Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Факультет информационной безопасности

Кафедра инфокоммуникационных технологий

В. А. Вишняков

ИНТЕЛЛЕКТУАЛЬНЫЕ ТЕХНОЛОГИИ В ИНФОКОММУНИКАЦИЯХ

Рекомендовано УМО по образованию
в области информатики и радиоэлектроники
в качестве учебно-методического пособия
для специальности 1-45 80 01
«Системы и сети инфокоммуникаций»

Минск БГУИР 2024

УДК [004+654.1](075)
ББК (32.973.202+32.88)я73
В55

Р е ц е н з е н т ы:

кафедра телекоммуникационных систем
учреждения образования «Белорусская государственная академия связи»
(протокол № 6 от 12.12.2022);

профессор кафедры робототехнических систем
Белорусского национального технического университета
доктор технических наук, профессор А. А. Лобатый

Вишняков, В. А.

В55 Интеллектуальные технологии в инфокоммуникациях : учеб.-метод. пособие / В. А. Вишняков. – Минск : БГУИР, 2024. − 266 с. : ил.

ISBN 978-985-543-743-8.

Включает теоретический и практический материал по учебной дисциплине «Теория системного анализа и принятия решений в инфокоммуникациях» специальности «Системы и сети инфокоммуникаций».

Включает базовые сведения по основам искусственного интеллекта и использованию интеллектуальных технологий для применения в инфокоммуникационных системах. Рассматриваются основы ИИ: представление и использование знаний, языков программирования, экспертных систем, нейронных и мультиагентных систем. Даны основы новых направлений – интеллектуализации ИК-бизнеса, принятия решений в ИК-системах, построения семантического веб-пространства, использования веб-сервисов, интеллектуализации принятия решений в ИКС, основы облачных и туманных технологий. Приведены примеры использования интеллектуальных систем в различных областях инфокоммуникаций.

УДК [004+654.1](075)
ББК (32.973.202+32.88)я73

ISBN 978-985-543-743-8

СОДЕРЖАНИЕ

Глоссарий основных терминов

БЗ – база знаний

БП – бизнес-процессы

ИА – интеллектуальный агент

ИАД – интеллектуальный анализ данных (*Data Mining)*

ИИ – искусственный интеллект

ИИС – интеллектуальные информационные системы

ИК – инфокоммуникации

ИОС – интеллектуальная обучающая система

ЛПП – логика первого порядка

МП – машинный перевод

НС – нейронные сети

ОО модель – объектно-ориентированная модель

ОСТИС – открытые семантические технологии интеллектуальных систем

ПЗ – представление знаний

ПО – программное обеспечение

ППР – поддержка принятия решений

ССП – сбалансированная система показателей

СУБЗ – система управления базой знаний

ФА – факторный анализ

ЭС – экспертная система

AI (artificial intelligence) – искусственный интеллект

ASM (Automatic Storage Manager) – автоматическое управление памятью

API (Application Program Interface) – прикладной программный интерфейс

BEEP (Blocks Extensible Exchange Protocol) – расширяемый протокол обмена блоками

BI (Business intelligence) – интеллектуализация бизнеса

BOD (Behavior-Oriented Design) – технология инженерии ПО

DAML-S (DAML-based Web Service Ontology) – онтология на базе структурированного языка DAML для описания веб-сервисов

DataWarehouse – хранилище данных

DL (Definition Logic) – логика описаний

DLG (Direct Label Graph) – прямой помеченный граф

DOM (Document Object Model) – объектная модель документа

DTD (Document Type Definition) – определение типа XML-документа

EbXML (e-business XML) – инфраструктура электронного бизнеса с использованием XML

ETL (Extract Transform Load) – извлечение, преобразование и загрузка

GPS (General Program Solving) − универсальный решатель

HTTPS (secure HTTP) – секретный протокол HTTP

ICA (Indexing Complex Analysis) – индексный комплексный анализ

ICAM (Integrated Computer Aided – Manufacturing) – интегрированная компьютеризация производства

KIF (Knowledge Interchange Format) – формат для обмена знаниями

KQML (Knowledge Query and Manipulation Language) – язык запросов и манипулирования знаниями

LSI (Latent Semantic Indexing) – скрытая семантическая индексация

MOLAP – многомерный OLAP

NLP ((Natural Language Processing) – языковый процессор

OIL (Ontology Interchange Language) – язык для обмена онтологиями

OLAP-системы (On-Line Analytical Processing) – онлайновая аналитическая обработка

OLTP-системы (On-Line Transaction Processing) – онлайновая обработка транзакций

OWL (Ontology Web Language) – язык для описания онтологий

PCA (Permanent Complex Analysis) – постоянный комплексный анализ

RDF (Resource Description Framework) – язык описания ресурсов в Semantic Web

PICS (Platform Internet Content Search) – платформа интернетного контентного выбора

RDA (remote data application) – доступ к удаленным данным

ROLAP – реляционный OLAP

RSS (RDF Site Summary) – сайт для автоматического обмена ссылками

RWC (Real World Computing) – вычисления в реальном мире

SAML (Security Authorization Markup Language) – секретный аутентифицированный язык разметки

SADT (Structured Analysis & Design Technique − метод структурного анализа и проектирования

SOAP (Simple Object Access Protocol) – транспортный протокол на базе HTTP и XML

SSL (Secure Socket Layer) – секретный уровень сокетов

SVD (Singular Vector Derivation) – сингулярное разложение значения

SWAP (Semantic Web and Peer-To-Peer) – проект сочетания технологий P2P и Semantic Web

SWWS (Semantic Web and Web-service) – проект сочетания технологий веб-сервисов и Semantic Web

TDWI (The Data Warehousing Institute) – институт хранилищ данных

UDDI (Universal Description, Discovery and Integration) – универсальные методы описания, отыскания и интеграции веб-сервисов

XKMS (XML Key Management Specification) – ключевая спецификация управления на базе XML

XML (Extensible Markup Language) – расширяемый язык разметки

XSL (Extensible Style Language) – расширяемый язык стилей

WI (Web intelligence) – интеллектуализация веб-пространства

WSDL (Web Services Description Language) – язык описания веб-сервисов

WWW (World Wide Web) – средство разметки информации в Интернете

ВВЕДЕНИЕ

Общая тенденция развития искусственного интеллекта показывает, что интеллектуальная обработка получает все большее развитие и распространение в различных областях техники и управления. Этому способствует развитие экономики и менеджмента знаний, интеллектуализация бизнеса, исследования в области построения семантического пространства веб, которые приводят к ускорению процессов интеллектуализации в различных областях техники.

Растут объемы и быстродействие накопителей информации. Все большей становится доля оптоволоконных каналов связи, которые позволяют передавать огромные массивы данных со скоростью 100–1000 Гбит/с и более. Все это ведет к увеличению удаленных рабочих групп (офисов). Эти факторы и способствуют развитию методов и средств интеграции данных и приложений, базирующихся на методах и технологиях ИИ. Особое значение технологии и системы ИИ приобретают в сетях инфокоммуникаций. На смену традиционным производствам приходят интеллектуальные.

Используя учебно-методическое пособие, студенты и магистранты ознакомятся с сущностью направлений в области искусственного интеллекта (глава 1), получат понятия и навыки по моделям представления знаний (глава 2), ознакомятся с основными элементами языков и инструментальных средств интеллектуальных систем (глава 3). Изучат вопросы создания и использования экспертных систем (глава 4), познакомятся с основами нейронных сетей (глава 5) и мультиагентных систем (глава 6).

Особое внимание уделено системам интеллектуализации в ИК-компаниях и системах (главы 7–10). Понятия и детализация интеллектуализации ИК-бизнеса приведены в главе 7. Обсуждаются различные интеллектуальные технологии для решения задач в системах и сетях. Рассмотрено использование ИС в области машинного перевода, автоматического решения задач, поисковых системах, распознавании образов (глава 8). Приведены основы интеллектуализации для приятия решений в инфокоммуникациях (глава 9). В качестве перспективных направлений рассматриваются интеллектуальные средства и системы Интернета (глава 10) и использование облачных и туманных технологий для интеллектуализации управления (глава 11).

Учебно-методическое пособие написано на основании результатов научной работы автора в университетах Германии и Польши, исследований, проводимых с аспирантами и магистрантами, научной работы над проектами в рамках программы «Информатизация» в 2010–2022 гг., а также чтения лекций в 2018–2023 гг. на кафедре ИКТ БГУИР. Естественно, изложить в одном учебно-методическом пособии все аспекты создания интеллектуальных бизнес-приложений не представляется возможным, поэтому автор отсылает читателей к литературе, приведенной в конце каждой главы.

ЧАСТЬ 1
ОСНОВЫ ИНТЕЛЛЕКТУАЛЬНЫХ ТЕХНОЛОГИЙ

1 ОСНОВЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

1.1 Основные понятия искусственного интеллекта

Искусственный интеллект (ИИ) – это направление науки, в рамках которого создаются и решаются задачи математического, алгоритмического и программного моделирования тех видов человеческой деятельности, которые считаются творческими.

Системы познания – это аппаратно-программные системы, которые могут получать, хранить и использовать знания, такие компьютерные системы называются системами искусственного интеллекта (ИИ, анг. artificial intelligence). Считается, что системы искусственного интеллекта должны быть неотличимы от людей (пройти тест Тьюринга: поведение объекта с ИИ не отличается от поведения человека при ответе на ряд вопросов). Эксперты полагают, что системы искусственного интеллекта должны субоптимально использовать любые методы работы со знаниями для решения трудноформализуемых задач («найди то, не знаю что») [1; 2].

Современные интеллектуальные информационные технологии для обработки информации и решения задач с помощью компьютерных технологий основаны на достижениях в области искусственного интеллекта. Результаты исследований по ИИ используются в интеллектуальных системах (ИС) – технических и/или программных системах, способных решать задачи, считающиеся творческими, относящиеся к определенной предметной области, знания о которых хранятся в памяти этих систем. Системы искусственного интеллекта состоят из трех основных блоков: базы знаний, решателя и интеллектуального интерфейса. Типичным представителем систем искусственного интеллекта являются экспертные системы (глава 4), нейронные сети (глава 5).

Системы искусственного интеллекта ориентированы на решение специального класса задач, называемых неформализуемыми, или трудноформализуемыми, которые имеют одну или несколько особенностей: алгоритмическое решение неизвестно (хотя, возможно, оно существует); задача не может быть определена в числовой форме (символическое представление является обязательным представлением); цели решения задачи не могут быть выражены в терминах определенной целевой функции; имеется большая размерность пространства решений; присутствуют динамически изменяющиеся данные и знания. Трудноформализуемые задачи отличаются неполнотой, двусмысленностью и/или противоречивостью исходных данных и знаний о предметной области. В исследованиях по ИИ можно выделить два основных направления [1; 4; 5; 8]:

– программно-прагматическое (software-pragmatic) – связано с разработкой интеллектуальных программ, с помощью которых можно решать те задачи, решение которых считалось исключительно прерогативой творческих людей

(изобретательство, музыка, живопись и т. д.). Направление включает в себя разработку экспертных систем, программы для решения логических задач, поиска, идентификации, классификации и т. д. Это направление ориентировано на поиск алгоритмы решения интеллектуальных задач или логической обработки информации на несуществующих компьютерных моделях. В рамках этого направления проводится поиск подходов, моделей и алгоритмов, характерных для человеческого мышления;

– бионическое (bionic) – связано с проблемами искусственного воспроизведения методик и процессов, которые характерны для человеческого мозга и лежат в основе процесса решения проблем жизни и деятельности человеческого общества. В рамках этого подхода сформировалось новое направление, наука нейроинформатика, практическим выходом которой стала разработка нейросетевых программных пакетов, СБИС и аппаратных нейрокомпьютеров.

Эксперты по искусственному интеллекту предложили следующие модели [1; 2; 4; 5]:

– поиск наилучших путей на графе. Этот подход представляет собой решение задачи с использованием некоторого трафика, отражающего пространство состояний, и в этом лабиринте осуществляется поиск оптимального пути от входных данных к результирующим;

– эвристическое программирование, в котором эвристика – это набор правил, которые математически не обоснованы, но позволяют сократить количество шагов, этапов, итераций в пространстве поиска;

– логическое программирование с использованием методов математической логики. Для логики первого порядка получен новый результат – метод резолюций, который позволяет автоматически доказывать теоремы при наличии исходных аксиом. В 1973 году на его основе разработан язык логического программирования – Prolog.

Технологии ИИ нашли применение во многих областях техники, среди них выделим [3]:

– обработка естественного языка, создание языковых процессоров (Natural Language Processing – NLP);

– представление знаний и формирование логических выводов;

– машинное обучение;

– экспертные системы (ЭС);

– нейронные сети;

– решение проблем и логическое программирование;

– программирование игр;

– распознавание образов и машинное зрение;

– инструменты робототехники.

1.2 Развитие искусственного интеллекта

Искусственный интеллект как наука сформировался на основе областей знаний: логика, теория вероятностей, вычислительная техника, принятие решений,

неврология, орнитология, лингвистика и компьютерная инженерия. Началом создания искусственного интеллекта считается 1956 год (семинар в Дартмуте (США)) [2]. В 50-х годах исследователи искусственного интеллекта пытались создать элементы интеллектуальных машин, имитируя функции мозга с помощью простейших нейронных сетей (персептронов). Эти попытки оказались безуспешными из-за непригодности на тот момент как аппаратных средств (ламповые элементы ЭВМ), так и программных средств (отсутствие ОС, простейших средств автоматизации программирования). Появляется язык Lisp и его диалекты (глава 3) [3; 9].

В 60-х годах были предприняты попытки найти представление знаний об общих методах решения широкого класса задач путем моделирования сложного процесса мышления с использованием моделей знаний (глава 2). Разработка универсальных программ оказалась слишком сложной, тем не менее была создана программа universal solver (общее решение программ – GPS), которая моделировала процесс решения задач человеком. Компания IBM создала программу для доказательства геометрических теорем. Математик Д. Робинсон открыла метод резолюций.

В 70-х годах была принята новая концепция, которая заключается в том, что для создания интеллектуальной программы она должна быть снабжена большим количеством знаний о предметной области. Появляется проект эвристического программирования (на основе опыта). Развитие этого направления привело к созданию экспертных систем (глава 4). Одним из их первых результатов была ЭС Dendral, работавшая в области химического синтеза. Значительный прорыв в практических приложениях искусственного интеллекта произошел в середине 70-х годов, когда на смену поиску универсального алгоритма мышления пришла идея моделирования специфических знаний опытных специалистов. В США появились первые коммерческие системы, основанные на знаниях – ЭС: Mysin − для диагностики инфекционных заболеваний крови; R1 − для проектирования конфигурации компьютера и т. д.

В 80-х годах в рамках новой технологии появились первые коммерческие программные продукты. В это время начала развиваться область машинного обучения. В передовых странах (Япония, ЕС, США, СССР) были запущены крупнейшие национальные и международные исследовательские проекты в истории обработки данных, направленные на создание «интеллектуальных компьютеров пятого поколения». В целом искусственный интеллект пережил быстрый рост с инвестициями от нескольких миллионов долларов в 1980 году до нескольких миллиардов долларов в 1989 году. Однако затем начался период «зимы искусственного интеллекта», когда многие компании не смогли реализовать свои интеллектуальные проекты.

В 90-е годы успешно развивались области, связанные с машинным переводом, распознаванием речи (глава 8) и нейроинфоматикой, основанной на разработке нейронных сетей (глава 5). Доминируют разработки, основанные на скрытых марковских моделях, основанных на строгой теории и использующих большие наборы данных. Сформировался новый подход к решению интеллектуальных задач – представление и использование знаний. Этот подход продолжает развиваться, в 90-е годы сформировалось направление – управление знаниями.

В начале XXI века начинается развитие интеллектуализации бизнеса (глава 7), интеллектуализации средств в Интернете (поиск информации, интеллектуальные агенты, главы 6, 9, 10), а также технологий анализа скрытых законов в данных (интеллектуальный анализ данных). Направление Big Data зародилось в 2000–2010 годах, а основные результаты получены в 2010–2023 годах. Развивается машинное обучение на базе сверточных (CNN) и рекуррентных (RNN) нейронных сетей с применением распознавания образов, обработки естественного языка и предсказательного анализа. Архитектура трансформеров, представленная в 2017 году, привнесла изменения в область обработки естественного языка и генерации текста, что и стало прообразом ChatGPT [11].

В связи с возникновением и развитием Интернета можно выделить такие области, как представление знаний в Интернете, извлечение знаний из информационных ресурсов Сети, интеллектуальный поиск релевантной информации, интеграция информационных ресурсов и приложений, управление знаниями, построение глобальных систем знаний в Интернете и их обработка (семантическая паутина), создание и использование веб-сервисов [3; 9].

