные агенты – это новый класс систем программного обеспечения, которое действует либо от лица пользователя, либо от лица системы делегировавшей агенту полномочия на выполнение тех или иных действий. Они являются, по сути, новым уровнем абстракции, отличным от привычных абстракций типа – классов, методов и функций. Но при этом, разработка МАС позволяет создавать системы обладающие

расширяемостью/масштабируемостью, мобильностью/переносимостью, интероперабельностью, что несомненно очень важно при разработке систем, основанных на знаниях.

Области знания и технологии, используемые интеллектуальными агентами

Рисунок 13- Области знания и технологии, используемые интеллектуальными агентами

Простая компьютерная программа отличается от агента тем, что "не утруждает" себя целевым поведением и анализом достигнутых результатов. Напротив, агент, представляющий интересы пользователя, "заинтересован" в том, чтобы задание было выполнено. В случае неудачи или какого-то сбоя он должен повторить попытку позднее или иметь про запас альтернативный вариант решения проблемы. Агенты в процессе отработки заданий всегда формирует список выполненных действий, результаты тестирования и верификации и отсылают его в управляющую систему.

Отметим, однако, что вопрос по определению того, что такое агент не закрыт до сих пор, и обсуждение этого вопроса периодически выносится на конференции самого высокого уровня. На рисунке 2 показаны области знания и технологии, с помощью которых формируются механизмы искусственного интеллекта и применения мультиагентных систем.

На основании изложенного выше мы можем скомпилировать следующее определение: "агент — это самостоятельная программная система:

имеющая возможность принимать воздействие из внешнего мира;

□□определяющая свою реакцию на это воздействие и формирующая ответное действие;

□□изменяющая свое поведение с течением времени в зависимости от накопленной информации и извлеченных из нее знаний,

□□обладающая мотивацией и способная после делегирования полномочий пользователем поставить себя на его место и принять решение, соответствующее ситуации".

Интеллектуальный агент должен обладать следующими свойствами:

□□автономность — способность функционировать без вмешательства со стороны своего владельца и осуществлять контроль внутреннего состояния и своих действий;

□□адаптивность — агент обладает способностью обучаться;

□□коллаборативность — агент может взаимодействовать с другими агентами несколькими способами, играя разные роли;

□□способность к рассуждениям — агенты могут обладать частичными знаниями или механизмами вывода, а также специализироваться на конкретной предметной области;

□□коммуникативность — агенты могут общаться с другими агентами;

□□мобильность — способность передачи кода агента с одного сервера на другой;

□□социальное поведение — возможность взаимодействия и коммуникации с другими агентами;

□□реактивность — адекватное восприятие среды и соответствующие реакции на ее изменения;

□□активность — способность генерировать цели и действовать рациональным образом для их достижения;

□□наличие базовых знаний — знания агента о себе, окружающей среде, включая других агентов, которые не меняются в рамках жизненного цикла агента;

□□наличие убеждений — переменная часть базовых знаний, которые могут меняться во времени;

□□наличие цели — совокупность состояний, на достижение которых направлено текущее поведение агента;

□ наличие желаний — состояния и/или ситуации, достижение которых для агента важно;

□□наличие обязательств — задачи, которые берет на себя агент по просьбе и/или поручению других агентов;

□□наличие намерений — то, что агент должен делать в силу своих обязательств и/или желаний.

Иногда в этот же перечень добавляются и такие человеческие свойства, как рациональность, правдивость, благожелательность.

Мультиагентная система (МАС) — сложная система, в которой функционируют два или более интеллектуальных агентов. Процесс самоорганизации в мультиагентных системах — внутренняя упорядоченность, согласованность, взаимодействие более или менее дифференцированных и автономных агентов агентной системы, обусловленной ее строением. Таким образом, в МАС несколько агентов могут общаться, передавать друг другу некоторую информацию, взаимодействовать между собой и решать поставленную. В такой системе задачи (или подзадачи) распределены между агентами, каждый из которых рассматривается как член группы или организации. Распределение задач предполагает назначение ролей каждому из членов группы, определение меры его "ответственности" и требований к "опыту".

Основой формой организации взаимодействия между агентами, характеризующаяся объединением их усилий для достижения совместной цели при одновременном разделении между ними функций, ролей и обязанностей является кооперация.

В общем случае это понятие можно определить формулой: {кооперация = сотрудничество + координация действий + разрешение конфликтов}. Под координацией обычно понимается управление зависимостями между действиями. Коммуникация между искусственными агентами зависит от выбранного протокола, который представляет собой множество правил, определяющих, как синтезировать значимые и правильные сообщения. Фундаментальными особенностями группы, составленной из агентов, сотрудничающих для достижения общей цели, являются социальная структура и распределение ролей между агентами.