1.3 Направления искусственного интеллекта

Искусственный интеллект – одно из направлений современной ИТ-науки, целью которого является разработка математических и аппаратно-программных средств, позволяющих пользователю, не являющемуся программистом, ставить и решать свои задачи, традиционно считающиеся творческими (интеллектуальными), общаясь с компьютером на ограниченном подмножестве естественного языка. Рассмотрим эти направления [1; 4; 5].

Представление знаний и разработка систем, основанных на знаниях. Это основное направление искусственного интеллекта. Это связано с разработкой моделей представления знаний, созданием баз знаний, которые формируют ядро экспертных систем (ЭС). Она включает в себя модели и методы привлечения и структурирования знаний и сливается с инженерией знаний.

Игры и творчество. Традиционно искусственный интеллект включает в себя интеллектуальные игровые задания – шахматы, шашки, карты. Они основаны на одном из ранних подходов – модели лабиринта, поиске решений плюс эвристика. Это скорее коммерческое направление, поскольку с научной точки зрения эти идеи считаются не очень перспективными [5]. Однако программа IBM Deep Blue победила чемпиона мира по шахматам Г. Каспарова.

Понимание естественного языка и решение проблем. Программа решает кроссворды лучше, чем большинство людей. Он использует базу данных кроссвордов, а также множество источников информации, включая словари и оперативные базы данных.

Новые компьютерные архитектуры. Это направление занимается разработкой новых аппаратных решений и архитектур, направленных на обработку символьных и логических данных. Создаются машины Prolog и Lisp, нейроком-

пьютеры. Последние разработки посвящены компьютерам с базами данных, знаниям и специализированным параллельным компьютерам.

Интеллектуальные роботы. Роботы – это электромеханические устройства, предназначенные для автоматизации человеческого труда. Идея создания роботов исключительно стара. С момента его создания сменилось несколько поколений роботов. Роботы с жесткой схемой управления относятся к первому поколению. Почти все современные промышленные роботы принадлежат к этому поколению. По сути, это программируемые манипуляторы. Адаптивные роботы с сенсорными устройствами. Есть образцы таких роботов, но они все еще используются в качестве образцов в промышленности. Самоорганизующиеся, или интеллектуальные, роботы. Это конечная цель развития робототехники. Главной проблемой при создании интеллектуальных роботов является проблема машинного зрения. В последнее время в мире производится более 180 тысяч роботов в год. Многие хирурги используют роботов-ассистентов в микрохирургии. Последние используют системы машинного зрения для создания трехмерной модели внутренних органов человека [5].

Специальное программное обеспечение. В рамках этого направления разрабатываются специальные языки для решения невычислительных задач. Эти языки были ориентированы на обработку символьной информации – LISP, PROLOG, SMALLTALK, REFAL и т. д. Кроме того, создаются пакеты прикладных программ, ориентированных на промышленную разработку интеллектуальных систем, или программных средств искусственного интеллекта, например, KEE, ARTS. Довольно популярно создавать так называемые пустые экспертные системы, или «оболочки» – BXSYS, Ml и т. д., в которых можно наполнять базы знаний информацией, создавая различные прикладные системы.

Обучение и самообучение. Активно развивающаяся область искусственного интеллекта. Она включает в себя модели, методы и алгоритмы, ориентированные на автоматическое накопление знаний на основе анализа и обобщения данных. Она включает в себя обучение на основе приложений (или индуктивное), а также традиционные подходы к распознаванию изображений [5].

Автономное планирование и составление расписаний. Работая на расстоянии сотен миллионов километров от Земли, станция удаленного агента обеспечивает график исследований для космического аппарата.

Автономное управление. Системы компьютерного зрения используются для управления движущимися объектами. Например, программа Alvinn используется для вождения автомобиля в Соединенных Штатах на протяжении 2850 миль, и человек участвовал в 2 % поездок на сложных выездах [5].

В связи с возникновением и развитием Интернета можно выделить такие области, как представление знаний в Интернете, извлечение знаний из информационных ресурсов сети, интеллектуальный поиск релевантной (соответствующей заданным условиям) информации. Также интеграция информационных ресурсов и приложений, управление знаниями, построение глобальных систем знаний в Интернете и их обработка (семантическая паутина), создание и использование веб-сервисов. Основные из этих областей будут более подробно рассмотрены в главах 9, 10, 11.

1.4 Использование ИИ в управлении и технике

Исследования в области искусственного интеллекта часто классифицируются в зависимости от сферы его применения, а не на основе различных теорий и школ. Каждая из этих областей десятилетиями разрабатывала свои собственные методы программирования и формализмы; каждая из них имеет свои собственные традиции, которые могут заметно отличаться от традиций соседней области исследований. В технике результаты исследований искусственного интеллекта применяются в следующих областях: экспертные системы; системы баз знаний; интеллектуальное обучение; нейронные сети, робототехника [4; 5; 7].

Экспертные системы (ЭС) – это набор программ, который выполняет функции эксперта при решении задачи в области его компетенции, подобно эксперту-человеку, оперирует знаниями в ходе своей работы. Знания о предметной области, необходимые для функционирования ЭС, определенным образом формализуются и представлены в памяти компьютера в виде базы знаний, которая может быть изменена и дополнена в процессе разработки системы.

ЭС дают советы, проводят анализ, выполняют классификацию, дают рекомендации и ставят диагноз. Они сосредоточены на решении проблем, которые обычно требуют экспертного рассмотрения специалистом-человеком. В отличие от машинных программ, использующих процедурный анализ, ЭС решают проблемы в узкой предметной области (конкретной области знаний) на основе дедуктивных рассуждений. Главным преимуществом ЭС является способность накапливать знания, сохранять их в течение длительного времени, обновлять и тем самым обеспечивать относительную независимость конкретной организации от наличия в ней квалифицированных специалистов.

Системы с базами знаний (БЗ). За последние 10–15 лет ЭС были возрождены в виде систем баз знаний, которые были тесно интегрированы с существующими бизнес-системами. Они используются в технических областях, здравоохранении, страховании, банковском деле, для того чтобы накапливать опыт с помощью правил и объектов, повышать качество принимаемых решений. Базы знаний встроены в самые современные большие системы. Они входят в структуру многоагентных программ, которые осуществляют поиск в Интернете и помогают командам пользователей справляться с наплывом информации. Рассмотрим факторы, которые стимулировали разработку систем с БЗ:

– компании, которые добились значительной экономии денег благодаря технологии базы знаний, разрабатывают и встраивают ее в специальные бизнес-процессы, которые были бы просто невозможны без компьютерной экспертизы;

– разработаны новые технологии создания БЗ, что является необходимым инструментом, способным изменить бизнес-процесс;

– современные системы внедряются не на специализированном, а на стандартном оборудовании [6].

Объектная технология, на основе которой могут создаваться и развиваться современные системы с БЗ, является значительным шагом вперед по сравнению с CASE-инструментами, поскольку она схожа с человеческим восприятием

окружающей действительности. Идея моделирования меняется, то же самое происходит и с объектами, поэтому обслуживание программируемых объектов может выполняться аналогично адаптации наших умозрительных изображений к изменяющимся условиям окружающей среды. Эта технология подходит для аналитиков и программистов, потому что это очень похоже на стратегию решения проблем и соответствует мыслительным процессам людей, которые считаются экспертами в своей области [6].

Распознавание образов. Традиционное направление искусственного интеллекта, берущее начало у самых его истоков. Каждому объекту присваивается матрица распознавания, в соответствии с которой он распознается. Это направление близко к машинному обучению, тесно связанному с нейрокибернетикой. Это также научно-техническое направление, связанное с разработкой методов и построением систем на базе специализированных компьютеров для установления принадлежности некоторого объекта (объекта, процесса, явления, ситуации, сигнала) к одному из заранее выделенных классов объектов (изображения) [5].

Интеллектуальное обучение. Будем называть интеллектуальной системой обучения (ИСО) программную систему, которая реализует определенную технологию обучения, основанную на знаниях экспертов в определенной предметной области (ПрО). Выделим основные задачи, которые возникают при создании такой системы: разработка методов мониторинга и диагностики ошибок учащихся; разработка методов управления обучением; представление и обработка знаний в предметной области, составляющей предмет обучения. Архитектура ИСО основана на следующей модели процесса обучения. Существует цель обучения, выраженная в терминах текущих характеристик учащегося. До тех пор, пока цель не будет достигнута, повторяется следующая последовательность действий: на основе текущего состояния студента и методики преподавания генерируется следующая задача; ответ учащегося сравнивается с эталонным решением и на основе различий диагностируются ошибки учащегося; по результатам диагностики корректируются текущие характеристики обучения.

По этой модели процесса обучения ИСО можно рассматривать как набор из трех взаимодействующих экспертных систем: ЭС для решения проблем (ЭС РП) в изучаемой предметной области; ЭС для диагностики ошибок учащихся (ЭСДО); ЭС для планирования процесса управления обучением (ЭСУО).

Экспертная система для решения проблем предназначена для разработки процесса решения. Этот компонент может представлять собой комбинацию ЭС, построенной на знаниях эксперта в решении проблем, решателя проблем; поиск или выбор решения проблемы из набора решений.

ЭС диагностики ошибок предназначена для выявления его неправильных представлений об изучаемой области на основе сравнения его ответа с эталонным. ЭС для управления процессом обучения – это система планирования в рамках ограничений, налагаемых доступным учебным материалом. Это формализует знания опытного методиста о методах обучения.

Взаимодействие ИСО с учащимся происходит следующим образом. В соответствии с текущей целью ЭС образовательного учреждения формирует сле-

дующее задание для учащегося, которое одновременно передается в ЭСУО. Затем ЭСДО сравнивает решение учащегося с решением, полученным ЭСРП, и на основе различий пытается установить, какие неправильные представления учащегося о ПрО могли привести к расхождениям. В результате диагностики представление ИОС учащегося, отраженное в модели учащегося (МО), изменяется, и руководство снова получает ЭСУО, который уточняет текущую цель и формирует новую задачу. Взаимодействие с учащимся происходит через интерфейс, который может содержать текстовый, графический или речевой ввод/вывод, лингвистический процессор и т. д.

При создании образовательных инструментов большое внимание уделяется созданию удобных средств взаимодействия учащихся с системой, в том числе средств визуализации изучаемых объектов и процессов. Принято объединять эти инструменты под названием интерфейса ученика (learner). В структуре ИМО можно выделить следующие базы знаний (БЗ): образовательная БЗ для данного программного обеспечения; модель обучающегося; БЗ о возможных ошибках обучаемого; БЗ о процессе обучения. Образовательная база знаний (ОБЗ) описывает не только основные понятия и методы решения задач, используя программное обеспечение, но также содержит определения понятий, описания методов, примеры, упражнения и задачи. Она также содержит необходимые данные для визуализации изучаемых объектов и процессов. В отличие от базы знаний ЭС для решения технических задач УБЗ должна по возможности отражать стратегические знания о методах решения задачи и соответствующую структуру программного обеспечения. С другой стороны, ОБЗ можно рассматривать как представление ограничений, в рамках которых ЭСУО планирует обучение.

База знаний об ошибках учащегося (БЗО) содержит каталог возможных ошибок учащегося и правила для выдвижения и проверки гипотез о неправильных представлениях, которые привели к этой ошибке, на основе различий между решениями, предложенными учащимся, и ЭС, а также текущим состоянием МО.

Модель учащегося содержит информацию о состоянии знаний учащегося: общие, интегрированные характеристики и те, которые отражают усвоение текущего учебного материала. Первоначально МО формируется в ходе предварительного тестирования обучаемого. В терминах МО выражается цель обучения.

База знаний по процессу обучения (БЗПрО) содержит знания о планировании и организации процесса обучения, общих и частных методах преподавания. ОБЗ обычно содержит знания для визуализации объектов и моделей, используемых в этой ИСО.

Выводы

По приведенным сведениям можно сформулировать заключение, используя материалы источников [1−11].

1. Искусственный интеллект – одно из направлений информационных трендов, в рамках которого ставятся и решаются задачи аппаратного и программного моделирования тех видов человеческой деятельности, которые тра-

диционно считаются интеллектуальными. Интеллектуальные системы (ИС) – это технические или программные системы, способные решать задачи, которые считаются творческими. Они состоят из трех блоков: базы знаний, решателя и интеллектуального интерфейса.

2. В исследованиях ИИ можно выделить два основных направления: программно-прагматическое − занимается созданием программ, которые могут быть использованы для решения интеллектуальных задач: распознавания, игр, логических ситуаций, поиска, классификации и т. д. на существующих компьютерах; бионическое − занимается проблемами искусственного воспроизведения тех структур и процессов, которые характерны для человеческого мозга, в которых сформировалась наука нейроинформатика, практическим результатом которой стала разработка нейронных систем ChatGPT.

3. Эксперты в области искусственного интеллекта предложили свои собственные модели: модель поиска в лабиринте, представляющая проблему в виде графа, в котором осуществляется поиск оптимального пути от входящих данных к результирующим данным; эвристическое программирование, которое сокращает количество итераций в пространстве поиска; логическое программирование, используя методы математической логики, основанные на методе разрешения, был создан язык Prolog.

4. Основные задачи, которые возникают при создании интеллектуальной системы обучения: разработка методов мониторинга и диагностики ошибок учащихся; разработка методов управления обучением; представление и обработка знаний в предметной области, составляющей предмет обучения.

5. Результаты исследований ИИ в управлении используются в следующих областях: экспертные системы (ЭС); системы баз знаний; интеллектуальное обучение; нейронные сети, интеллектуализация бизнеса, управление знаниями, диагностика и управление в технических системах, использование исследований в области семантической паутины, веб-сервисов и интеллектуальных агентов.

Вопросы для контроля

1. Дайте понятие искусственного интеллекта.
2. Дайте понятие интеллектуальных информационных технологий.
3. Охарактеризуйте программно-прагматическое направление ИИ.
4. Охарактеризуйте бионическое направление ИИ.
5. Поясните эвристическое и логическое программирование.
6. Охарактеризуйте основные результаты развития ИИ.
7. Поясните направления развития ИИ.
8. Определите направления ИИ в технике.
9. Охарактеризуйте направления ИИ в управлении.

Литература, используемая в главе

1. Искусственный интеллект : справ. В 3 т. Т. 2. : Модели и методы / под ред. Д. А. Поспелова. – М. : Радио и связь, 1990. – 304 с. ; Т. 3. : Программные и аппаратные средства / под. ред. Д. А. Поспелова. – М. : Радио и связь, 1990. – 363 с.

2. Рассел, С. Искусственный интеллект, современный подход / С. Рассел, П. Норвинг. – М. : Вильямс, 2006. – 1408 с.

3. Вишняков, В. А. Интеллектуальные системы в управлении / В. А. Вишняков. – Минск : Изд-во МИУ, 2010. – 364 с.

4. Масленникова, О. Е. Основы искусственного интеллекта : учеб. пособие / О. Е. Масленникова, И. В. Попова. – Магнитогорск : МаГУ, 2008. – 282 с.

5. Информатика : учебник / под ред. проф. Н. В. Макаровой. – 3-е изд., перераб. – М. : Финансы и статистика, 2009. – 768 с.

6. Сальников, Е. А. Методы формализации и автоматизации маркетинговых задач конкурентного анализа : дис. канд. эконом. наук : 08.00.05, 08.00.13 / Е. А. Сальников. – СПб., 2005. – 248 л.

7. ДП 150906 ФАИТ Кадыров ША.txt [Электронный ресурс]. – Режим доступа: https://studizba.com/files/show/pdf/69251-1-vyvod-otcheta-na-pechat-prosmotr-po.html. – Дата доступа: 15.12.2021.

8. Саак, А. Е. Информационные технологии управления : учебник для вузов /А. Э. Саак, Е. В. Пахомов, В. Н. Тюшняков. – СПб. : Питер, 2011. – 320 с.

9. Развитие и перспективы интеллектуальных технологий управления [Электронный ресурс]. – Режим доступа: http://elibrary.ru/item.asp?id=36366046. – Дата доступа: 15.12.2021.

10. Самое интересное для ИИ начинается с 2010 года [Электронный ресурс]. – Режим доступа: https://vc.ru/future/964901-samoe-interesno-dlya-ii-nachinaetsya-s-2010-goda. – Дата доступа: 12.05.2024.

2 ОСНОВЫ ПРЕДСТАВЛЕНИЯ И ИСПОЛЬЗОВАНИЯ ЗНАНИЙ

2.1 Данные и знания

Информация, которая представляется и обрабатывается в компьютере, разделяется на *процедурную и декларативную*. Процедурная информация реализована в *программах*, которые выполняются в процессе решения задач, декларативная информация – в *данных*, с которыми эти программы работают [1; 7].

С развитием структуры компьютеров происходило изменение информационных структур представления данных. Появились разные представления данных, такие как векторы, матрицы, списочные и иерархические структуры. Поэтому в языках программирования высокого уровня описываются *абстрактные типы данных,* структура которых задается программистом [7].