Укрупнённая структура агента

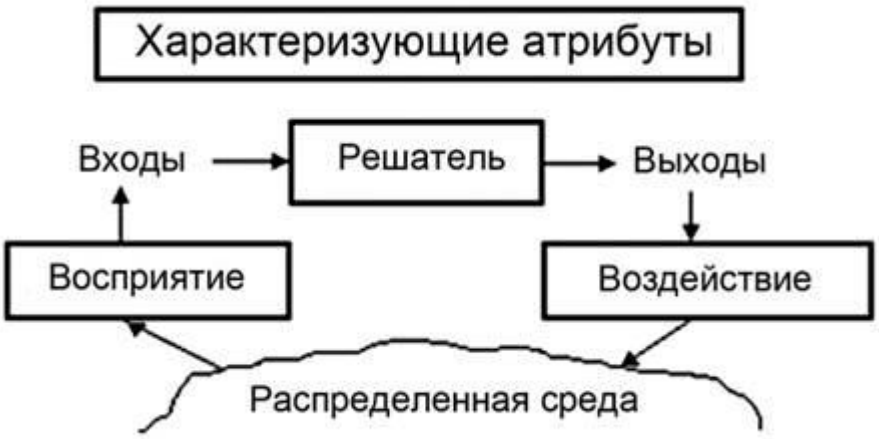

Рисунок 14 - Укрупнённая структура агента

Основой архитектуры агента является контекст, или серверная среда, в котором он исполняется. Каждый агент имеет постоянный идентификатор — имя. В серверной среде может исполняться не только исходный агент, но и его копия. Агенты способны самостоятельно создавать свои копии, рассылая их по различным серверам для исполнения работы. По прибытии агента на следующий сервер его код и данные переносятся в новый контекст и стираются на предыдущем местонахождении. В новом контексте агент может делать все, что там не запрещено. По окончании работы в контексте агент может переслать себя в другой контекст или по исходящему адресу отправителя. Агенты способны также выключаться ("умирать") сами или по команде сервера, который переносит их после этого из контекста в место, предназначенное для хранения.

Входами являются внутренние параметры агента и данные о состоянии среды. Выходы — параметры, воздействующие на среду и информирующие пользователя (или программу, выполняющую роль менеджера в системе) о состоянии среды и принятых решениях. Решатель — процедура принятия решений. Решатель может быть достаточно простым алгоритмом или элементом системы искусственного интеллекта.

Архитектура ядра мультиагентной системы

Рисунок 15 - Архитектура ядра мультиагентной системы

В архитектуре МАС основную часть составляет предметно-независимое ядро, в составе которого выделяются следующие базовые компоненты:

- служба прямого доступа обеспечивает непосредственный доступ к атрибутам агентов;
- служба сообщений отвечает за передачу сообщений между самим агентами, а также между агентами и дополнительными системами ядра;
- библиотека классов агентов (часть базы знаний) содержит информацию о классификации агентов в данной MAS.

□□сообщество агентов — серверное "место", где размещаются агенты; этот блок, кроме жизнедеятельности агентов, обеспечивает еще функции по загрузке/записи агентов и их свойств и за оптимизацию работы агентов с ресурсами.

□□онтология — предметная база знаний, содержащая конкретные знания об объектах и среде функционирования, представляемые в виде соответствующей семантической сети.

Общая методология восходящего эволюционного проектирования МАС может быть представлена цепочкой: {среда – функции МАС – роли агентов – отношения между агентами – базовые структуры МАС – модификации}, и включает следующие этапы:

□□формулирование назначения (цели разработки) МАС;

□□определение основных и вспомогательных функций агентов в МАС;

□□уточнение состава агентов и распределение функций между агентами, выбор архитектуры агентов;

□ выделение базовых взаимосвязей (отношений) между агентами в МАС;

□□определение возможных действий (операций) агентов;

□□анализ реальных текущих или предполагаемых изменений внешней среды.

При проектировании организацию агентов можно рассматривать как набор ролей, находящихся между собой в определенном отношении, и взаимодействующих друг с другом. Таким образом, методология восходящего проектирования МАС требует предварительного задания исходных функций (ролей агентов), определения круга их обязательств по отношению друг к другу, формирования исходных и развивающихся структур на основе выделенных функций, и исследования адекватности этих структур характеру решаемых задач в выделенных проблемных областях

Главная идея нисходящего проектирования состоит в определении общих социальных характеристик МАС по некоторому набору критериев, построении базовых типов их организаций с последующим определением требований к архитектуре агентов. Когда речь идет о "выращивании" искусственных социальных систем и сообществ, на первый план выдвигается нисходящий подход к организационному проектированию.

5.3Распределенные интеллектуальные системы на основе агентов

Современные системы искусственного интеллекта часто строятся как системы взаимодействующих и сотрудничающих агентов.

Одним из расширений понятия программы стало понятие агента. Оно появилось в связи с использованием программ не только для решения

численных задач или иной подобной обработки информации, но и в системах реального времени (real-time systems).

Обычная вычислительная программа, например, программа расчета колебаний струны, запускается на ЭВМ, получает входные данные, выполняется как последовательность заранее написанных команд, выводит результат. До следующего запуска извне (если он вообще состоится) она пребывает в состоянии "комы". В программе расчета колебаний струны, конечно, фигурирует такой параметр как время, но он используется наряду с другими параметрами, такими как амплитуда, и никакого отношения к реальному времени, в котором выполняется программа, не имеет.

Программа реального времени функционирует (запускается, приостанавливается, возобновляется, завершается, запускается вновь) в соответствии с внешними, физическими, а не абстрактными, часами. Это ее главный отличительный признак.

Кроме того, физическое время чаще всего является одним из существенных параметров расчета. Программа реального времени взаимодействует с окружением (получает и передает информацию) посредством интерфейсов, т.е. она влияет на окружающую среду и испытывает, в свою очередь, влияние окружающей среды. Отсюда следует второй важный отличительный признак программы реального времени: потенциально она может оценивать результаты своего выполнения (обратная связь) и менять свои действия в будущем с учетом прошлого.

Таким образом, можно говорить о поведении программы реального времени по отношению к окружению. В связи с этим программа реального времени уже не рассматривается как элемент stand-alone системы, она является частью системы, имеющей признаки распределенной системы.

Эти и другие важные отличия от обычной программы привели к возникновению понятия программного агента и более широкого понятия – автономного агента.

В традиционных системах программы могут правильно функционировать только в условиях полностью предсказуемого окружения. Для того чтобы выполнить с помощью программы некоторое вычисление пользователь должен задать всю входную информацию (аргументы). Если программа берет исходные данные из базы данных, то должна быть полная уверенность в том, что эти данные имеются в базе.

Программные агенты обладают некоторыми знаниями об окружающем мире, которые позволяют им самим решать небольшие проблемы без вмешательства пользователя. Например, Интернет-агенты могут самостоятельно отыскивать нужную информацию в Сети (разумеется, в определенных пределах). Таким образом, агенты – это более гибкая конструкция, чем традиционные программы. Дополнительно, если агент может общаться с другими агентами, то они могут помочь ему решить проблему, действуя совместно.

Как и в случае классических алгоритмов, когда мы обсуждаем интуитивное понятие алгоритма и его формализации, можно обсуждать интуитивное понятие агента и различные его формализации.

Начнем с интуитивного понятия агента. Несмотря на рост числа теоретических исследований и приложений технологий, основанных на применении агентов, так и нет общепринятого определения термина "агент". Существует несколько трактовок, зависящих от целей исследования.

В частности, П.Маес из Media Lab Массачусетского технологического института считает, что "автономные агенты – это вычислительные системы, которые существуют в сложном динамическом окружении, чувствуют и действуют в этом окружении автономно, и так, чтобы реализовать множество целей и задач, для которых они спроектированы".

Б.Хайес-Рот считает, что "мыслящие агенты непрерывно выполняют три функции: восприятие динамически изменяющихся условий окружения; действия по влиянию на окружение; логический вывод для интерпретации получаемой информации, решения проблем, построения заключений и определения действий".

М.Вулдридж и Н.Дженнингс разделяют слабое и сильное понятия агента. Слабый агент рассматривается как система со свойствами автономности, возможности работать в обществе (social ability), реактивности, целенаправленности. Сильное понятие агента предполагает возможность непредписанных действий, предпринимаемых на основе собственных "идей".

Их определение включает следующие свойства: 1) автономия: агенты действуют без непосредственного вмешательства людей или чего-либо еще, определенным образом контролируют свои действия и внутреннее состояние; 2) социальные возможности: агенты взаимодействуют с другими агентами (и, возможно, людьми) с помощью некоторого языка; 3) реактивность: агенты воспринимают свое окружение (которое может быть физическим миром, пользователем – через графический пользовательский интерфейс, набором других агентов, Интернетом, или комбинацией всего этого) и отвечают периодически так, чтобы изменить что-то в своем окружении; 4) целенаправленность: агенты не просто отвечают своему окружению, они способны на целенаправленное поведение и проявление инициативы.

Т.Хесс и др. разделяют свойства на базовые и продвинутые. К базовым относятся целенаправленность, постоянство функционирования и реактивность. Продвинутые свойства включают наличие искусственного интеллекта, мобильность и интерактивность.

С.Франклин и А.Грэссер в 1996 году предложили следующее обобщенное определение агента:

Автономный агент – это система, находящаяся внутри окружения и являющаяся его частью, воспринимающая это окружение (его сигналы) и

воздействующая на окружение для выполнения собственной программы действий.

Расшифровка этого определения дается перечнем свойств, которыми должен обладать автономный агент:

1. реактивность;
2. автономность;
3. целенаправленность;
4. непрерывность функционирования;
5. коммуникативность;
6. обучаемость (адаптивность);
7. мобильность;
8. гибкость;
9. индивидуальность.

Разные авторы не совсем одинаково трактуют перечисленные свойства. Попытаемся объяснить их подробнее.

Свойство реактивности означает, что агент временами отвечает на изменения в окружении. Агент имеет сенсоры, с помощью которых получает информацию от окружения. Сенсоры могут быть самыми различными. Это могут быть микрофоны, воспринимающие акустические сигналы и преобразующие их в электрические, видеокарты захвата изображений, клавиатура компьютера или общая область памяти, в которую окружение помещает данные и из которой программный агент берет данные для вычислений.

Не все изменения окружения становятся известными (доступными) сенсорам агента. Это вполне естественно. Ведь и человек не воспринимает звуки частотой свыше 30 кГц, радиоволны и т.д. Таким образом, окружение не является полностью наблюдаемым для агента.

Аналогично, агент воздействует на окружение путем разнообразных исполнительных механизмов, включая общую память. Разумеется, степень воздействия как и степень восприятия является ограниченной. Агент может перевести окружение из некоторого состояния в некоторое другое, но не из любого в любое.

Свойство автономности означает, что агент является самоуправляющимся, сам контролирует свои действия. Программный агент, находящийся на некотором сервере, обладает возможностью "самозапуска". Он не требует от пользователя каких-либо специальных действий по обеспечению его старта (подобно тому, как мы "кликаем" два раза по иконке некоторого файла).

Свойство целенаправленности означает, что у агента имеется определенная цель и его поведение (воздействие на окружение) подчинено этой цели, а не является простым откликом на сигналы из окружения. Иначе говоря, агент является управляющей системой, а не управляемым объектом.

Свойство непрерывности функционирования означает, что агент постоянно находится в состоянии исполнения. Это не значит, что он ежесекундно выдает в окружение какие-то сигналы. Он может прекратить активные действия, "заснуть", до некоторого момента времени или какого-то события. Но в необходимый момент времени он сам возобновляет действия, не требуя какого-то "включения" извне.