В базах данных могут одновременно храниться огромные объемы информации, а средства, образующие систему управления базами данных (СУБД), позволяют удобно манипулировать с данными, извлекать их из базы данных и записывать их обратно. По мере развития исследований в области ИС возникла идея описания знаний, которое объединило в себе элементы процедурной и декларативной информации [14].

В машине знания, так же как и данные, отображаются в знаковой форме – в виде текста, файлов, информационных массивов и т. д. Проще говоря, знания – это особым образом организованные данные. В таких системах ИИ как экспертные системы, знания являются основным объектом формирования, обработки и исследования. База знаний (БЗ) наравне с БД – необходимая составляющая системы ИИ. Машины, реализующие элементы ИИ, называются основанными на знаниях, а раздел теории ИИ, связанный с представлением знаний и экспертными системами, – инженерия знаний. Особенности знаний включают пять элементов [1; 3; 7; 14]:

1. *Внутренняя интерпретируемость.* Информационная единица должна иметь имя, по которому ИС находит ее, а также отвечает на запросы, в которых это имя встречается. Когда данные, хранящиеся в памяти, были без имен, отсутствовала возможность их идентификации системой. Данные могла идентифицировать лишь программа, извлекающая их из памяти по указанию программиста, написавшего программу. Что скрывается за тем или иным двоичным кодом машинного слова, системе было неизвестно.

Например, если в память нужно было записать сведения о рабочих, представленные в таблице 2.1, то без внутренней интерпретации в память была бы занесена последовательность из четырех групп машинных слов, соответствующих строкам этой таблицы [8]. При этом информация о том, какими группами двоичных разрядов в этих машинных словах закодированы сведения о рабочих, в системе отсутствуют. Они только известны программисту, который работает с данными для решения возникающих у него задач. Система не в состоянии ответить на вопросы типа «Что тебе известно о Ветрове?» или «Есть ли среди специалистов станочник?».

Таблица 2.1 − Данные по рабочим

Фамилия	Год рождения	Специальность	Стаж, лет
Топов	1985	Слесарь	20
Видоров	1980	Токарь	25
Банов	1991	Токарь	14
Ветров	1995	Станочник	10

При переходе к знаниям в память вводится информация о протоструктуре информационных единиц. В данном примере она представляет собой машинное слово, в котором указано, в каких разрядах хранятся сведения о фамилиях рабочих, годах их рождения, специальностях и времени работы. При этом должны быть словари, в которых перечислены имеющиеся в памяти системы фамилии, даты рождения, специальности и время стажа. Эти *атрибуты* могут играть роль имен для машинных слов, которые соответствуют строкам таблицы. По ним можно выполнить поиск нужной информации. Каждая строка таблицы есть экземпляр протоструктуры. СУБД обеспечивают реализацию интерпретируемости всех информационных единиц, хранящихся в базе данных.

2. *Структурированность* [8]. Информационные единицы могут иметь не жесткую структуру, а гибкую, т. е. иметь вложенность одних информационных единиц в другие. Каждая информационная единица может быть включена в состав любой другой, и из каждой единицы информации можно выделить отдельные составляющие. Должна существовать возможность произвольного установления между отдельными элементами информации различных отношений, типа «часть – целое», «род – вид», «элемент – класс».

3. *Связность* [7; 8]. В информационной структуре между информационными единицами должна быть возможность установления связей разного типа. Характер отношений может носить декларативный или процедурный характер. Например, две или более информационные единицы могут быть связаны отношением «одновременно», другие информационные единицы – отношением «причина – следствие», третьи отношением «взаимоположение». Приведенные отношения характеризуют декларативные знания.

Если между двумя информационными единицами установлено отношение «аргумент – функция», то оно характеризует процедурное знание, связанное с вычислением некоторых функций. Будем различать отношения структуризации, функциональные, каузальные и семантические отношения. С помощью первых задается иерархия информационных компонент, вторые несут процедурную информацию, позволяющую вычислять одни информационные единицы через другие, третьи задают причинно-следственные связи, последние соответствуют комплексным отношениям. Между информационными единицами могут устанавливаться другие связи, определяющие порядок выбора информационных единиц из памяти или указывающие на то, что две информационные единицы несовместимы друг с другом в одном описании.

Перечисленные три особенности знаний позволяют определить модель знаний, называемую семантической сетью, которая представляет собой иерархический граф, в вершинах которого находятся информационные единицы. Эти единицы имеют индивидуальные имена. Дуги семантической сети соответствуют разным связям между информационными единицами. При этом иерархические связи определяются отношениями структуризации, а неиерархические связи – отношениями других типов.

4. *Семантическая метрика.* [7; 8]. На множестве информационных единиц полезно задавать отношение, характеризующее ситуационную их близость, т. е. силу ассоциативной связи между ними. Это можно назвать отношением релевантности для информационных единиц. Такое отношение дает возможность выделять в информационной базе некоторые типовые ситуации (например, «продажа», «нерегулирование движения на перекрестке», «банкротство»). Отношение релевантности при работе с информационными единицами позволяет находить знания, близкие к уже найденным.

5. Активность. С разделением в системе используемых информационных единиц на данные и команды создалась ситуация, при которой данные пассивны, а команды активны. Все процессы, протекающие в компьютере, инициируются командами, а данные используются ими лишь в случае необходимости. Для интеллектуальных систем эта ситуация не подходит. В человеческой деятельности и в компьютере актуализация тех или иных действий запускается новыми знания. Выполнение программ в нем должно инициироваться новым состоянием информационной базы. Появление в базе новых фактов, установление связей может стать источником активности системы [14].

Перечисленные пять особенностей информационных единиц определяют ту границу, за которой данные превращаются в знания, а базы данных переходят в базы знаний. Совокупность средств, обеспечивающих работу с знаниями, образует систему управления базой знаний (СУБЗ). Однако в полной мере не существует баз знаний, в которых были бы реализованы внутренняя интерпретируемость, структуризация, связность, семантическая мера и активность знаний.

2.2 Знания и модели их представления

Переход от данных к знаниям является логическим следствием развития и усложнения информации и логических структур, обрабатываемых на ЭВМ. Знание абстрактно, оно не имеет какого-либо точного определения, есть много способов характеризовать это понятие. Один из используемых методов основан на идее интенционала – это его определение через понятие более высокого уровня абстракции с указанием конкретных свойств. Другой способ определения понятия – это перечисление понятий на более низком уровне иерархии или фактов, связанных с определенным понятием. Это определение с помощью данных, или расширение понятия. Экстенционал – специфические характеристики каждого элемента этого набора понятий и отношений [1; 6; 7; 10; 14].

Знания можно определить как совокупность информации, формирующей полное описание, соответствующее определенному уровню осведомленности об описываемом вопросе, предмете, проблеме и т. д. Иначе, знания – это выявленные закономерности предметной области (принципы, связи, законы), которые позволяют решать задачи и проблемы в этой области. С точки зрения искусственного интеллекта знания можно определить как формализованную информацию, на которую ссылаются и обрабатывают в процессе логического вывода [10].

База знаний – это совокупность знаний, описанных с использованием определенной формы их представления. База знаний – основа любой интеллектуальной системы. База знаний определяет интенциональную семантику моделей и содержит описание абстрактных сущностей: объектов, отношений, процессов.

Удобно разделить знания на две большие категории – факты и эвристику. Первая категория обычно указывает на общеизвестные обстоятельства в данной предметной области, поэтому ее знания называются текстовыми. Вторая категория основана на собственном опыте специалиста (эксперта) в данной предметной области, накопленном в результате многолетней практики [2; 10].

Знания делятся на процедурные и декларативные. Первые знания, по которым выполняются процедуры, «растворяются» в алгоритмах. Чтобы изменить их, необходимо изменить реализующие программы. Приоритет данных постепенно менялся, все больше знаний концентрировалось в структурах данных (таблицах, списках, абстрактных типах данных), т. е. возрастала роль декларативных знаний. Произошел сдвиг центра тяжести с машинного представления процедур на машинное представление знаний.

Традиционно структуры данных понимаются как декларативные знания, несущие только функцию отображения предметной области. Упорядоченная последовательность операций может быть выполнена над структурами данных − программой (процессом), реализующей некоторый алгоритм. Результатом программы является декларативное знание, а сама программа – процедурное знание.

Декларативные знания – это совокупность информации о качественных и количественных характеристиках конкретных объектов, явлений и их элементов, представленных в виде фактов и эвристик. Традиционно такие знания накапливались в виде различных таблиц и справочников, а с появлением компьютеров приобрели вид информационных массивов (файлов) и баз данных [10]. Декларативные знания хранятся в памяти компьютера, они доступны для использования. В форме декларативных знаний записывается информация о свойствах предметной области, фактах, которые в ней имеют место.

Процедурные знания хранятся в памяти компьютеров в виде описаний процедур, с помощью которых они могут быть получены. В форме процедурных знаний обычно описывается информация о предметной области, характеризующая способы решения проблем в этой области, а также различные инструкции, техники. Процедурные знания – это методы, алгоритмы, программы для решения различных задач, последовательности действий (в выбранной про-

блемной области) – они составляют ядро баз знаний. Например, в производственных моделях это набор производственных правил вида «ЕСЛИ А– ТО В»; в производственной сфере анализ процедурных знаний – это технологические знания о способах организации и реализации различных производственных процессов.

Дальнейшее развитие структур данных в рамках исследований искусственного интеллекта привело к появлению специальных структур данных: фреймов, семантических сетей, называемых моделями знаний. С появлением систем, основанных на знаниях, знания рассматриваются как записи на языках представления знаний, близких к естественным и понятных неспециалистам [10].

В семантическом плане обработка информации приобретает новую окраску, связанную с представлением и обработкой знаний, с получением требуемых знаний, но не с самим процессом. Одной из наиболее важных проблем, характерных для систем искусственного интеллекта, является представление знаний. Это связано с тем, что форма представления знаний оказывает существенное влияние на характеристики и свойства системы. Для того чтобы манипулировать видами знаний из реального мира с помощью компьютера, необходимо их смоделировать [10].

Проблема представления знаний – это представление взаимосвязей в конкретной предметной области в форме, понятной интеллектуальной системе. В рамках направления «Представление знаний» решаются задачи, связанные с формализацией и представлением знаний в памяти интеллектуальной системы (ИС). Для этого разрабатываются специальные модели представления знаний и языки описания знаний, выделяются различные типы знаний. Изучаются источники, из которых ИС может черпать знания, и создаются процедуры и методы, с помощью которых можно приобретать знания для ИС [10].

Проблема представления знаний для ИС актуальна, поскольку ИС – это система, функционирование которой основано на знаниях о проблемной области, хранящихся в ее памяти. Представление знания – это его формализация и структурирование, с помощью которых отражаются характерные черты знания: внутренняя интерпретируемость, структурность, связность, семантические показатели и активность. Представление знаний – это соглашение о том, как описывать объекты и связи реального мира. В рамках этого направления задачи, связанные с формализацией и представлением знаний в памяти ИС, решаются с использованием специально разработанных моделей представления знаний.

В отличие от методов представления данных, основанных на строгих алгоритмах, модели представления знаний имеют дело с информацией, полученной от специалиста (эксперта) в данной предметной области, которая может иметь противоречивый характер. Такая информация должна быть приведена к формализованному виду. Это делается с использованием различных методов и приемов, в частности, на основе многозначной логики, теории нечетких множеств и других математических моделей, а также специальных языков описания знаний. При работе со знаниями используются два основных подхода [2; 7; 9; 10]:

1. Логический (формальный) подход, при котором внимание уделяется изучению и применению теоретических методов представления знаний, формализации, а также логической полноте (например, создание моделей представления знаний на основе некоторого логического исчисления).

2. Эвристический (когнитивный) подход, который фокусируется на предоставлении возможностей для решения проблем. В то же время внимание сосредоточено на принципах организации человеческой памяти и эвристическом моделировании. В отличие от формальных моделей, эвристические модели обладают разнообразным набором инструментов, которые передают специфику конкретной области. Таким образом, эвристические модели превосходят логические с точки зрения выразительности и способности адекватно представлять предметную область. Типичными моделями представления знаний являются: логические модели; продукционные модели; семантические сети; фреймы. Существует два типа моделей представления знаний: формальные (логические, продукционные), неформальные (семантические, реляционные) [7].

В отличие от формальных моделей, которые основаны на строгой математической теории, неформальные модели не придерживаются такой теории. Каждая неформальная модель подходит только для определенной предметной области и поэтому не обладает универсальностью, присущей формальным моделям. Логический вывод строг и корректен в формальных системах, поскольку подчиняется строгим аксиоматическим правилам. Вывод в неформальных системах во многом определяется самим исследователем, который несет ответственность за его правильность. Каждый из методов ИИ соответствует своему собственному способу описания знаний [7].

2.3 Логические модели

Система искусственного интеллекта в приближенном смысле моделирует интеллектуальную деятельность человека, используя логику рассуждений. В упрощенном виде логические построения сводятся к такой схеме: из одной или нескольких посылок (которые считаются истинными) следует получить «логически правильный» вывод (заключение, следствие). Очевидно, что для этой цели необходимо, чтобы и посылки, и заключение были представлены на неком языке, отражающем предметную область, в которой делается вывод. В обычной жизни это тот или иной естественный язык общения, в математике – это язык определенных формул, в ИИ – язык представления знаний и т. д. Наличие языка предполагает, во-первых, наличие алфавита (словаря), отображающего в символической форме весь набор базовых понятий (элементов), с которыми придется иметь дело, во-вторых, набор синтаксических правил, на основе которых, используя алфавит, можно их использовать для построения выражений [8; 9].

Логические выражения, построенные на данном языке, могут быть истинными или ложными. Некоторые из этих выражений всегда верны, и они

объявлены аксиомами (постулатами). Аксиомы составляют базовую систему посылок, основываясь на которой и используя определенные правила вывода, можно получить новые выражения, которые также являются истинными.

Если эти условия выполняются, то система соответствует требованиям формальной теории. Это называется формальной системой (ФС). Система, построенная на основе формальной теории, также называется аксиоматической системой. Формальная теория $F = (A, V, W, R)$ должна удовлетворять следующему определению, характеризующему некоторую аксиоматическую систему [1; 7; 8; 12]: наличие алфавита (словаря) − A; набора синтаксических правил − V, набора аксиом, лежащих в основе теории, − W; набора правил вывода − R [8].

Основная идея подхода при построении логических моделей представления знаний заключается в том, что вся информация, необходимая для решения прикладных задач, рассматривается как набор фактов и утверждений, которые представлены в виде формул в некоторой логике. Знания представляются набором таких формул, а получение новых знаний сводится к реализации процедур логического вывода. Модели этого типа основаны на формальной системе, определяются множеством из четырех компонентов:

$$M = <T, P, A, B>.$$

Компонент T − это набор базовых элементов различной природы (слов из некоторого ограниченного словаря, деталей детского конструктора, входящих в определенный набор, и т. д.). Для подмножества T существует некоторый способ определить, принадлежит ли произвольный элемент этому множеству или нет. Процедура такой проверки может быть любой, но за конечное число шагов она должна дать положительный или отрицательный ответ на вопрос, является ли x элементом множества T. Обозначим эту процедуру через $N(T)$ [10].

Компонент P − это набор синтаксических правил, посредством которых T-элементы образуют синтаксически правильные агрегаты. Получаем, что синтаксически правильные фразы строятся из слов ограниченного словаря, новые конструкции собираются из деталей детского конструктора с помощью гаек и болтов. Объявляется существование процедуры $P(p)$, с помощью которой за конечное число шагов можно получить ответ на вопрос, является ли множество X синтаксически правильным.

В наборе синтаксически правильных агрегатов выделяется определенное подмножество A. Элементы A называются аксиомами. Что касается других компонентов формальной системы, то должна существовать процедура $P(A)$, с помощью которой для любого синтаксически правильного множества можно получить ответ на вопрос о том, принадлежит ли оно множеству A.

Компонент B − это набор правил вывода. Применяя их к элементам A, можем получить новые синтаксически правильные агрегаты, к которым снова можно применить правила из B. Именно так формируется набор агрегатов, полученных в этой формальной системе. Если существует процедура $N(B)$, с помощью которой можно определить для любой синтаксически корректной сущ-

ности, является ли она выводимой, то соответствующая формальная система называется разрешимой. Получается, правило вывода является наиболее сложным компонентом формальной системы.

Для знаний, включенных в базу знаний, можно предположить, что множество *A* формирует все информационные единицы, которые вводятся в базу знаний извне, и с помощью правил вывода из них выводятся новые производные знания. То есть формальная система – это генератор новых знаний, представляющий собой набор элементов знаний, выведенных в этой системе. Это свойство логических моделей делает их удобными для использования в БЗ, что позволяет хранить в БЗ только те знания, которые формируют множество А, и получать из них все остальные знания в соответствии с правилами вывода [7; 14].

Исчисление высказываний и исчисление предикатов являются классическими примерами аксиоматических систем. Эти ФС хорошо изучены и имеют хорошо разработанные модели логического вывода – основной процедуры в интеллектуальных системах. Следовательно, все, что может и гарантирует каждая из этих систем, справедливо для прикладных ФС как моделей конкретных предметных областей. В частности, это гарантии непротиворечивости вывода, алгоритмической разрешимости (для исчисления утверждений) и разрешимости (для исчисления предикатов первого порядка). По всем этим причинам математическая логика лежит в основе различных представлений в ИИ. Логическое представление служит отправной точкой для других представлений (сетевых и фоновых), используемых в ИИ.

Синтаксис логики предикатов. Язык логики предикатов задается синтаксисом. Для представления знаний основные синтаксические категории языка представлены такими символами, которые несут достаточно четкую информацию и дают достаточно четкую картину области рассуждений (экспертных знаний). Язык логики предикатов определяется следующим образом [12]. Алфавит содержит следующие классы символов [6]:

a) константы, обозначаемые ***a, b, c, d***, ...;

b) функциональные символы, представляемые как ***f, g, h,...;***

c) символы отношений **p, q, r, s,** ...;

d) переменные, обозначаемые через **x, y, z, v, u**, ...;

e) символы констант: TRUE (истина) и FALSE (ложь);

f) логические операторы (связки): – (отрицание, НЕ), v (дизъюнкция, ИЛИ), & (конъюнкция, И), → (импликация, ЕСЛИ ...ТО), <-> (эквивалентность, ЕСЛИ И ТОЛЬКО ЕСЛИ);

g) кванторы: ∃ – существования, ∀ – всеобщности;

h) круглые скобки ( ) и запятую «,».

Для конъюнкции используется также символ ∧, а для эквивалентности ↔ или ≡.

Далее опишем класс термов [10]: переменная есть терм; константа есть терм;

a) если f есть n – местная функция и $t_1,\ldots t_n$ – термы, $f(t_1,\ldots t_n)$ также терм.

Логическая формула задается следующими элементами:

a) если **p – n** – местное отношение и $t_1, \ldots, t_n$ – термы, то p $(t_1, \ldots, t_n)$ есть формула (называемая атомарной);

b) пропозициональные константы TRUE и FALSE суть формулы;

c) если **F** и **G** формулы, то формулами также являются $(\overline{F})$, $(F \vee G)$, $(F\&G)$, $(F \to G)$, $(F \leftrightarrow G)$;

d) если **F** – формула и **x** – переменная, то $(\exists$ **x F**$)$ и $(\forall$ **x F**$)$ – также формулы.

Для упрощения записи логических формул часто отбрасывают скобки, используя отношение порядка (старшинства) между логическими операторами и кванторами. Так, будем считать, что –, $\exists, \forall$ связывают сильнее, чем &, который в свою очередь связывает сильнее оператора $\vee$, а последний связывает сильнее, чем операторы $\to$, $\leftrightarrow$.

Основными задачами, решаемыми посредством логических моделях, являются следующие:

− установить или опровергнуть выводимость некоторой формулы (в общем случае эта задача алгоритмически неразрешима);

− доказательство полноты/неполноты некоторой формально логической системы, представленной множеством логических формул;

− установление выполнимости системы логических формул (нахождение интерпретирующей функции) или отыскание контрпримера, опровергающего их;

− определение следствий из заданной системы формул;

− доказательство эквивалентности двух формально-логических систем;

− поиск решения задачи на основе доказательства теоремы существования решения и др.

Константы. Они служат именами объектов, людей или событий в отличие от имен совокупностей. Константы представляются символами типа − Иван_2 (добавление 2 к слову Иван указывает на вполне определенного человека), Книга_22, Посылка_8, записываются с заглавного символа.

Переменные. Обозначают имена совокупностей, таких как человек, книга, посылка, событие, записываются со строчного символа. Запись − Книга_22 представляет вполне определенный экземпляр, а символ книга указывает либо множество «неких книг», либо «понятие книги». Символами x, y, z представлены имена совокупностей (определенных множеств или понятий).

Предикатные имена. Они задают правила соединения констант и переменных, например, правила грамматики, процедуры, математические операции. Для предикативных имен используются символы, такие как: Посылать, Писать, Разделить. Предикатное имя иначе называется предикатной константой.

Функциональные имена. Они представляют такие же правила, как и предикаты. Чтобы не спутать с предикатными именами, функциональные имена пишут одними строчными буквами: фраза, посылать, писать, плюс, разделить. Их называют также функциональными константами.

Символы, которые применяются для представления констант, переменных, предикатов и функций, не являются «словами естественного языка». Они суть символы некоторого представления – элементы «объектного языка» (в этом

случае языка предикатов). Описание должно исключать всякую неопределенность языка, поэтому имена индивидуумов содержат цифры, приписываемые к именам. Иван_1 и Иван_2 представляют двух людей с одинаковыми именами. Эти представления суть конкретизации имени совокупности «Иван». Предикат – это предикатное имя вместе с подходящим числом термов. Предикат называют также предикатной формой. Приведем примеры записей на логике предикатов, сопоставляя нескольким фразам естественного языка их перевод на язык логического описания [1; 3; 6].

По-русски: Иван дарит книгу Марии,

Логически: Дарит(Иван_1, Марии_4, Книга_20).

По-русски: Каждый человек гуляет,

Логически: $\forall$ x(Человек(x)&Гуляет(x)).

По-русски: Некоторые люди дремлют,

Логически: $\exists$ x(Человек(x)VДремлет(x)).

Сравнивая два последних примера, видим, что замена прилагательного «каждый» на «некоторые» влечет при переводе не только замену квантора $\forall$ на $\exists$, но и замену связки & на V. Это иллюстрирует тот факт, что перевод фразы естественного языка на логический в общем не является стандартной операцией [6]. Еще пример.

По-русски: ни один человек не опаздывает,

Логически: – $\forall \neq$x (Человек(x)&Опаздывает(x)).

К достоинствам логических моделей представления знаний относятся [5]:

1. В качестве базы используется классический аппарат математической логики, методы которой достаточно хорошо изучены и формально обоснованы.

2. Существуют эффективные процедуры вывода, особенно в логике первого порядка (метод резолюций), реализованный в языке логического программирования Пролог.

3. В базах знаний можно хранить лишь множество аксиом и правил вывода, а все остальные знания получать из них, используя правила и процедуры логического вывода [13].

ФС имеют и недостатки, которые заставляют искать иные формы представления [7]. Главный недостаток – это «закрытость» ФС, их негибкость. Модификация и расширение всегда связаны с перестройкой всей ФС, что для практических систем сложно и затратно. Также в этих моделях сложно учитывать динамические изменения. Поэтому логические модели представления знаний используются в тех предметных областях, которые хорошо локализуются и мало зависят от внешних факторов. Однако действительность не укладывается в рамки классической логики. Так называемая человеческая логика, применяемая при работе с неструктурированными знаниями, − это интеллектуальная модель с нечеткой структурой, и в этом ее отличие от классической логики. Логики, адекватно отражающей человеческое мышление, пока еще не создано [13].

2.4 Продукционные модели

Продукции являются распространенными моделями представления знаний в области ИИ. Они, с одной стороны, близки к логическим моделям, что позволяет организовать для них эффективные процедуры вывода, а с другой стороны, они лучше отражают знания, чем классические логические модели. В них отсутствуют строгие ограничения, характерные для математического исчисления, что позволяет изменять интерпретацию элементов описания [6; 7; 8; 14].

Продукционные модели могут использовать отдельные элементы логических и сетевых моделей. Применение правил вывода, которые называются продукциями, взято из логических моделей, а описание знаний в виде сети заимствовано из сетевых моделей. В результате применения правил вывода к фрагментам описания сети последняя трансформируется путем изменения ее фрагментов, путем наращивания сети и исключения из нее ненужных фрагментов [7]. В этих моделях процедурная информация выделяется и описывается средствами, отличными от декларативной информации. Вместо логического вывода для логических моделей в продукциях появляется вывод, основанный на знаниях.

Психологические исследования процессов принятия решений человеком показали, что рассуждая и принимая решения, он использует правила продукций (от английского Production − правило дедукции, порождающее правило). Продукционное правило может быть представлено следующим образом:

n: CP; C; $A \rightarrow B$; Q,

где i − индивидуальный номер продукции; CP − описание класса ситуаций, в котором данная структура может использоваться; C − условие, при котором продукция активизируется; $A \rightarrow B$ − ядро продукции, в виде

ЕСЛИ A_1, A_2,..., A_n ТО B.

Такая запись означает, что «если все условия от A_1 до A_n являются истиной, то B также истина», или же «когда все условия от A_1 до A_n становятся истиной, то следует выполнить действие B»; Q − следствие продукционного правила, описывает операции и действия (процедуры), которые необходимо выполнить после выполнения B. Например, внести изменения в данные либо в саму продукцию. *Пример 1:*

ЕСЛИ y является матерью x, (A1)
z является сестрой y, (A2)
ТО z является тетей x, (B1)

В этом случае $n = 2$. При $n = 0$ получаем знания, состоящие только из вывода, т. е. простой факт, например, «Удельный вес золота равен 19 г/см^3».

Суть использования правил продукции для представления знаний состоит в том, что левой части ставится в соответствие некоторое условие, а правой части − действие:

ЕСЛИ <перечень условия>, ТО <перечень действий>.

В такой интерпретации левая часть правил оценивается по отношению к базе данных (набору фактов) системы, и если эта оценка в определенном смысле соответствует логическому значению «ИСТИНА», то выполняется действие, заданное в правой части продукции.

В общем случае под условием понимается некоторое предложение − образец, по которому осуществляют поиск в базе знаний, а под действием − процедуры, выполняемые при успешном завершении поиска, − это могут быть реальные действия, если система управляющая, или заключение − вывод, представляющий собой новое знание, или некоторая цель [10]. При использовании продукционной модели база знаний состоит из множества правил. Процедура, управляющая перебором правил, называется *машиной вывода.* Механизм выводов связывает знания в одно целое, а затем выводит из их множества некое заключение или дает отрицательный ответ.

*Пример 2: п*усть в БЗ вместе с описанными выше знаниями содержатся еще и такие знания:

ЕСЛИ *z* является отцом *x, z* является отцом *y, x* и *y* не являются одним и тем же лицом и они женского пола, ТО *x, y* являются сестрами; где *x, y, z* − переменные.

В продукционных системах, основанных на этих моделях, процесс обработки информации реализуется двумя вариантами. По первому варианту запись просматривается в прямом направлении, когда проверяется левая часть продукционного правила, задача решается в направлении от начального состояния к конечному. После совпадения левой части, правая часть продукционных правил срабатывает, что соответствует логическому выводу нового факта. После добавления полученных фактов в БЗ эта процедура повторяется. Процесс закончен, если выполняется последнее продукционное правило, или в БЗ поступает факт, являющийся искомым решением [13].

Во втором варианте обработка информации осуществляется в обратном направлении − «выбор гипотезы и ее проверки («от цели к фактам»). При обратном движении возникает подцель, из которой решение может быть получено при прямом движении. Проверяются правые части продукционных правил с целью обнаружить в них нужный факт. Если такие продукционные правила существуют, то проверяется, удовлетворяется ли левая часть продукционного правила имеющимися в БЗ фактами. Если да, то гипотеза считается подтвержденной, если нет – она отвергается [13].

Пример 3: Имеется фрагмент БЗ из двух правил [5]:

П 1 : ЕСЛИ «отдых – весной» и «человек – лыжник», ТО «ехать в горы».

П 2 : ЕСЛИ «не любит жару», ТО «отдых весной».

Предположим, в систему поступили факты: «человек – лыжник» и «не любит жару»

Прямой вывод: исходя из фактов, получить ответ.

1-й проход:

Шаг 1. Проверяем П1 – не работает – не хватает факта «отдых – весной».

Шаг 2. Проверяем П2, правило работает, в базу поступает новый факт «отдых – весной».

2-й проход:

Шаг 3. Пробуем снова П1, оно сработает, появляется факт «ехать в горы», который и выступает как совет, который дает ЭС.

Обратный вывод: подтвердить выбранную цель (гипотезу) при помощи имеющихся правил и фактов.

1-й проход:

Шаг 1. Гипотеза – «ехать в горы» становятся новой целью, и имеется правило П2, где она в правой части.

Шаг 2. Подцель «отдых весной» есть в правиле П2, есть факт «не любит жару», который подтверждает эту подцель и активизирует ее в виде дополнительного факта.

2-й проход:

Шаг 3. Пробуем П1, подставляем новый факт из П2, подтверждается искомая гипотеза «ехать в горы».

Достоинства продукционных правил [13].

1 Масштабируемость − часть продукционных правил может удаляться или дописываться в базу знаний, также принцип изменения состава продукционных систем позволяет использовать заранее известные (класс).

2 Простота интерпретации — «прозрачная» структура продукционных правил облегчает их смысловую интерпретацию.

3 Прозрачность − знания в виде правил являются естественными с точки зрения человеческого интеллекта.

Недостатком продукционных моделей является конфликтность, когда число правил становится очень большим и возникают противоречия между ними от добавления нового правила.

Ядро продукционных моделей. Основным элементом продукции является ее ядро в виде $A \rightarrow B$. Интерпретация ядра продукции может быть различной и зависит от того, что стоит слева и справа от знака секвенции →. Обычное прочтение ядра продукции выглядит так: ЕСЛИ A, ТО B, более сложные конструкции ядра допускают в правой части альтернативный выбор, например,

ЕСЛИ A, ТО C_1, ИНАЧЕ C_2.

Это может истолковываться в обычном логическом смысле как знак логического следования C_1 из истинного A (если A не является истинным выражением, то о C_1 ничего сказать нельзя). Возможны и другие интерпретации ядра продукции, например, A описывает некоторое условие, необходимое для того, чтобы можно было совершить действие C_1.

В модели продукций заданы процедуры управления компонентами, с помощью которых происходит актуализация продукций и выбор для выполнения продукции из числа активированных [7].

2.5 Семантические сети

В основе семантических моделей лежит структура, названная семантической сетью. Сетевые модели формально можно задать в виде $H = <IE, C_1, C_2, ..., C_n, D>$. Здесь *IE* есть множество информационных единиц; $C_1, C_2, ..., C_n$ – множество типов связей между информационными единицами. Отображение *D (display)* задает между информационными единицами, входящими в *I*, связи из заданного набора типов связей [1; 7; 8; 13].

В зависимости от типов связей, используемых в этой модели, имеются следующие разновидности сетей: для описания классификаций, функциональные и сценариев. В сетях классификаций используются отношения структуризации, они позволяют в БЗ вводить разные отношения дерева между информационными единицами. Функциональные сети имеют отношения процедурные, вычисление одних элементов через другие. В сетях типа сценарии используются отношения: «причина – следствие», «средство – результат», «орудие – действие» и т. д. Если в сетевой модели допускаются связи всех трех видов, то ее называют семантической сетью [13].

Способ представления знаний с помощью сетевых моделей близок к тому, как они представлены в текстах на естественном языке. В его основе лежит идея о том, что вся необходимая информация может быть описана как совокупность троек *(a, c, b),* где *a* и *b* − объекты или понятия, *c* − бинарное отношения между ними [10].

Семантическая сеть − это модель, основой для которой является формализация знаний в виде ориентированного графа с размеченными вершинами и дугами. Вершинам соответствуют объекты, понятия или ситуации, а дугам − отношения между ними. Это наиболее общая модель представления знаний, так как в ней имеются средства реализации всех характерных для знаний свойств: внутренней интерпретации, связанности, структурированности, семантической метрики, активности [13].

Достоинства сетевых моделей: большие графические возможности; наглядность модели знаний, представленной картинкой; близость структуры сети, семантической структуре фраз на естественном языке; соответствие медицинским представлениям об организации долговременной памяти человека.

Недостатки: формирование и модификация такой модели трудоемки, сетевые модели представляют собой неактивные структуры, для обработки которых необходим специальный математический аппарат формального вывода.

Проблема поиска решения на основе БЗ семантической сети сводится к задаче нахождения части сети, соответствующей некой подсети решаемой задачи. Это формирует еще недостаток сетевой модели − сложность поиска вывода на семантических сетях [13].

Вывод: сетевые модели являются наглядным и достаточно универсальным средством представления знаний. Но их формализация в конкретных случаях представления, использования и модификации знаний оказывается трудоемкой при наличии множества отношений между ее элементами.

2.6 Фреймовые модели

Термин *фрейм* (frame − каркас) предложен профессором Марвином Минским в 70-е годы для обозначения структуры знаний представления пространственных сцен окружающего мира. Под фреймом понимается абстрактный образ или ситуация окружающего мира. Допустим, слово «комната» вызывает образ комнаты − «жилое помещение с четырьмя стенами, полом, потолком, окнами, дверью, мебелью». Из этого описания ничего нельзя убрать, например, убрав окна, мы получим уже чулан, а не комнату. Но в нем есть «слоты» − незаполненные значения некоторых атрибутов − количество окон, цвет стен, высота потолка, покрытие пола и т. д. Этот образ называется фреймом, формализованная модель этого образа тоже фрейм [1; 7; 8; 10; 13].