Свойство коммуникативности означает, что агент общается с другими агентами (включая людей), используя для этого некоторый язык. Это не обязательно единый язык для всех агентов. Достаточно, чтобы у пары общающихся агентов был общий язык. Язык может быть сложным как, например, естественный язык. Но может быть и примитивным: обмен числами или короткими словами. Если многословные фразы сложного языка несут всю информацию, как правило, в себе, то слова простого языка предполагают "умолчание": обе стороны диалога "знают", о чем идет речь (как в известном анекдоте о занумерованных анекдотах).

Свойство обучаемости означает, что агент может корректировать свое поведение, основываясь на предыдущем опыте. Это не просто накопление в памяти параметров окружения, т.е. использование исторических данных, но сопоставление истории собственных действий с историей их влияние на окружение, и изменение в связи с этим своей программы действий.

Свойство мобильности означает, что агент может транспортировать себя с одной машины на другую. В общем случае можно сказать, что агент меняет свое положение в окружении (не меняя самого окружения).

Свойство гибкости означает, что действия агента не предписаны. Это, конечно, весьма относительное утверждение. Имеется в виду, что программа, заложенная в агента, предполагает выбор из многочисленных вариантов поведения. Такого рода выбор производится если не на каждом шагу, то достаточно часто, и зависит от выполнения или невыполнения различных условий. Выбор не является ни случайным, ни недетерминированным. Гибкость означает, что действия агента не примитивны, не легко предсказуемы.

Свойство индивидуальности означает наличие персональных свойств, данных, возможно, "эмоционального" состояния.

Систему, состоящую из нескольких взаимодействующих агентов, называют мультиагентной системой (МАС). Если мультиагентная система, в свою очередь, по отношению к ее окружению действует как (единый) агент, то естественно, составляющих ее агентов назвать подагентами (субагентами).

В мультагентной системе необязательно все агенты взаимодействуют (общаются) между собой. В крайнем случае, общения нет вообще. Такие системы назовем дискретными мультиагентными системами. Второй крайний случай – каждый агент общается с каждым. Такую систему назовем полносвязной мультиагентной системой.

Мультиагентная система, действующая как единый агент, должна характеризоваться и некоторой общей для всех субагентов целью и координацией действий по достижению этой цели.

Поскольку встречаются и другие ситуации, когда агенты не связаны столь тесно, то такие системы можно назвать обществами агентов. Отсутствие единой цели, однако, не отрицает возможного группового поведения агентов. Но оно является, скорее, эпизодическим, чем систематическим.

Важным отличием мультиагентной системы от программы или одного агента является то, что входящие в систему программные агенты (по крайней мере, некоторые) не были спроектированы специально для этой системы. Может быть, это – повторно используемые агенты, или агенты, разработанные для решения более универсальных задач. В этих случаях агенты имеют собственные цели, не совпадающие полностью с целями системы (организации), но совместимые с ними. Тем не менее, они могут быть полезны друг другу для решения стоящих перед ними задач и, поэтому, очень важным для них с этой точки зрения является свойство коммуникативности.

С организационной точки зрения существуют общие цели всего сообщества, и эти общие цели выражаются, прежде всего, в ролях (которые играют агенты) и нормах взаимодействия.

Исследования в области систем поддержки принятия решений (DSS) в последние годы все больше переходят от создания систем в виде традиционного "ящика с инструментами" (toolbox) к парадигме сотрудничества и интеграции независимых приложений. Быстро растущая область исследований интеллектуальных агентов и мультиагентных систем предлагает возможности создания более эффективных систем на основе единого подхода. Например, Р.Вахидов и Б.Фазлоллахи предлагают плюралистическую мультиагентную систему поддержки принятия решений. Слово "плюралистический" в названии подхода говорит о множественности, различности источников информации, точек зрения и перспектив, используемых в рамках одной системы. Агенты распределены по ролям в соответствии с различными фазами решения проблем. Они взаимодействуют с пользователем, окружением, друг с другом для улучшения процесса принятия решения.

5.4Агенты и МАС

Развитие интеллектуальных агентов и МАС очень популярны в среде исследователей ИС.

В области ИС интеллектуальные агенты используются, прежде всего, для интеграции информационных систем, пользователей, оборудования, для поддержки принятия решений, управления знаниями.

Существенными в понятии агента являются понятия делегирования и автономии. Это означает ограниченное непосредственное управление и значительный плюрализм (разнообразие) в системах, в которых работают сразу несколько агентов.

Исследования в области МАС отличаются от исследований в области распределенного ИИ. В ИИ функционирование системы распределено по множеству вершин для совместного решения проблемы. В МАС агенты имеют свои собственные цели, которые могут вступать в противоречие с целями других агентов. Тем не менее, группа логически децентрализованных агентов решает работать вместе для достижения общей цели. Это возможно сделать различными путями. Один из путей – разделить задачу на части между агентами. Другой – позволить каждому из агентов решать исходную задачу своим способом. В случае отыскания приближенных решений затем может быть выбрано наилучшее. Либо может применяться процедура голосования агентов за то или иное решение.

В традиционной клиент-серверной модели приложения выполняются на клиентских машинах, которые отправляют время от времени запросы одному или нескольким серверам объектов, таким, например, как серверы базы данных. Сервер объектов посылает клиенту ответ на запрос. В ответе могут содержаться объекты, полученные с другого сервера. Таким образом, основой для коммуникации является передача и прием сообщений.