Фреймовая модель представляет собой систематизированную в виде единой теории модель памяти человека. Ключевым элементом в этой модели является понятие фрейма − структуры данных для представления концептуального объекта. Информация, относящаяся к этому фрейму, содержится в компонентах фрейма − слотах. В отличие от рассмотренных моделей знаний во фреймовых моделях фиксируется строгая структура, которая называется протофреймом (фреймом-прототипом, образцом) [10, 13]. Она выглядит так:

(Имя фрейма:
Имя слота 1(значение слота 1)
Имя слота 2(значение слота 2)
. . . . . .
Имя слота К (значение слота К)).

Значением слота может быть множество элементов (числа, математические соотношения, тексты на естественном языке, программы, правила вывода, ссылки на другие слоты фрейма). В качестве значения слота может использоваться набор слотов более низкого уровня, что позволяет во фреймовых представлениях реализовать «принцип вложенности» [10].

В качестве слота может выступать имя некого фрейма, образуются сети фреймов. Все фреймы взаимосвязаны и образуют единую структуру, в которой объединены декларативные и процедурные знания. Это позволяет производить композицию и декомпозицию информационных структур аналогично тому, как это происходит в памяти человека при формировании структуры его знаний.

Кроме абстрактных фреймов-прототипов, хранящихся в БЗ, используются *фреймы-экземпляры,* которые формируются для отображения конкретных ситуаций на основе введенных данных. При конкретизации фрейма ему и слотам присваиваются конкретные имена, происходит наполнение слотов, т. е. из протофреймов получаются их экземпляры.

Структура таблицы 2.1, содержащей список рабочих, записанная в виде протофрейма, имеет вид [10; 13]

(СПИСОК РАБОТНИКОВ:
Фамилия (значение слота 1);
Год рождения (значение слота 2);
Специальность (значение слота 3);
Стаж (значение слота 4)).

Если в качестве значений слотов использовать данные таблицы 2.1, то получится фрейм-экземпляр [8]:

(Список работников:
Фамилия (Топов – Видоров – Банов – Ветров);
Год рождения (1985 – 1980 – 1991 – 1995);
Специальность (слесарь – токарь – токарь – станочник);
Стаж (20 – 25 – 14 – 10)).

Связи между фреймами задаются значениями слота с именем «Соединение». Эксперты по ИС считают, что нет необходимости специально выделять фреймовые модели в представлении знаний, поскольку в них объединены все особенности моделей остальных типов.

Если в качестве значений слотов использовать реальные данные из таблицы, то получится фрейм-экземпляр. Важнейшим свойством фреймов является заимствованное из теории семантических сетей наследование свойств. И во фреймах, и в семантических сетях наследование происходит по АКО-связям (анг. *A Kind Of*). Слот АКО указывает на фрейм более высокого уровня иерархии, откуда он наследуется, но наследование свойств может быть частичным [13]. При конкретизации фрейма ему и слотам присваиваются конкретные имена, и происходит заполнение слотов. Из протофреймов получаются *фреймы-экземпляры.* Переход от исходного протофрейма к фрейму-экземпляру может быть многошаговым за счет пошагового уточнения значений отдельных слотов.

Фреймовые модели является универсальными, поскольку позволяют сформировать все множество знаний о реальном мире через: фреймы-структуры для обозначений понятий (заем, залог, вексель); фреймы-роли (маркетолог, кассир, пользователь); фреймы-сценарии (создание, собрание ученых, празднование юбилея); фреймы-ситуации (тревога, катастрофа, аварийный режим устройства).

Главными достоинствами фреймовой модели являются способность отражать основу организации памяти человека, а также естественность, наглядность представления, модульность, поддержка возможности использования значений слотов по умолчанию. Однако фрейм-представление является не конкретным элементом представления знаний, а некоторой абстрактной концепцией, реализуемой по-разному в разных языках. Теория фреймов послужила толчком к разработке нескольких языков представления знаний (см. главу 3).

Отметим, что концепция объектно-ориентированного программирования может рассматривать как техническую реализацию понятий, соответствующих фреймовой модели знаний (классы можно рассматривать как фреймы). Недостаток фреймовых моделей – нет универсального механизма вывода. Он устраняется при помощи процедур, реализуемых в каждом случае программистом ИС.

2.7 Элементы нечеткой и вероятностной логик

Математическая теория нечетких множеств (fuzzy sets) и нечеткая логика (fuzzy logic) являются обобщениями классической теории множеств и классической формальной логики. Данные понятия были впервые предложены американским ученым Лотфи Заде (Lotfi Zadeh) в 1965 году. Основной причиной появления новой теории стало наличие нечетких и приближенных рассуждений при описании человеком процессов, систем, объектов [2; 7].

Нечеткая логика, выделившаяся из теории нечетких множеств, − это разновидность непрерывной логики, в которой логические формулы могут принимать истинностные значения между 1 и 0. В нечеткой логике достоверность представляется как истинностное значение между 1 и 0, и значения, приписанные правилам, это и есть истинностные значения (вероятность определяется в статистическом смысле, и в отличие от нее истинностное значение это некоторое произвольное субъективное значение, не имеющее никакого статистического смысла). Пусть t_x и t_y – истинностные значения предпосылок *X* и *Y* некоторого правила, тогда истинностное значение *t* предпосылки в случае связей И и ИЛИ определяется следующим образом.

1. При связи И *t* предпосылки = min $\{t_x, t_y\}$.
2. При связи ИЛИ *t* предпосылки = max $\{t_x, t_y\}$.

Если в общем случае *t* правила есть истинностное значение, приписанное правилу, то истинностное значение *t A* , распределенное на вывод, определяется как *t A* = min { *t* предпосылки, *t* правила).

Определение минимума – это идея, свойственная нечеткой логике и отличающая ее от других методов (в которых производится умножение). В качестве связи **КОМБ** можно рассматривать одну из связей И или ИЛИ. Собственно говоря, в нечеткой логике и нечетких выводах рассматривается случай, когда множества *X, Y, A* и другие, описанные в предпосылках и выводах правил, суть нечеткие множества.

Нильсон предложил идею расширения логики и ввел понятие вероятностной логики, в которой всем логическим формулам приписывается вероятность. Здесь вероятность вновь соответствует законам Байеса. Связь логики и вероятности важна также с точки зрения рационального построения новой теории на основе теории логического моделирования. И хотя эта теория еще не доведена до использования на уровне вычислений, ознакомимся с ней на простых примерах [2]. Рассмотрим три логические формулы в логике высказываний:

A , A € B , B.

Представим следующие вертикальные векторы:
1 = (1, 1, 1) − мир истинности A, A € B, B;
2 = (1, 0, 0) – мир истинности A и лжи A € B, B;
3 = (0, 1, 1) − мир лжи A и истинности A € B, B;
4 = (0, 1, 0) − мир лжи A, B и истинности A € B.

Если выбрать один из возможных миров, то образуется традиционная двузначная логика. В вероятностной логике рассматриваются состояния, когда одновременно с некоторой вероятностью могут существовать несколько возможных миров. Например, пусть вероятность, с которой возможна интерпретация в мире 1, равна 0,4, а вероятности интерпретации в мирах 2, 3, 4 соответственно равны 0,3, 0,2, 0,1 (сумма вероятностей возможных миров равна 1), тогда представим следующим образом вектор вероятностей возможных миров P = (0,4, 0,3, 0,2, 0,1).

Таблица 2.2 − Различие нечеткой и вероятностной логики

Логики / Критерии оценки	Нечеткая логика	Вероятностная логика
Значения истинности	Интерввал [0,1]. Истинное начзначение – субъективная величина	Существование различных исходов событий с какой-то степенью вероятности
Основные логические формулы	Логические формулы такие же, как и в четкой, принимают значения истины на интервале [0,1]	Всем логическим формулам приписывается вероятность, с которой эта формула будет работать
Правила вывода	Верны всегда. Но интерпретация полученного результата субъективна	Правила вывода верны с вероятностью. Результат зависит от того, как сработает правило в конкретной ситуации
Расширяемость	Все знания прописаны жестко. Возможность добавления новых знаний отсутствует	Все знания прописаны жестко. Возможность добавления новых знаний отсутствует

2.8 Приобретение и формализация знаний

Сходство и различие моделей. Рассмотренные модели представления знаний близки друг к другу. Разница лишь в том, насколько удобно и естественно представлять определенные понятия в виде логических формул, семантических сетей, фреймов или продукций. Можно отметить следующее. Логическая и продукционная модели отличаются некой выраженной процедурной формой, поэтому они часто используются для описания процедурных знаний. Вместо строгого вывода в логических моделях продукционные используют вывод, основанный на знаниях. Модели знаний, основанные на семантических сетях, используются для описания декларативных знаний [3; 10].

Смешанные модели. В системах ИИ одновременно может использоваться несколько моделей представления знаний. Например, значения некоторых слотов во фрейме могут быть продукциями. Смешанные представления оказываются наиболее многообещающими. Некоторые элементы логических и сетевых

моделей используются в производственных системах. Они позволяют организовать эффективные процедуры вывода (близость к логическим моделям) и визуально отражать знания в виде сетей (близость к семантическим сетям). В результате применения правил вывода к фрагментам описания сети последняя преобразуется путем изменения ее фрагментов, наращивания и исключения из нее ненужных компонент [1; 2; 7].

Фрейм можно рассматривать как фрагмент семантической сети, предназначенный для описания объекта (ситуации) проблемной области со всеми присущими ему свойствами. Фреймовый подход к представлению знаний более жесткий, чем тот, который основан на использовании семантической сети. Все, что касается объекта или ситуации, что важно с точки зрения решаемой проблемы, не «размыто» в сети, а представлено в виде фрейма. В свою очередь, фрейм может быть представлен в виде сети, состоящей из вершин и дуг (взаимосвязей), так что «нижние уровни» фрейма заканчиваются слотами, которые заполняются конкретной информацией при вызове фрейма.

Подводя итог анализу моделей представления знаний, можно сделать два основных вывода: невозможно дать универсальные рекомендации по выбору моделей. Выбор конкретной модели определяется возможностью и удобством представления изучаемой проблемной области с учетом необходимости не только представлять, но и использовать знания. Однако чаще используются эвристические, а не логические модели представления знаний. Смешанные представления оказываются наиболее мощными [12].

Представление – это действие, которое делает концепцию воспринимаемой с помощью рисунка, обозначения, языка или формализма. Теория знания изучает связи между субъектом (учащимся) и объектом. Знание (в объективном смысле) – это то, что известно (то, что мы знаем после изучения). Представление знаний – это формализация истинных убеждений с помощью цифр, записей или языков. Особенно интересна формализация, воспринимаемая компьютером. Возникает вопрос о представлении знаний в его памяти, т. е. о создании языков и формализмах представления знаний. Они преобразуют визуальное представление (через речь, изображение, естественный язык, формальный язык, такой как алгебра или логика, рассуждения и т. д.) в пригодное для ввода и обработки в компьютере. Результатом формализации должен быть набор инструкций, составляющих часть языка программирования [10; 12].

Представление знаний имеет пассивный аспект: книга, таблица, память, наполненная информацией. ИИ подчеркивает активный аспект репрезентации: знание должно стать активной операцией, позволяющей не только запоминать, но и извлекать воспринимаемые (приобретенные, ассимилированные) знания для суждений, основанных на них. Во многих случаях знания, которые должны быть представлены, относятся к довольно ограниченной области, например, описания: состояния человека; ситуации в игре (например, расположение фигур в шахматах); размещение персонала предприятия; ландшафт местности [12].

Численная формализация таких описаний, как правило, не очень эффективна. Напротив, использование символического языка, такого как язык мате-

матической логики, позволяет формулировать описания в форме, одновременно близкой как к обычному языку, так и к языку программирования. Однако математическая логика позволяет рассуждать на основе знаний: логические выводы – это активные операции получения новых знаний из приобретенных.

Приобретение знаний – это их выявление из различных источников и преобразование в желаемую форму, а также передача в БЗ. Источником знаний могут быть книги, архивы. Другой тип знаний – это экспертные знания, которые доступны специалистам, но не зафиксированы в документах. Третий тип источников знаний – это наблюдения за окружающей средой. Ввод знаний в БЗ осуществляется инженером по знаниям, который может преобразовать знания в форму, удобную для дальнейшей обработки. Существует три возможных этапа взаимодействия инженера по знаниям с экспертом [12].

1. Подготовительный, в ходе которого оба участника готовятся к диалогу. Эксперт должен быть заинтересован и уметь объяснить свои знания. Аналитик должен быть подготовлен в этой предметной области.

2. Установление общего кодекса. Необходимо определить основные понятия, взаимосвязи между ними, уровень детализации.

3. Эпистемологический этап, на котором выясняются законы предметной области, условия достоверности и истинности утверждений, структурирование через введение отношений и т. д.

При внедрении системы сбора знаний возможна следующая последовательность действий [1; 3]:

– интервью для определения области и разделения на поддомены;

– формирование декларативной модели предметной области;

– анализ протокола и идентификация концепций и отношений предметной области для дополнения модели процедурными знаниями;

– проверка полноты модели, если концепции недостаточно описаны, повторяется анализ взаимодействия и протокола.

Формализация знаний. При формализации качественных знаний может быть использована теория нечетких множеств задач, особенно те аспекты, которые связаны с лингвистической неопределенностью при работе с экспертом на естественном языке. В этом случае вводятся функции принадлежности к определенному диапазону, например, для указания возраста: очень молодой – до 25 лет; молодой – до 40 лет; средний возраст до 60 лет; выше среднего – до 70, пожилой – старше 70 лет.

Существует две группы методов построения функций принадлежности – прямые и косвенные. В прямых методах эксперт устанавливает правила определения значений. В косвенных методах значение функции принадлежности выбирается таким образом, чтобы удовлетворять заранее сформулированным условиям. Прямые методы используются для описания понятий, которые характеризуются количественными характеристиками (вес, рост, масса, объем). Косвенные методы используют интервальную шкалу (пример с возрастом человека) [12].

2.9 Основы извлечения знаний из Интернета

Виды поиска информации. Извлечение знаний можно определить как нахождение и анализ полезной информации. Данную область деятельности принято подразделять на две части: автоматический поиск информации в документах Сети − Web content mining и обнаружение и обработка информации, касающейся работы пользователей с сервером, − Web usage mining. Рост объема доступных через данных, хранимых в слабоструктурированном виде, способствовал появлению автоматических программных средств поиска информации и получения данных об использовании определенных ресурсов. Возник целый ряд интеллектуальных систем, основная задача которых состоит в эффективном извлечении знаний из Интернета [5; 9; 11].

Процесс автоматического изучения характеристик доступа пользователей к серверам может включать изучение наиболее популярных путей посещения, нахождение ассоциативных правил, кластеризацию и т. д. Для решения этих задач можно использовать накопленные технические документы. Организации собирают огромные объемы информации, автоматически создаваемой серверами и оседающей в журналах. Источниками информации являются также ссылочные журналы, в которых содержится информация для каждой страницы, на которую есть ссылка, журналы браузеров и регистрационные или анкетные данные пользователей, собранные CGI-сценариями [11].

Основные потребители систем категории usage mining − организации, торгующие или предоставляющие услуги в Сети. Главными задачами для них являются персонификация наполнения страниц и оптимизация сайта с точки зрения упрощения навигации [4; 9]. Также подобные системы представляют интерес для провайдеров Internet и сетевых администраторов. Основными областями применения в этом случае являются оптимизация работы сети, минимизация трафика и оптимизация предоставляемых услуг (например, интеллектуальное кэширование данных [11].

Большинство традиционных систем мониторинга Сети предоставляют возможность фильтрации и получения статистической информации о пользователях. Подобный инструментарий помогает определять количество обращений к разным файлам и серверам, адреса отдельных пользователей, при этом такие системы рассчитаны на малый или ограниченный поток данных и редко предоставляют возможности анализа связи между обращениями к файлам и логикой их расположения. Рассмотрим инструменты, дающие аналитику более полную информацию [11].

Сбор информации. Сбор информации на уровне сервера представляет собой отбор информации непосредственно из журналов веб-сервера. Все серверы автоматически ведут журнализацию; при этом журналы, как правило, хранятся годами. Рассмотрим детально, какую именно информацию предоставляет журнал сервера, отвечающий требованиям стандарта Common Log Format (CLF) [11].

Большинство современных веб-серверов предоставляют возможность администратору выбирать, какие поля должны включаться в журнал, а какие − нет. Самые распространенные из дополнительных полей, которые при добавлении к Common Log Format образуют так называемый Combined Log Format, таковы: обратившееся приложение; URL документа, с которого осуществлено обращение; значения cookies [4; 9].

У журналов сервера есть и недостатки. Основным из них является неполнота информации. Обращения к сохраненным на каком-либо уровне страницам, например, у пользователя в локальном кэше, не заносятся в журнал сервера, также в журналы сервера не попадают данные, пересылаемые с помощью метода POST. Альтернативный метод сбора данных на самом сервере – анализ на уровне пакетов. Таким образом, можно анализировать на уровне отдельных запросов TCP/IP, но для накопления таких данных, как правило, требуется написание дополнительных программ. На уровне сервера можно собирать данные запросов, полученных через формы на страницах или после выполнения различных сценариев. Достаточно интересным может быть анализ выданных различным пользователям cookie; информация об этом также хранится на сервере [4; 11].

Информация о клиентах. Нужно собирать информацию о посещениях на уровне клиента. Один из способов – использование Java-программ, подгружаемых через страницы интересующего нас сервера; однако функциональность подобных приложений ограничена, кроме того, сам пользователь с помощью настроек своего браузера способен исключить или ограничить возможность подобного сбора информации. Вторым способом могло бы стать внесение изменений в программы для просмотра Сети. Но следует понимать, что вносить в журнал придется все сразу, поскольку если в будущем понадобится собирать данные по какому-либо новому параметру, то внести соответствующие изменения в браузеры всех клиентов будет почти невозможно [4; 11].

Также при таком подходе возникают две неразрешимые проблемы: во-первых, как правило, никто не хочет, чтобы его шаги протоколировались, а затем собранные данные куда-либо отсылались, а во-вторых, мало кто станет обновлять свое программное обеспечение из-за нужд третьей стороны, осуществляющей сбор данных. Таким образом, сбор информации на стороне клиента любых других методов затрагивает проблему сохранения неприкосновенности личной жизни, и пока такие методики мало применимы. Тем не менее на данный момент существует несколько систем, использующих подобный подход [4; 11].

Анализ данных прокси-сервера может предоставить информацию о характере просмотра Сети анонимной группы пользователей, использующих один прокси-сервер. В случае создания специализированного программного обеспечения для прокси-серверов можно добиться некоторых преимуществ по сравнению со сбором на стороне сервера или клиента. Решается проблема со снижением быстродействия сервера; кроме того, достаточно просто можно осуществить подключение нового сайта к сбору статистики или обновление системы для взаимодействия с новыми версиями браузеров.

Как альтернативу сбору информации на стороне сервера или шлюза можно рассмотреть сбор данных на узлах сети. Во-первых, не всегда возможен доступ к журналам сервера, во-вторых, не всегда данные, собираемые на сервере, релевантные к решаемой задаче. Кроме того, добавление на сервер каких-либо программных средств сбора информации может быть невозможно, или может просто замедлить сервер, что крайне нежелательно. Выходом может служить размещение датчиков в узлах сети на подходе к серверу. В таком случае сервер разгружается от излишнего программного обеспечения. Работа ведется на уровне протоколов и, как правило, сбор идет на уровне пакетов TCP/IP [4; 11].

Хорошим примером системы сбора данных и знаний служит Web Traffic Warehouse [5]. Создатели системы обнаружили, что расположение сборщика данных влияет на качество получаемых результатов. Вследствие асинхронного характера передачи данных по Сети входящий и исходящий трафики могут проходить по разным физическим каналам. Просматривая его в некоторой точке сети, можно увидеть только одну из сторон диалога. Чтобы этого избежать, система собирает данные непосредственно на уровне приложений. Тогда, получая данные как от приложения, так и от клиента, можно получать дополнительные параметры (такие, как потеря и задержка пакетов). Кроме того, много коррелированных источников могут использоваться для выявления точек потерь или задержек информации [4; 11].

Информация о сети. Коллекционирование всех пакетов позволяет получить подробные данные о сети – дополнительная информация извлекается из журналов приложений (межсетевые экраны, серверы и др.). Информация о состоянии сети может быть получена периодичным просмотром счетчиков, непосредственно находящихся на сетевых элементах, имеющих SNMP-доступ к базе Management Information Base. Более детальные данные можно найти в полях данных, которые поддерживаются датчиками RMON, что расширяет возможности хранения данных большинства сетевых элементов. Совмещая множество источников данных, можно получать очень подробную картину.

Следует заметить, что независимо от места сбора информации в полном потоке данных содержатся пароли, частная корреспонденция, тексты документов. Даже IP-адреса источника или получателя в некоторых случаях могут быть сочтены частной информацией, особенно с учетом того, что по адресу можно определить компьютер, с которого была произведена операция. Можно исключать из обработки подобные данные, но это приводит к потерям ценной информации. Разработчики должны выбирать подходящий компромисс. Например, желательно скрывать IP-адреса; при этом есть потребность определять вхождения на один сайт с разных компьютеров, для чего необходимо осуществлять проекции от истинных к зашифрованным адресам [4; 11].

Подготовка данных. На этом этапе могут выполняться некоторые простые интеграционные задачи, например, совмещение нескольких журналов и отсев ненужных для решаемой задачи данных. Найденные ассоциации полезны только в том случае, если данные в журнале показывают точную картину доступа пользователей к сайту, иногда удаление записей о файлах с «неважными»

суффиксами (jpg, gif, map и др.) может существенно очистить записи. Во многих случаях требуется также очистить записи от неудачных запросов. Обычно также требуется отсекать запросы со стороны различных автоматических агентов (в частности, это агенты поисковых систем, служащие для создания индексов страниц и слов во внутренних базах данных, автоматические верификаторы ссылок и инструментарий для управления сайтом) [4; 11].

Сходная, но гораздо более сложная проблема состоит в определении обращений, которые не заносятся в журнал, механизмы локального кэша или прокси-сервера искажают картину перемещений пользователей в Internet. Методы борьбы с этой проблемой используют топологию сайта и ссылочные журналы с временной информацией для обнаружения пропущенных ссылок. Более точную картину перемещений пользователя можно составить, только если пропуски страниц были единичными, в таком случае можно дополнять путь пользователя; эта проблема получила название «заполнение пути» (path completion) [4; 11].

2.10 Обработка знаний из Интернета

После очистки данных возникает задача разделения журнала на разные сеансы разных пользователей. Для того чтобы однозначно отличать запросы разных пользователей от вышеупомянутых полей журнала, можно использовать IP-адрес, пользовательский агент и адрес запрашиваемого документа. Рассмотрим три типа спорных ситуаций для идентификации разных пользователей [4; 9; 11].

A. Один IP-адрес/много пользователей. Очень распространенная ситуация возникает, когда прокси-сервер используется провайдером, кроме того, когда случайный адрес выделяется любому пользователю при установлении связи с провайдером (очень типично при общении по телефонной линии), два разных пользователя могут получить один и тот же адрес.

B. Много IP-адресов/один пользователь. Это также очень распространенный случай, который возникает, когда провайдер динамически распределяет адреса. В некоторых случаях (пример – America Online) при каждом новом доступе к странице пользователю предоставляется новый адрес. Для случаев A или B можно выделить разных пользователей в зависимости от типа браузера и отслеживать путь пользователя в одном сеансе, находя для каждого документа, который его вызвал, и, таким образом, выделять отдельные сеансы от входа на сайт до страницы, с которой не было перехода внутри сайта.

C. Один пользователь использует разные браузеры. В этом случае, если IP-адрес не предоставляет достоверных данных, вы можете использовать только два метода, описанных ниже, принимая при этом во внимание, что файлы cookie не всегда будут работать корректно.

В любом из упомянутых случаев, если данных журнала недостаточно для идентификации, могут быть использованы файлы cookie и уникальная регистрация пользователя. У каждого из этих методов есть недостатки: пользователь может удалять файлы, расположенные на его компьютере, а обязательная регистрация, не обязательно позволяет получать точные данные.

Идентификация сеанса доступа. Прежде чем можно будет выполнить какой-либо анализ использования, необходимо разделить данные на логические части, представляющие различные сеансы или транзакции. Сессия пользователя – это весь набор использованных ссылок на страницы, сделанных им во время одного посещения сайта. Проблема определения сеансов аналогична определению отдельных пользователей [4; 11].

Методом решения этой проблемы является распределение сеансов использования по времени, когда два последовательных вызова с одного и того же адреса считаются принадлежащими одному сеансу, если промежуток между этими вызовами не превысил заданный порог [8]. Второй метод заключается в поддержании «файлов cookie за сеанс» (данные хранятся на стороне пользователя *только* с первого посещения страницы до выключения браузера; анализ этих данных позволяет отличить одно посещение пользователя от другого).

Транзакции отличаются от сеанса пользователя тем, что они могут включать от одной до всех страниц маршрута пользователя для одного или нескольких сеансов в зависимости от указанного условия. Основной задачей разделения работы пользователя на транзакции является идентификация групп семантически близких вызовов одного пользователя, поэтому для разделения могут использоваться как операции разделения, так и операции слияния. Опишем три различных подхода к разделению на транзакции [4; 11].

Идентификация транзакций с учетом продолжительности посещений. Этот метод основан на том факте, что время, проведенное пользователем на странице, зависит от важности этой страницы для пользователя. Таким образом, если вы выберете определенное ограничение по времени, вы можете отделить страницы, которые интересны пользователю, от остальных. Этот метод предлагает форму для расчета такого интервала в зависимости от распределения посещений страниц с разными временными интервалами. Окончание транзакции – это первая из страниц, время посещения которых превысило выбранный порог, а начало – это первая страница после окончания предыдущей.

Идентификация транзакций методом максимальной глубины привязки. В этом случае новая транзакция начинается с первой прямой ссылки (перехода на страницу, которую пользователь еще не посещал). Окончанием транзакции является достижение наибольшей глубины, т. е. если пользователь вернулся на уже посещенную страницу [4; 11].

Разделение транзакций по принципу времени перекликается с методом распределения сеансов, все посещения делятся на части продолжительностью, не превышающей указанный порог. Этот метод может быть применен после одного из семантически ориентированных методов для отсечения вырожденных транзакций путем объединения транзакций, меньших чем пороговое значение.

Статистический анализ. Как правило, согласно журналу сервера, читается самая популярная страница, наибольшее количество посетителей за день, неделю, месяц, а также могут быть выделены страницы, вызвавшие наибольшее количество ошибок. Можно применить статистический анализ к журналу, который уже был очищен и разделен на транзакции. В этом случае функциональ-

ная полезность резко возрастает, становится возможным вычислять статистические данные по продолжительности пребывания на разных страницах или продолжительности транзакции. OLAP (On-Line Analytical Processing) используется в качестве интерфейса для анализа полученных данных [11].

После идентификации отдельной транзакции аналитик может применить один из методов извлечения знаний из схемы доступа: анализ путей, поиск ассоциативных правил и последовательностей изображений, кластеризацию или классификацию.

Для анализа пути используются различные типы графиков, поскольку график представляет некоторое отношение, определенное на странице (или другом объекте). Наиболее распространенным является построение графика, соответствующего физической структуре сервера, где страницы являются узлами, а ссылки между ними – направленными ветвями [4; 11].

Из-за того что такие базы данных транзакций содержат много информации, технологии поиска обычно фокусируются только на записях, к которым обращались по крайней мере определенное количество раз. Ознакомление с этими правилами для организаций электронной коммерции может помочь в разработке эффективного маркетинга. Эта информация также помогает в улучшении организации сетевого пространства [4; 11].

Поиск последовательностей выборок – это обнаружение связи между различными операциями, происходящими в течение одного временного интервала. В журналах транзакций сервера каждое посещение клиента записывается с определенным временным интервалом. Исследование временных соотношений между различными данными может иметь, например, следующие результаты: 30 % клиентов после посещения исследуемого сервера зарегистрировались на нем в течение 10 дней. Можно найти общие характеристики среди клиентов, которые обращались к одному и тому же файлу за один и тот же период времени, или интервал времени, в течение которого файл используется чаще.

Обнаружение правил классификации позволяет создать описание записей, принадлежащих к определенной группе, благодаря общности атрибутов. Это описание затем используется для классификации вновь добавленных записей. Кластерный анализ позволяет группировать клиентов или данные, которые имеют схожие характеристики. Кластеризация информации о клиентах с данными в журналах может позволить разработать и реализовать ряд маркетинговых стратегий [4; 11].

Кластеризация транзакций [11]. Основной областью применения кластерного анализа при анализе веб-использования является персонификация содержимого страницы. Пользователь присваивается одной из категорий, после чего информация, отображаемая для этого пользователя, соответствующим образом изменяется [6]. Другой традиционной областью применения кластеризации является поддержка принятия решений. При использовании кластеризации всегда необходимо решать две задачи: выбор метрики и алгоритма. Основной задачей при кластеризации транзакций является выбор метрик. По ряду причин классические евклидовы метрики неэффективны.

Можно попробовать сравнивать сеансы пользователей. Пусть все сессии будут в виде вектора одинаковой длины, где длина − общее количество анализируемых страниц. Значениями элементов будет Истина, если такая страница входит в сеансы, Ложь − если не входит. Используя подходящие методы кластеризации, при таком подходе можно добиться достаточно точных результатов (алгоритмы ROCK или CACTUS). Второй проблемой, сопряженной с выбором метрики, является нормализация транзакций. Часто приходится сравнивать между собой транзакции из двух-трех страниц и транзакции длиной свыше 25 переходов. Следует отметить, что проблема нормализации данных отпадает при применении некоторых специальных метрик.

Анализ полученных схем. OLAP является мощным инструментом для стратегического анализа БД. Показано, что анализ, требуемый при извлечении знаний из сети, сходен с проводимым в хранилищах данных. Хорошим примером применения OLAP в данной области является система WebLogMiner [5], включающая четыре уровня.

На первом − записи журналов очищаются и помещаются в реляционные таблицы. На втором − строится куб данных на основании выбранных атрибутов. В качестве атрибутов могут быть выбраны: пользователь, расположение пользователя, тип ресурса, время, затраченное на просмотр ресурса, дата, ответ сервера и т. д. На третьем − используется механизм OLAP для изучения полученных данных экспертами. Например, можно получить статистику по всем запросам, по запросам от одного домена или от одного типа браузера. Можно получать информацию по различным пользовательским сессиям или временным отрезкам. На четвертом уровне используются методы data mining для предсказания, классификации и нахождения интересных закономерностей. Этот этап может предоставлять информацию, которую не удалось обнаружить на предыдущем.

Визуализация является инструментом для облегчения понимания различного рода задач. Была разработана система WebViz для визуализации образцов сетевого доступа. Система использует парадигму веб-пути, при которой наборы записей в журнале используются для извлечения последовательных наборов прохождения сети. WebViz позволяет аналитику обработать часть сети, отфильтровав неподходящую информацию по различным критериям: по именам серверов или по локальным адресам страниц. Сеть представлена как направленный, цикличный граф, в котором узлам соответствуют страницы и ребрам − переходы пользователей по ссылкам со страницы на страницу [4; 11].

Выводы

На основе рассмотренной информации с использованием [1−15].

1. Информация, которая представляется и обрабатывается в компьютере, разделяется на процедурную и декларативную**.** Процедурная представлена в программах, декларативная информация – в данных, с которыми эти программы работают. По мере развития исследований в области ИС возникла концеп-

ция знаний, которые объединили в себе многие черты процедурной и декларативной информации. Машины, реализующие алгоритмы ИИ, называются основанными на знаниях, а подраздел теории ИИ, связанный с построением экспертных систем, – инженерией знаний.

2. Основные отличия данных от знаний: внутренняя интерпретируемость (информационная единица должна иметь имя, по которому ИС находит ее); структурируемость (рекурсивная вложимость одних информационных единиц в другие); связность (в базе между информационными единицами должно быть предусмотрено установление связей различного типа); семантическая метрика (на множестве информационных единиц имеются ассоциативные связи); активность (появление в базе фактов, описаний событий, установление связей).

3. Знания – это выявленные закономерности предметной области (принципы, связи, законы), позволяющие решать задачи в этой области. База знаний – это совокупность знаний, описанных с использованием выбранной формы их представления. Существуют два типа методов представления знаний (ПЗ): формальные модели ПЗ (логические и продукционные модели); неформальные (семантические сети, реляционные модели, фреймы).

4. Формальная теория $F = (A, V, W, R)$, определяющая некоторую аксиоматическую систему, характеризуется: наличием алфавита (словаря) – A; множеством синтаксических правил – V; множеством аксиом, лежащих в основе теории – W; множеством правил вывода – R.

5. При построении логических моделей представление знаний состоит в том, что вся информация рассматривается как совокупность фактов и утверждений, которые представляются как формулы в некоторой логике. Знания отображаются совокупностью таких формул, а получение новых знаний сводится к реализации процедур логического вывода. Исчисление высказываний и исчисление предикатов являются классическими примерами логических систем.

6. Продукционные модели сдержат продукционные правила, которые представить в следующем виде: $i : S; L; A \rightarrow B; Q$, где i − индивидуальный номер продукции; S − описание класса ситуаций, в котором данная структура может использоваться; L − условие, при котором продукция активизируется; $A \rightarrow B$ − ядро продукции, например: «ЕСЛИ A_1, A_2,,..., A_n, ТО В» (если условия от A_1 до A_n являются истиной, то В также истина); Q − постусловие продукционного правила, описывает операции и действия (процедуры), которые необходимо выполнить после выполнения B.