Парадигма мультиагентного проектирования и реализации систем состоит в том, что программные агенты для достижения цели (выполнения некоторой работы) перемещаются с одного сервера на другой. Агенты выполняют свою работу локально на том сервере, на котором они в данный момент находятся. Обмен сообщениями (между серверами) по сети агенты, как правило, не используют. Т.Комийя, Т.Енокидо и М.Такидзава рассмотрели следующую модель.

Пусть в системе имеются серверы $Serv_1$, $Serv_2$, …, $Serv_m$, соединенные каналами связи, и агенты A_1, A_2, …, A_n, пользующиеся услугами этих серверов. Агенты автономно выполняются на серверах. Агент самостоятельно инициализируется на сервере: процедура и данные агента записываются в память сервера объектов, если на этом сервере достаточно ресурсов для работы агента. Говорят, что агент A_k размещается (land – "приземляется") на сервере $Serv_i$ и этот сервер становится для агента текущим.

Для размещения на сервере важны два условия:

1. достаточность ресурсов сервера таких, как объем доступной памяти и вычислительная мощность компьютера (с учетом

того, что часть памяти и вычислительных мощностей сервера в этот момент времени может быть занята выполнением других задач);

2. отсутствие на сервере в этот момент времени других агентов, конфликтующих с агентом Ak. Конфликт может быть вызван различными причинами, например, тем, что два агента будут работать с одним и тем же объектом в режиме изменения этого объекта (в режиме записи).

После выполнения работы с объектами на сервере $Serv_i$ агент A_k может переместиться на сервер $Serv_j$ для выполнения работы с находящимися там объектами. Перемещения агента зависят от того, какая перед ним стоит цель и на каких серверах находятся нужные ему объекты.

Агенты выполняют определенные операции над объектами, но и сами агенты могут быть объектами операций. Следующие операции обычно входят в перечень операций над агентами:

1. создание агента (по некоторому описанию). Создание происходит всегда на некотором сервере. Этот сервер называют "домашним" для созданного агента;

2. создание копии A^c агента A. Возникает дополнительный агент с теми же процедурами и данными, которыми обладает агент A на момент копирования. С этого момента будут существовать два агента. Сначала они тождественны и находятся на одном сервере (создавшем копию), но в дальнейшем они могут переместиться на различные серверы и даже видоизмениться независимо друг от друга.

3. расслоение агента A. Агент A разделяется на несколько агентов A_1, A_2, …, A_l, начинающих самостоятельное существование в системе. Это разделение может коснуться программ и данных агента, а также его целей и задач;

4. слияние агентов A_1, A_2, …, A_l. Вместо нескольких агентов, прекращающих самостоятельное существование, возникает один агент, обобщающий процедуры и данные составляющих агентов, а также их цели и задачи;

5. уничтожение агента A. Агент A перестает существовать в системе.

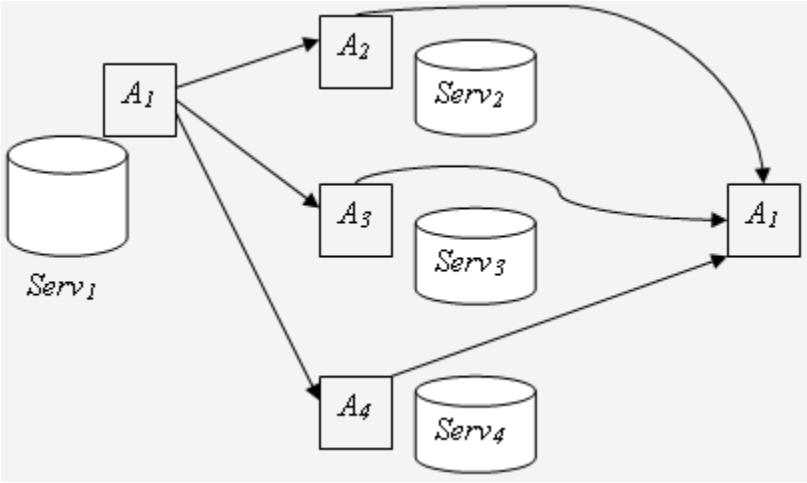

Рисунок 16 Расслоение и слияние агентов

Модель агентов важна, например, для обработки транзакций. Как известно, транзакция – это последовательность взаимосвязанных действий, которые должны быть выполнены на нескольких серверах распределенной системы.

Транзакция должна удовлетворять четырем условиям (их называют ACID):

1. атомарности (atomicity). Работа, которую должна выполнить транзакция, не может быть сделана "наполовину". Либо работа выполнена полностью, либо она не выполнена совсем. Это не означает, что транзакция не может состоять из субтранзакций. Если из всех транзакций, которые должны были выполниться, не выполнится хотя бы одна, то считается невыполненной вся транзакция. Должен быть произведен откат (roll back) – последовательность действий, возвращающая сервер (или базу данных) в исходное (до начала выполнения транзакции) состояние;
2. согласованности (consistency). Транзакция должна переводить базу данных (или сервер) из одного согласованного состояния в другое согласованное состояние;
3. изолированности (isolation). Транзакции изолированы друг от друга, кроме случая, когда одна транзакция является субтранзакцией другой. Транзакции либо выполняются последовательно, либо (что вполне возможно в распределенных системах) пересекаются по времени, но не пересекаются по данным. Иными словами, никакие данные, которые читает или изменяет транзакция T_1, не должны быть доступны для изменения или чтения транзакцией T_2 до тех пор, пока T_1 не завершилась.
4. продолжительности (durability). Результаты зафиксированной (завершенной) транзакции должны сохраняться до тех пор, пока их не заменит новая транзакция. Здесь подразумевается, что попытка пользователя изменить данные приводит не к

немедленному их изменению, а к инициированию соответствующей транзакции.