7. В основе сетевых моделей лежит конструкция, названная семантической сетью. Сетевые модели формально можно задать в виде $H = <I, C_1, C_2, ..., C_n, \Gamma>$. Здесь I есть множество информационных единиц; $C_1, C_2, ..., C_n$ – множество типов связей между информационными единицами. Отображение Γ задает между информационными единицами, входящими в I, связи из заданного набора типов связей. В зависимости от типов связей, используемых в модели, различают классифицирующие сети, функциональные сети и сценарии.

8. Фреймовая модель представляет собой систематизированную модель памяти человека и его сознания, важным элементом в которой является понятие

фрейма − структуры данных для представления некоторого концептуального объекта. Информация, относящаяся к этому фрейму, содержится в составляющих фрейма − слотах. Во фреймовых моделях фиксируется жесткая структура, которая называется протофреймом (фреймом-прототипом, или образцом), которая выглядит следующим образом: (Имя фрейма: имя слота 1(значение слота 1); имя слота 2 (значение слота 2);...имя слота *K* (значение слота *K*)).

9. Ввод знаний в БЗ осуществляется инженером по знаниям, который может получать знания от эксперта и преобразовывать знания в форму, удобную для дальнейшей обработки. Возможны три этапа при их взаимодействии: подготовительный (знакомство эксперта и аналитика); установление общего кода (определение главных понятий, взаимосвязи между ними, уровня детализации); гносеологический (происходит выяснение закономерностей предметной области, условия достоверности и истинности утверждений, структурирование за счет введения отношений).

10. При реализации системы приобретения знаний возможна следующая последовательность действий: интервью для определения области; формирование декларативном модели предметной области; протокольный анализ и выявление понятий предметной области для пополнения модели процедурными знаниями; проверка полноты модели, если понятия недостаточно описаны, интервью и протокольный анализ повторяются.

11. Извлечение знаний можно определить как нахождение и анализ полезной информации. Данная область деятельности подразделяется на две части: автоматический поиск информации в документах Сети − Web content mining и обнаружение и обработка информации, касающейся работы пользователей с сервером, − Web usage mining.

Вопросы для контроля

1. Дайте отличия данных от знаний.
2. Назовите модели знаний.
3. Охарактеризуйте формальные модели знаний.
4. Охарактеризуйте неформальные модели знаний.
5. Назовите достоинства и недостатки логических моделей.
6. Назовите достоинства и недостатки продукционной модели.
7. Назовите достоинства и недостатки семантических сетей.
8. Назовите достоинства и недостатки фреймов.
9. Поясните особенности нечеткой логики.
10. Определите особенности вероятностной логики.
11. Охарактеризуйте ввод знаний в БЗ.
12. Поясните извлечение знаний из Интернета
13. Определите элементы обработки знаний в Интернете.

Литература, используемая в главе

1. Искусственный интеллект : справ. В 3 т. Т. 2. : Модели и методы / под ред. Д. А. Поспелова. – М. : Радио и связь, 1990. – 304 с. ; Т. 3. : Программные и аппаратные средства / под ред. Д. А. Поспелова. – М. : Радио и связь, 1990. – 363 с.

2. Рассел, С. Искусственный интеллект, современный подход / С. Рассел, П. Норвинг. – М. : Вильямс, 2006. – 1408 с.

3. Вишняков, В. А. Интеллектуальные системы в управлении / В. А. Вишняков. – Минск : Изд-во МИУ, 2010. – 364 с.

4. Российский сервер «Информационные технологии в управлении» [Электронный ресурс]. – Режим доступа: http://www.citforum.ru/internet/webservers/mining.shtml. – Дата доступа: 15.12.2022.

5. Информатика: учебник. / под ред. проф. Н. В. Макаровой. – 3-е, изд. перераб. – М. : Финансы и статистика, 2009. – 768 с.

6. Логическое программирование. – М. : Мир, 1988. – 368 с.

7. Адилов, Р. М. Системы искусственного интеллекта / Р. М. Адилов. – Пенза : ПГТУ. – 95 с.

8. Остроух, А. В. Основы построения систем искусственного интеллекта для промышленных и строительных предприятий / А. В. Остроух. – М. : Технополиграфцентр. – 280 с.

9. Щербина, А. Основы извлечения данных из Интернета / А. Щербина // Открытие системы. – 2003. – № 4. – С. 23–28.

10. Знания и модели их представления [Электронный ресурс]. – Режим доступа: https://megaobuchalka.ru/9/7967.html. – Дата доступа: 15.12.2021.

11. Метаданные [Электронный ресурс]. – Режим доступа: http://www.citforum.ru/internet/webservers/mining.shtml. – Дата доступа: 15.12.2021.

12. Представление знаний в Интернете. Режим доступа: http://bigc.ru/publications/other/km/predst_znan_v_intel_shemah.php. – Дата доступа: 15.12.2021.

13. Батенькина, О. В. Создание системы автоматизации конструкторско-технологической подготовки производства : дис. канд. : 05.12.13 / О. В. Батенькина. – Омск, 2005. – 189 л.

14. Сводная коллекция ЭБС [Электронный ресурс]. – Режим доступа: http://e.lanbook.com/books/element.php?pl1_id=63569. – Дата доступа: 15.01.2023.

3 ЯЗЫКИ ПРОГРАММИРОВАНИЯ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

3.1 Классификация языков и стилей программирования

Все языки программирования можно разделить на процедурные и декларативные [7]. Большинство используемых в прошлом языков программирования (C, Pascal и т. д.) относятся к процедурным языкам. Основными классами декларативных языков являются функциональные (Lisp, Logo, APL и т. д.) и логические (Prolog, Planer, Coniver и т. д.) языки. На практике языки программирования не являются чисто процедурными, функциональными или логическими, но содержат особенности языков различных типов. На процедурном языке можно написать функциональную программу или ее часть, и наоборот. Естественно, разные языки в разной степени поддерживают разные стили [3; 5; 7; 10].

Программа на процедурном языке состоит из последовательности утверждений и предложений, которые управляют последовательностью их выполнения. Основными операторами являются операторы управления, присваивания, передачи, ввода-вывода и специальные для организации циклов. Они могут быть использованы для создания фрагментов программ и подпрограмм. Процедурное программирование основано на введении значения переменной, выполнении над ней действия и сохранении нового значения с помощью оператора присваивания, пока не будет получено желаемое конечное значение (и отображено).

Логическое программирование – один из подходов к интеллектуальному описанию задач, в котором в качестве языка высокого уровня используется логика предикатов первого порядка в виде дизъюнктов Хорна. Логика предикатов первого порядка − это универсальный абстрактный язык, предназначенный для представления знаний и решения проблем.

Логическое программирование позволяет программисту описать ситуацию с помощью формул логики предикатов, а затем, чтобы сделать выводы из этих формул, применить автоматический решатель проблем (строгую процедуру). При использовании этого языка основное внимание уделяется описанию структуры прикладной задачи, а не разработке предписаний для компьютера о том, что он должен делать. Другие концепции информатики из таких областей, как теория реляционных баз данных, разработка программного обеспечения и представление знаний, также могут быть описаны (и реализованы) с помощью логических программ [2; 8].

Функциональное программирование. Функциональная программа состоит из набора определений функций. Функции, в свою очередь, являются вызовами других функций и предложениями, которые управляют последовательностью вызовов. Вычисления начинаются с вызова некоторой функции, которая, в свою очередь, вызывает функции, включенные в ее определение, в соответствии с иерархией определений и структурой условных предложений. Функции часто прямо или косвенно называют сами себя [2; 7].

Каждый вызов возвращает некоторое значение вызвавшей его функции, вычисление которой продолжается после этого; этот процесс повторяется до

тех пор, пока функция, запустившая вычисления, не вернет пользователю конечный результат. Функциональное программирование не распознает назначения и передачи управления. Ветвление вычислений основано на механизме обработки аргументов условного предложения. Повторяющиеся вычисления выполняются с помощью рекурсии, которая является основным средством функционального программирования.

Специфика ИС заключается в их сосредоточенности на описании и обработке знаний в предметной области. Язык программирования определяет не только программное обеспечение, но и дизайнерские и технологические решения. На этапах разработки ИС используются передовые средства автоматизации программирования. Такие среды характеризуются быстрой компиляцией исходных текстов, передовыми средствами графического редактирования, интерактивными средствами отладки и библиотеками отдельных компонентов. Примером реализации таких сред является Visual Prolog. Эта реализация имеет многооконный интерфейс с иерархическими меню. Поддерживается несколько режимов работы: редактирование − Edit, компиляция − Compile, запуск − Run, работа с файловой системой − Files, определение параметров перевода – Options, настройка параметров − Setup.

На первом этапе развития искусственного интеллекта (в конце 50-х – начале 60-х годов) не существовало языков и систем, ориентированных конкретно на области знаний. Появившиеся к тому времени универсальные языки программирования казались подходящим инструментом для создания любых (в том числе интеллектуальных) систем, поскольку в этих языках можно выделить декларативные и процедурные компоненты. Казалось, что на этой основе можно интерпретировать любые модели и системы представления знаний. Но сложность и запутанность интерпретаций оказались велики, так что прикладные системы были недоступны для внедрения.

Исследования показали, что производительность программиста остается постоянной независимо от уровня инструментального языка, на котором он работает, а соотношение между длиной исходного кода и результирующих программ составляет примерно 1:10. Использование адекватного инструментального языка повышает производительность разработчика системы на порядок. Языковые платформы, предназначенные для программирования интеллектуальных систем, содержат иерархические (многоуровневые) трансляторы и повышают производительность труда. Все это подтверждает важность использования адекватных инструментов [12; 14].

Язык Lisp был разработан в Стэнфорде под руководством Дж. Маккарти в начале 60-х годов. Согласно первоначальным планам он должен был включать наряду со всеми возможностями Fortran инструменты для работы с матрицами, указателями и структурами из указателей и т. д. Но средств на такой проект не хватило. Окончательно сформировались принципы, лежащие в основе языка Lisp: использование единого представления списка для программ и данных; использование выражений для определения функций; заключенный в квадратные скобки синтаксис языка [7].

После создания мощных систем Lisp в начале 70-х годов: Maklisp, Interlisp – попытки создать языки искусственного интеллекта, отличные от Lisp, но на той же основе, сходят на нет. Дальнейшее развитие языка идет, с одной стороны, по пути его стандартизации (Standard-Lisp, French-Lisp, Common Lisp), а с другой − в направлении создания концептуально новых языков для представления и манипулирования знаниями в среде Lisp. В настоящее время Lisp реализован на всех классах компьютеров, начиная с персональных компьютеров и заканчивая высокопроизводительными вычислительными системами. Lisp – не единственный язык, используемый для решения задач искусственного интеллекта. Уже в середине 60-х годов разрабатывались языки, предлагавшие другие концептуальные рамки. Наиболее важными из них в области обработки символической информации являются Снобол (SNOBBALL) и Refal [3; 7].

Язык Снобол предназначен для обработки строк, в его рамках появилась концепция поиска по шаблону и была реализована. Язык был одной из первых практических реализаций разработанной производственной системы. Самой известной и интересной версией этого языка является Snobol-4, в котором техника основных конструкций (сэмплов) и работы с ними значительно опередила потребности практики. Концепции SNOBBALL повлияли на Lisp и другие языки программирования для решения задач искусственного интеллекта.

Язык Рефал − это алгоритмический язык рекурсивных функций. Он был создан Турчиным как метаязык, предназначенный для описания различных, в том числе алгоритмических, языков и различных типов обработки таких языков. Также подразумевалось использование Рефала в качестве метаязыка над самим языком. Для пользователя это язык для обработки символьной информации. Поэтому, помимо описания семантики алгоритмических языков, он нашел и другие применения. Это выполнение аналитических вычислений в теоретической физике и прикладной математике, интерпретация и компиляция языков программирования, доказательство теорем, моделирование целенаправленного поведения и задачи искусственного интеллекта. Общим для всех этих приложений являются сложные преобразования над объектами, определенными на некоторых формализованных языках.

Язык Refal основан на концепции рекурсивной функции, определенной на наборе произвольных символьных выражений. Базовая структура данных этого языка – списки, но не просто связанные, как в Lisp, а двунаправленные. Обработка символов ближе к производственной парадигме. В то же время активно используется концепция поиска паттернов, характерная для Снобола. Программа, написанная на Refal, определяет определенный набор функций, каждая из которых имеет один аргумент. Вызов функции заключен в функциональные скобки.

Во многих случаях возникает необходимость вызывать программы, написанные на других языках, из программ, написанных на Refal. Это просто, так как с точки зрения Refal первичные функции (функции, описанные не в Refal, но которые могут быть вызваны из программ, написанных на этом языке) − это всего лишь некоторые функции, внешние по отношению к этой программе, поэтому при вызове функции вы можете даже не знать, что это основная функция.

Семантика Рефал-программы описывается в терминах абстрактной рефал-машины, которая имеет поле памяти и поле обзора. Исходная программа помещается в поле памяти машины Refal, а данные, которые будут обрабатываться с ее помощью, помещаются в поле обзора, т. е. перед началом работы в поле памяти вводится описание набора функций, а в поле обзора вводится выражение, подлежащее обработке.

Удобно разбить Рефал-программу на части, которые могут обрабатываться компилятором Рефала независимо и являются модулями. Результат компиляции исходного модуля на Рефале представляет собой объектный модуль, который Рефал-программой должен быть объединен с другими модулями, полученными компиляцией. Затем объединение откомпилированных модулей с Рефала или/и других языков выполняется с помощью редактора связей. Рефал вобрал в себя лучшие черты интересных языков обработки символьной информации 60-х годов. Иногда Рефал используется для автоматизации построения трансляторов, систем аналитических преобразований, а также, подобно Лиспу, в качестве инструментальной среды для реализации языков представления знаний [12].

3.2 Язык логического программирования Пролог

Пролог. В 70-х годах во Франции появился язык для ИС, который конкурировал с Lisp в реализации систем, ориентированных на знания, − Prolog. Этот язык не предоставляет новых сверхмощных инструментов программирования по сравнению с Lisp, но поддерживает иную модель организации вычислений. Его привлекательность заключается в том, что точно так же, как Lisp скрывал от программиста устройство компьютерной памяти, Prolog позволял ему также не заботиться о потоке управления в программе [3; 7; 8].

Prolog − это язык, который был разработан в университете Марселя в 1971 году. Но популярность он начал набирать только в начале 80-х годов. Это связано с двумя обстоятельствами: во-первых, логическая основа этого языка была основана на автоматическом выводе, во-вторых, в японском проекте вычислительных систем пятого поколения он был выбран в качестве основы для одного из основных компонентов − машины вывода.

Язык Prolog основан на ограниченном наборе механизмов, включая сопоставление с образцом, древовидное представление структур данных и автоматический возврат. Пролог особенно хорошо адаптирован для решения задач, связанных с объектами и отношениями между ними [2; 8].

Prolog обладает мощными инструментами, которые позволяют извлекать информацию из баз данных, а используемые в нем методы поиска данных принципиально отличаются от традиционных. Мощь и гибкость БД Prolog, простота их расширения и модификации делают этот язык очень удобным для коммерческих приложений. Он успешно используется в таких областях, как реляционные базы данных; автоматическое решение задач; понимание естественного языка; реализация языков программирования; представление знаний; экспертные системы и другие задачи искусственного интеллекта [7; 8].

Теоретической основой Пролога является исчисление предикатов. Пролог обладает рядом свойств, которыми не обладают традиционные языки программирования. Такие свойства включают механизм поиска и возврата выходных данных и встроенный механизм сопоставления с образцом. Пролог отличается единообразием программ и данных. Это просто разные точки зрения на объекты Пролога. В языке отсутствуют указатели, операторы присваивания и безусловный переход. Естественным методом программирования является рекурсия [8].

Программа на Прологе состоит из двух частей: БД (набор аксиом) и последовательности целевых утверждений, описывающих отрицание доказываемой теоремы в совокупности. Основное фундаментальное различие между интерпретацией программы Пролога и процедурой доказательства теоремы в исчислении предикатов первого порядка заключается в том, что аксиомы в базе данных упорядочены и порядок их последовательности очень важен, поскольку сам алгоритм, реализованный программой Пролога, основан на этом. Другим существенным ограничением Пролога является то, что в качестве логических аксиом используются формулы ограниченного класса − так называемые дизъюнкты Хорна. Однако при решении многих практических задач этого достаточно для адекватного представления знаний [8].