Агент может "сопровождать" выполнение транзакции на различных серверах, обеспечивая соблюдение перечисленных свойств. Либо, агенты на разных сайтах могут отвечать за выполнение субтранзакций данной транзакции.

Агенты могут помогать и в решении проблем работы с документами для мобильных пользователей.

Мобильные пользователи обычно находятся в условиях относительно низкой полосы пропускания каналов связи и не слишком плотной (с точки зрения количества связей между узлами) сети. Это затрудняет, в частности, работу с объемными Web-документами, передавать которые целиком на терминал пользователя неэффективно.

Решением может быть (С.Яу, Х.Леонг, А.Си) "нарезка" документа на части и передача непосредственно пользователю только той части, с которой он работает в данный короткий отрезок времени. Таким образом, будет решена проблема недостаточной пропускной способности канала связи.

Для того чтобы пользователю своевременно была предоставлена следующая необходимая часть, с учетом его возможного перемещения от одной базовой станции к другой, используется агент. Агент управляет документом в целом, перемещается между серверами базовых станций вслед за пользователем (но не на его терминале) и перемещает документ, организует пересылку пользователю необходимого фрагмента документа с текущей базовой станции в необходимое время. Можно сказать, что агент создает и поддерживает для пользователя виртуальное соединение с документом.

Серверы базовых станций соединены высокоскоростными каналами с большой пропускной способностью, поэтому перемещение по ним агента с документом не вызывает трудностей. Агент, как и пользователь, является мобильным, движется "параллельно" пользователю, но, в отличие от пользователя, по хорошей "дороге".

Сказанное иллюстрирует на рисунке17 на котором изображены две базовые станции с зонами покрытия (пунктирные линии), перемещение мобильного пользователя из одной зоны в другую, и перемещение агента с документом. С базовых станций с участием агента пользователь получает фрагменты документа.

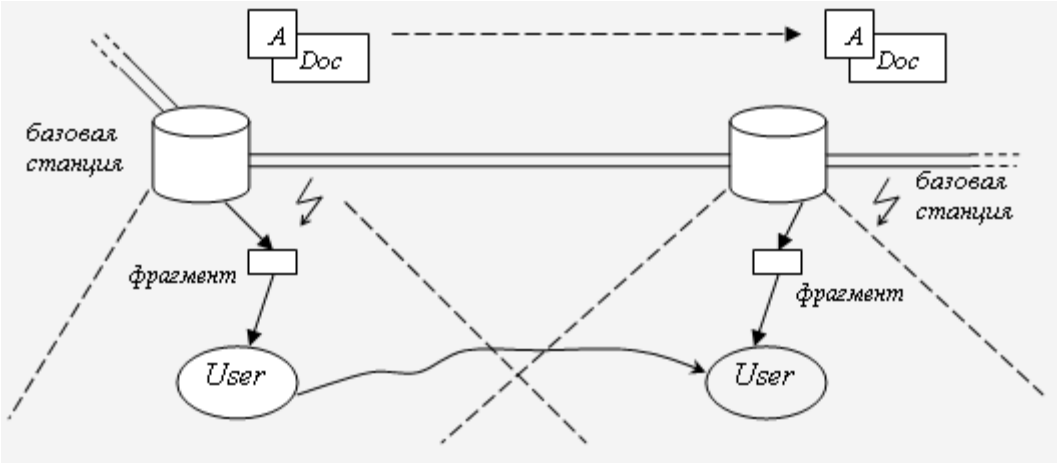

Рисунок 17 Перемещение мобильного пользователя и мобильного агента

Взаимодействующие агенты могут быть частью системы искусственного интеллекта, участвовать в распределенном решении задач и распределенном планировании.

Распределенная группа агентов работает над решением задачи совместно. В связи с тем, что такие ресурсы как знания, умения, возможности, информация у каждого агента свои, то один агент не может решить задачу в целом или, по крайней мере, не может решить ее также хорошо, как это возможно при сотрудничестве агентов.

Решение распределенных задач требует групповой согласованности действий агентов – сотрудничества и компетентности каждого агента в отдельности, т.е. обладания некоторым "ноу-хау". Если агенты имеют собственные цели, то достичь согласованности непросто. Но обычно предполагается, что агенты создаются для решения общей распределенной задачи или могут достичь своей частной цели, только решив общую задачу.

Разумеется, распределенное решение задачи возможно и целесообразно, если задача обладает внутренним параллелизмом (не все действия по ее решению укладываются в строгую последовательность). Кроме этого предполагается, что одновременно могут выполняться разнородные действия с разнородными ресурсами, находящимися в разных точках пространства.

Примером такой задачи является построение сети датчиков интенсивности автомобильного движения на большом пространстве, например, города или района города, и организация на основе этой сети управления движением.

Действительно, это типичная распределенная задача. Имеется карта города с регулируемыми (с помощью светофоров) перекрестками. При интенсивном движении автомобилей перекресток работает как четыре взаимосвязанных системы массового обслуживания (по одной на каждое

направление движения), что приводит к автомобильным "пробкам" (очередям) на улицах, прилегающих к перекрестку. "Пробка", возникшая у одного перекрестка, может увеличиться до такой степени, что захватит и смежный перекресток.

Из теории массового обслуживания известно, что величина очереди зависит от интенсивности потока и пропускной способности сервера (перекрестка), а также от вероятностных характеристик потока. Одним из очевидных способов уменьшения очереди является увеличение пропускной способности сервера. Но увеличение пропускной способности перекрестка в одном направлении (увеличение длительности "зеленого" цвета) приведет к снижению пропускной способности в поперечном направлении.

Поэтому для нахождения оптимального решения задачи требуется мониторинг реальных очередей в различных точках города и соответствующее динамическое регулирование работы светофоров. Т.е. результаты решения задачи также требуются в различных точках распределенной системы.

Постановка задачи:

Разработать централизованный алгоритм балансировки. Решение о переносе объекта с одного вычислительного узла распределенной системы на другой выполняется одним из процессов, который предварительно получает сообщения от всех вычислительных узлов об их загрузке. Сеть имеет произвольную топологию и является ориентированной. Схема сети прилагается.

Рекомендации по выполнению

Предположим, что распределенное приложение представляет собой взаимодействующие процессы, располагающиеся на различных вычислительных узлах. Поскольку алгоритм централизованный, то будем считать, что существует сервер, которые рассылает сообщение всем другим компьютерам в сети, что они должны сообщить о своей загрузке. Затем, получив от всех компьютеров нужную информацию, сервер принимает решение о балансировке.

При выполнении работы использовать программные средства технологии .Net.

Централизованный алгоритм балансировки приведен в первой лабораторной работе

Для рассылки сообщений в сети следует реализовать волновой алгоритм "Эхо".

Алгортм "Эхо"

Алгоритм "Эхо" может работать для любой топологии распределенной системы, в. нем имеется один инициатор.

Алгоритм использует метод прохода по графу, называемый "поиск (или просмотр графа) в ширину", описанный, в частности, в учебнике

Л.Н.Королева, А.И.Микова "Информатика. Введение в компьютерные науки" для поиска множества R достижимых вершин из данной вершины start.

Метод состоит в том, чтобы продвигаться от начальной вершины по ширине всего фронта, включив сначала в множество R все вершины, смежные с вершиной start, затем смежные со смежными и так далее. В описанном ниже методе Expand(R) расширения множества R множество Out(R) – это множество всех вершин, смежных с вершинами из R, так сказать, достижимых за один шаг. В частности, может оказаться, что все они уже принадлежат R и тогда дальнейшее расширение невозможно. На этом процесс прекращается.

```
Expand(R):
  begin
    if Out(R)  R then
      begin Out(R) => R;
        Expand(R)
      end
  end;
```

Первый вызов – Expand(Out(start)) ; Предполагается, что O $ut(\varnothing) = \varnothing, \varnothing \subseteq \varnothing$.

В алгоритме "Эхо" инициатор посылает маркеры всем своим соседям. Любой сайт s (не инициатор), получивший первый раз маркер от одного из своих соседей (обозначим этого соседа pre), рассылает маркеры всем соседним сайтам, кроме того, от которого получил маркер. Соседи поступают точно так же. Волна удаляется от инициатора.

Дальнейший процесс рассмотрим на примере дерева. Волна доходит до некоторых из висячих вершин. Висячей вершине отправлять маркер дальше некуда. Тогда она возвращает его той вершине, от которой получила (вот оно "эхо"). Вершины, получившие "эхо" от своих соседей, возвращают маркеры своим вершинам pre. Те, в свою очередь, генерируют "эхо" своим предшественникам. Наконец "эхо" доходит до инициатора. Инициатор, получив "эхо" от всех своих соседей, выполняет процедуру return(OK).

Ниже приведен распределенный алгоритм, состоящий из описаний процесса вычислений для сайта – инициатора и описаний процессов для сайтов – не-инициаторов. В тексте описания процесса тот сайт, на котором этот процесс выполняется (свой сайт), обозначен идентификатором this (в традициях языка Симула-67). Множество Out(this) – это множество сайтов, смежных по выходу с сайтом this, т.е. тех сайтов, на которые с сайта this можно отправить сообщения по однонаправленным или двунаправленным каналам связи. Функция card( ) задает число кардинальности (мощность) множества, являющегося аргументом этой функции. Переменные, встречающиеся в текстах процессов – локальные (по экземпляру для каждого процесса): обмен информацией между процессами происходит только

посредством сообщений, разделяемые переменные отсутствуют. Начальные значения счетчиков counter равны 0.

```
begin for u in Out(this) do out token to u ;
  while counter < card(Out(this)) do
    begin receive token; counter := counter + 1 end ;
  return(OK)
end ;
```

2.1. Процесс для инициатора:

```
begin receive token from u ; pre(this) := u ; counter := counter + 1 ;
  for (u in Out(this))&(u ≠ pre(this)) do out token to u ;
  while counter < card(Out(this)) do
    begin receive token; counter := counter + 1 end ;
  out token to pre(this)
end
```

2.2. Процессы для не-инициаторов:

Отчетность

В результате выполнения лабораторной работы должны быть представлены следующие материалы:

1. Программа;
2. Исходные тексты;
3. Презентация работы;

6.Заключение

Распределенные системы согласования начинают играть важную роль в разработки распределенных приложений. Большинство подобных систем отличаются отсутствием ссылочной связности процессов в том смысле, что процессы для связи между собой не нуждаются в явных ссылках друг на друга. Кроме того, может обеспечиваться и отсутствие временной связности, в этом случае для взаимодействия процессов друг с другом им вовсе не обязательно выполняться одновременно. Одна из важных групп систем согласования — это системы, основанные на парадигме издателя/подписчика, такие как TIB/Rendezvous. В этой модели сообщения доставляются получателям не в соответствии с их адресами, а в соответствии с темой сообщений. Процессы, желающие получать сообщения, должны подписаться на определенную тему. За выбор пути сообщений от издателя к подписчику отвечает промежуточный уровень. Другая группа систем

согласования использует модель генеративной связи, впервые предложенную в системе Linda. Генеративная связь реализуется разделяемыми пространствами кортежей. Кортеж — это типовая структура данных, сходная с записью. Чтобы извлечь из пространства кортежей нужный кортеж, процесс отыскивает его при помощи эталонного кортежа. Кортеж, соответствующий эталону, выбирается и возвращается запросившему его процессу. Если совпадений нет, процесс блокируется. Системы согласования отличаются от большинства других распределенных систем тем, что в них в основном решается задача предоставления удобного способа связи между процессами, которые ничего не знают друг о друге заранее. Связь может и далее осуществляться с сохранением анонимности. Основное преимущество подобного подхода — его гибкость. Можно дополнять и изменять продолжающую работать систему. Принципы распределенных систем, применимы и к системам согласования, хотя кэширование играют в современных их реализациях не столь важную роль. Кроме того, именование в них сильно связано с поиском по атрибутам. Аналогичный подход реализован и в службах каталогов.

7.Вопросы и задания

1. К какому типу модели согласования вы отнесете системы очередей сообщений.
2. Отсутствие ссылочной связности в TIB/Rendezvous компенсируется адресацией по теме. Какими еще средствами компенсируется отсутствие рхе ссылочной связности в этой системе?
3. Опишите реализацию системы публикации/подписки на основе системы очередей сообщений, такой как MQSeries компании IBM.
4. Во что на самом деле разрешается имя темы в TIB/Rendezvous и как протекает это разрешение?
5. Опишите простую реализацию полностью упорядоченной доставки сообщений в системе TIB/Rendezvous.
6. Когда в ходе транзакции TIB/Rendezvous сообщение доставляется процессу P, процесс подтверждает доставку. Можно ли на уровне транзакций сделать так, чтобы подтверждение отсылалось демону транзакций автоматически?
7. Предположим, что процесс в системе TIB/Rendezvous реплицируется. Приведите два решения, позволяющие исключить случаи публикации сообщений реплицированными процессами более одного раза.

8. Насколько необходима полностью упорядоченная групповая рассылка при репликации процессов в системе TIB/Rendezvous?

9. Опишите простую схему для PGM, которая позволяла бы получателям обнаруживать пропавшие сообщения, даже если пропадет первое сообщение в последовательности.

10. Могут ли журналы регистрации в TIB/Rendezvous, реализованные в виде файлов, гарантировать, что сообщение, даже при наличии отказавших процессов,не будет потеряно?

И. Как следовало бы реализовать в TIB/Rendezvous модель согласования на основе генеративной связи?

12. Не считая отсутствия временной связности в Jini, в чем, по вашему мнению,состоит наиболее существенная разница между Jini и TIB/Rendezvous с точки зрения моделей согласования?

13. Срок аренды в Jini всегда определяется его продолжительностью и никогда —в виде абсолютного значения времени истечения аренды. Почему?

14. В чем состоят наиболее серьезные проблемы масштабирования в Jini?

15. Пусть в распределенной реализации JavaSpace кортежи реплицируются на нескольких машинах. Приведите протокол удаления кортежа, в котором при попытке двух процессов удалить один и тот же кортеж не возникает условий гонок.

16. Представьте себе, что транзакция Г в Jini требует блокировки объекта, который уже заблокирован другой транзакцией *T'*. Опишите, что произойдет.

17. Представьте себе, что клиент Jini кэширует полученный из JavaSpace кортеж, чтобы иметь возможность в следующий раз не обращаться к JavaSpace. Имеетли такое кэширование смысл?

18. Ответьте на предыдущий вопрос для случая, когда клиент кэширует результаты, полученные службой поиска.

19. Опишите простую реализацию отказоустойчивого пространства JavaSpace.

8.Рекомендуемая литература:

1. Ван Стеен М, Танненбаум Э. Распределенные системы. Принципы и парадигмы. СПб.:Питер, 2003.-877с.
2. Джексон П. Введение в экспертные системы. М.: Изд. дом "Вильяме", 2001.
3. Abdellah Bedrouni, Ranjeev Mittu, Abdeslem Boukhtouta, Jean Berger. Distributed Intelligent Systems: A Coordination Perspective. Springer Science & Business Media, 2009, P. 181.
4. Bond and L. Gasser. Readings in Distributed Artificial Intelligence. Morgan Kaufman, San Mateo, CA, 1988.
5. Bond and L. Gasser. Readings in Distributed Artificial Intelligence. Morgan Kaufman, San Mateo, CA, 1988.
6. Coulouris G., Dollimore J., Kindberg T Distributed Systems. Concepts and Design. Addison-Wesley Publishing Company, 1995, 644 p.
7. Yu-Kwong Kwok and Lap-Sun Cheung A new fuzzy-decision based load balancing system for distributed object computing Journal of Parallel and Distributed Computing, 64 (2004) 238-253.
8. H. Baala, J. Gaber, M. Bui and T. El-Ghazawi, O. Flauzac A self-stabilizing distributed algorithm for spanning tree construction in wireless ad hoc networks Journal of Parallel and Distributed Computing, 63 (2003) 97-104.