Поиск формул, полезных для доказательства, является комбинаторной задачей, и с увеличением числа аксиом количество шагов вывода растет катастрофически быстро. Поэтому в реальных системах используются всевозможные стратегии, ограничивающие поиск вслепую. В языке Пролог реализована стратегия линейного разрешения, предполагающая использование отрицания теоремы или ее «потомка» в качестве одной из сравниваемых формул на каждом шаге и одной из аксиом в качестве другой. В то же время выбор той или иной аксиомы для сравнения может сразу или после нескольких шагов завести в «тупик». Это заставляет вернуться к той точке вычислений, когда был сделан выбор, чтобы выбрать новую альтернативу и т. д. [8].

Порядок просмотра альтернативных аксиом не является произвольным − он устанавливается программистом, размещающим аксиомы в базе данных в определенном порядке. Кроме того, Пролог предоставляет довольно удобные «встроенные» средства для запрета возврата в определенную точку в зависимости от выполнения определенных условий. Таким образом, процесс доказательства в Прологе проще и целенаправленнее, чем в классическом методе резолюций. Значение программы на языке Prolog может быть понято либо с точки зрения декларативного подхода, либо с точки зрения процедурного подхода [8].

Декларативный смысл программы определяет, является ли данная цель истинной (достижимой) и, если да, то при каких значениях переменных она достигается. Он подчеркивает статичное существование отношений. Порядок подцелей в правиле не влияет на декларативный смысл этого правила. Декларативная модель ближе к семантике логики предикатов, что делает Пролог эффективным языком для представления знаний. В декларативной модели невозможно адекватно представить те фразы, в которых важен порядок подцелей. Чтобы объяснить значение фраз такого рода, необходимо использовать процедурную модель.

В процедурной интерпретации программы Пролог определяются не только логические связи между главой предложения и целями в его теле, но и порядок, в котором обрабатываются эти цели. Но процедурная модель не подходит для уточнения значения фраз, которые вызывают побочные эффекты управления, такие как остановка выполнения запроса или удаление фразы из программы [8]. Для решения реальных задач ИИ необходимы машины, скорость которых должна намного превышать скорость обычных компьютеров, а это возможно только в параллельных системах. Поэтому последовательные реализации следует рассматривать как рабочие станции для создания программного обеспечения для будущих высокопроизводительных параллельных систем, способных выполнять сотни миллиардов логических выводов в секунду. Существуют десятки моделей параллельного выполнения логических программ в целом и программ Пролог в частности. Значительное внимание уделяется более современным схемам организации параллельных вычислений − потоковым моделям. Модели параллельного выполнения учитывают традиционный Prolog и присущие ему источники параллелизма.

На эффективность Пролога в значительной степени влияют ограниченные ресурсы во времени и пространстве. Это связано с неспособностью традиционной архитектуры вычислительных машин реализовать Пролог-метод выполнения программ, предусматривающий достижение целей из определенного списка. Вызовет ли это трудности при практическом применении, зависит от поставленной задачи. Фактор времени практически не имеет значения, если программа Prolog, запускаемая несколько раз в день, занимает одну секунду, а соответствующая программа на другом языке − 0,1 секунды. Но разница в эффективности становится существенной, если для этих двух программ требуется 50 и 5 минут соответственно.

Во многих областях применения Prolog это может значительно сократить время разработки программы. Программы Prolog легче писать, понимать и отлаживать, чем программы, написанные на традиционных языках, т. е. язык Prolog привлекателен своей простотой. Программа легко читается, что является фактором, способствующим повышению производительности при программировании и повышению удобства обслуживания программы [8].

Поскольку Пролог основан на фразах описания, исходный код программ Prolog значительно меньше подвержен влиянию машинно-зависимых функций, чем исходные тексты программ, написанных на других языках. Кроме того, существует тенденция к единообразию в разных версиях языка Prolog, так что программа, написанная для одной версии, может быть легко преобразована в программу для другой версии этого языка. Кроме того, Пролог прост в освоении.

3.3 Структура программы на Visual Прологе

Visual Prolog. Начиная с 1996 г. существуют версии *Visual Prolog.* Программа на Visual Прологе включает четыре основных раздела. Это разделы выражений − clauses, предикатов − predicates, доменов − domains и цели − goal. Раздел clauses − это ядро программы; в нем размещаются факты и правила, которыми оперирует Пролог. В разделе predicates объявляются предикаты пользователя. В разделе domains описываются любые используемые домены (типы аргументов пользователя), которые не являются стандартными доменами Пролога. В разделе goal помещается встроенная (внутренняя) цель, когда программисту требуется, чтобы программа могла выполняться вне среды Visual Пролога и независимо от нее [3; 7; 8].

Раздел clauses. В нем программист размещает включаемые в программу факты и правила. Выражения, относящиеся к определенному предикату, должны размещаться в разделе clauses. Если выражение не является частью, ведущей к решению логического пути, то Пролог возвращается к установленному указателю и ищет другое соответствие. Такой процесс называется поиском с возвратом (backtracking) [8].

Раздел predicates. Если программист определяет в разделе clauses свой собственный предикат, то он должен объявить его в разделе predicates. В разделе predicates программы перечисляется каждый предикат со своими аргументами.

Раздел domains. Они дают возможность присваивать различным видам информации отличные имена. В программе на Прологе объекты в отношении (аргументы предиката) принадлежат доменам; это могут быть домены стандартные или специальные, определяемые программистами [7].

Раздел goal. Задаваемые в этом разделе цели называются внутренними целями, поскольку они являются частью исходного текста программы и компилируются наряду со всеми другими ее частями. Содержание раздела goal аналогично правилу, это список подцелей. Но между разделом goal и правилом есть два отличия: после ключевого слова goal не следует знак :- (если); при запуске программы на выполнение Пролога отрабатывает цель автоматически.

Раздел базы данных объявляется с помощью ключевого слова database, куда включаются объявления фактов, предназначенных для организации динамической базы данных (БД).

Раздел constants. Рассмотрим следующий пример [3; 8]:

```
constants
нуль = 0
один = 1
два = 2
сотня = (10*(10-1)+10)
пи = 3.141592653
красный = 4
```

Элементы программирования. Рассмотрим, например, следующие трансформации предложений на естественном языке в запись на языке логики предикатов [3]:

Быстрый автомобиль доставляет_удовольствие − удовольствие (быстрый_автомобиль).

Если Пете нравится автомобиль − если нравится (пете, Автомобиль) автомобиль доставляет удовольствие − доставляет_удовольствие (Автомобиль).

Предложения: факты и правила. Программист на Прологе определяет «объекты» и «отношения», а затем описывает «правила», в которых указывается, когда эти отношения являются истинными, например:

Лене нравятся собаки, если собаки привлекательны.

Символьное имя отношения называется именем предиката. Объекты, которые связывает предикат, − это его аргументы; в факте «нравится (лена, собаки)» отношение «нравится» − предикат, а объекты «лена» и «собаки» − аргументы.

Факты. Те же факты, изложенные с использованием синтаксиса Пролога:

нравится (лена, ира).
нравится (володя, маша).
нравится (лена, собаки).

Правила. Некоторые правила, касающиеся отношения «нравится» [3]:

Ире нравится все, что нравится Лене.
Алле нравится все, что зеленое.

Чтобы записать эти правила на Прологе, необходимо изменить синтаксис, например:

нравится (ира, Нечто) если нравится (лена, Нечто).
нравится (алла, Нечто) если зеленый(Нечто).

В этом примере объект «Нечто» − переменная, поэтому его имя начинается с прописной буквы (все переменные в Прологе начинаются с прописной буквы), а объекты «ира», «лена», «алла» − константы (все константы в Прологе начинаются со строчной буквы).

Запросы. На естественном языке вопрос может выглядеть так:
Нравится ли Лене Ира? В случае же использования синтаксиса Пролога этот вопрос принял бы вид:

нравится (лена, ира).

На этот вопрос Пролог ответил бы «Да», потому что Прологу известен факт, позволяющий дать на этот вопрос положительный ответ. Более сложный и обобщенный вопрос на естественном языке можно сформулировать так [3]:

Что нравится Лене?

На Прологе мы имели бы в этом случае:

нравится(лена, Что).

Это объясняется тем, что «лена» представляет собой константу (известное значение), а «Что» − переменную. На запрос относительно того, что нравится Лене, Пролог даст ответ:

Что=ира
Что=собаки
2 решения

Это потому, что Прологу известны следующие факты:
нравится (лена, ира).
нравится (лена, собаки).

Переменные. Предположим, что у нас имеются следующие факты :
нравится (елена, теннис).
нравится (борис, теннис).

В запросе переменные можно использовать для того, чтобы спросить у системы Пролога, кому нравится теннис.
нравится(Человек, теннис).

Пролог отвечает
Человек=елена
Человек=борис
2 solution (решения)

Предположим имеются следующие факты:
родитель (николай, егор)
родитель (мария, егор)
родитель (егор, тамара)

На запрос
родитель(Родитель,)
Пролог отвечает :
Родитель=николай
Родитель=мария
Родитель=егор
3 solution

Цели (внутренние запросы). Цели могут быть простыми, например:
нравится (елена, плавание).
нравится (николай, Что).
или более сложными:
нравится (Человек, чтение), нравится (Человек, плавание),
т. е. найти человека, которому нравится и чтение, и плавание.

Комментарии. Многострочные комментарии должны начинаться с символов /*, а заканчиваться символами */. Для указания однострочного комментария можно использовать эти же символы или воспользоваться символом % как началом комментария.
/* Это пример комментария */
% Это также комментарий

Простейшие программы. Пусть заданы факты о книгах, их авторах и количестве страниц. Необходимо составить правило о том, какой роман можно считать большим, и реализовать это на Прологе. Ниже приводится один из вариантов решения поставленной задачи [3; 8].

```
 /Пример 1, демонстрирующий работу с фактами, правилами и переменными */
domains
% Описание доменов пользователя
заглавие, автор = symbol
страниц = integer
predicates
% Описание предикатов пользователя
```

```
книга(заглавие, страниц)
написал(автор, заглавие)
большой_роман(заглавие)
clauses
% Факты
написал(горький, «МАТЬ»).
написал(шолохов, «ТИХИЙ ДОН»).
книга(«ТИХИЙ ДОН»,1250).
книга(«МАТЬ», 310).
/* Правило, определяющее,
какая книга считается большим романом */
большой_роман(Заглавие) :-
написал(_, Заглавие), %Автор в этом случае безразличен
книга(Заглавие, Страниц),
Страниц > 300.
Например,
Goal:написал(Кто,Что)
```

При ответе на этот вопрос производится унификация значений переменных Кто с обьектами «автор» и Что с обьектами «заглавие». В результате выдается ответ:

```
Кто = горький Что = «МАТЬ»
Кто = шолохов Что = «ТИХИЙ ДОН»
2 Solutions
```

Или такой запрос:

```
Goal:большой_роман(Заглавие)
```

В нашей базе данных хранятся два романа, которые можно считать большими, поскольку их объем превышает 300 страниц. Поэтому выдается следующее решение:

```
Заглавие = «МАТЬ»
Заглавии = «ТИХИЙ ДОН»
2 Solutions
```

Организация циклов. Бесконечный цикл – организуется предикатом fail. В Прологе предусмотрен режим организации цикла через стандартный предикат fail (неудача), который позволяет получать альтернативные решения.

Рекурсия. Другим способом организации циклов является рекурсия, т. е. вызов предикатами самих себя. Рассмотрим вычисление значения N! [9].

```
/* пример 4 − вычисление факториала с использованием обычной рекурсии */
trace /* этот стандартный предикат поможет понять логику */
predicates
/* Значение факториала вещественное, чтобы избежать переполнения*/
factorial(integer, real)
clauses
/* если N=1, то значение факториала равно 1 */
factorial(1, 1) :- !. % условие останова рекурсии
/* Во всех других случаях надо найти факториал N-1,
а затем умножить его на N */
factorial(N, FactN) :-
Происходит связывание переменных N и NN */NN = N-1,
/* рекурсивный вызов factorial с НОВЫМИ параметрами */
```

```
factorial(NN, FactNN),
FactN = N*FactNN.
```
Goal:factorial(3, F),
то в режиме трассировки будет получено решение F=6.

В качестве примера рассмотрим программу, содержащую базу данных родственных отношений и попробуем использовать различные запросы. Создадим файл parents.pro, в котором создадим базу родственные отношений. Опишем тип имени в *секции domains* [8]:
```
domains
name = symbol
```
Опишем отношение базы данных, задающее родство двух имен:
```
Database
parent(name, name)
```
Добавим элементы во внутреннюю базу данных в *секции clauses*:
```
Clauses
parent(marina, ira).
parent(elena, ivan).
parent(nikolay, ira).
parent(marina, sasha).
parent(sergei, ivan).
```
Первый запрос, который мы реализуем, должен проверять, что *"Марина является родителем Саши"*. Запросы описываются в *секции goal*:
```
Goal
parent(marina, sasha).
```

Запускаем программу, выводится «yes», т. к. утверждение является верным для описанных в базе фактов. Следующее утверждение, которое проверим *«Алексей является родителем Саши»* – ложно, т. к. в базе нет факта.
Запишем этот запрос в секции *goal* :
```
Goal
parent(aleksej, sasha).
```

Получим ответ «No».
В запросе *«Кто является ребенком Николая»* нужно использовать анонимную переменную (ее имя начинается с большой буквы), т. к. имя ребенка неизвестно. Интерпретатор попробует подобрать все возможные решения, сопоставив эту переменную с каждым значением базы данных.
```
Goal
parent(nikolay, X).
```

В результате он сообщит нам, что ребенок Николая – Ира:
```
X = ira
1 Solushion
```
Однако в запросе *«Кто родители Ивана»*
```
parent(X, Ivan).
```
получим 2 решения:
```
X = elena
X = sergey
```

2 Solushions

При определении всех родителей и их детей используем две анонимные переменные в запросе (т. к. ни конкретные родители, ни дети не известны):

```
Goal
parent(Parent, Kid).
```

В результате будет выведено все содержимое базы данных:

```
Parent = marina, Kid = ira
Parent = elena, Kid = ivan
Parent = nikolay, Kid = ira
Parent = marina, Kid = sasha
Parent = sergei, Kid = ivan
5 Solushions
```

3.4 Язык ЛИСП и его диалекты

Лисп – язык низкого уровня, его можно рассматривать как ассемблер, ориентированный на работу со структурами списков. Поэтому на протяжении всего существования языка было предпринято много попыток улучшить его путем введения дополнительных базовых примитивов и управляющих структур. Но все эти изменения не становились самостоятельными языками. В своих изданиях Lisp ассимилировал все ценные изобретения своих конкурентов [2; 4; 12].

Маклисп. В дополнение к символьной обработке Maclisp широко использовался в традиционных численных вычислениях, используемых, например, при обработке речи и изображений. Помимо исследователей искусственного интеллекта и разработчиков алгебраической системы Maxim на Маклисп также повлияла работа групп Массачусетского технологического института по робототехнике, обработке речи и изображений. Исходя из требований этих областей, в Maclisp были включены новые математические типы данных, такие как матричная и битовая обработка, а также широкий спектр арифметических функций и инструментов. Пожалуй, наиболее важным из них является возможность вычислений с неограниченной точностью, основанных на алгоритмах, созданных Кнутом (1969) [2; 4; 7].

Maklisp также была первой системой Lisp, для которой был создан хороший переводчик. Переводчик генерирует машинную программу в виде списков. Машинный код в виде списка легко обрабатывается, и результирующий код для числовых задач был получен более эффективно, чем лучшие переводчики Fortran. Именно так в язык попали макросы для чтения и таблицы для чтения, которые позволяют легко изменять и расширять структуру языка. Таким же образом структуры управления, механизмы обработки прерываний и ошибок, а также использование управляющих символов возникли из требований к программам и окружению, был создан и интегрирован в систему экранный редактор, появилось управление и взаимодействие параллельных процессов [2].

Разработчики Maklisp сосредоточили свое основное внимание на эффективности. Этому служат инструкции, разъясняющие способы обработки аргу-

ментов функции, а также позволяющие избежать вмешательства программиста во внутренние механизмы системы. Благодаря этим мерам скорость Маклиспа в 1,5–2,5 раза выше, чем Интерлиспа [2; 7].

В общей сложности в Maklisp используется около 400 функций. Самым большим недостатком системы является то, что она никогда не была должным образом задокументирована. Документация по этой системе разбросана по различным отчетам и руководствам. Maklisp была исследовательской системой и не предназначалась для обучения и промышленного использования.

МуЛисп. Интерпретатор МуЛисп-85, разработанный для персональных компьютеров, является удачным вариантом реализации диалекта языка, включающим относительно ограниченный набор базовых функций (около 260) и, как следствие, оказался более легким в освоении [7].

По сравнению с Common Lisp диалект МуЛисп не имеет такого широкого спектра доступных типов данных. Он обеспечивает работу только с двумя типами числовой информации: целыми числами с любым основанием и рациональными числами. В диалекте отсутствуют инструменты для работы со структурами, массивами, потоками и другими типами данных, указанная реализация языка Lisp имеет одно существенное преимущество − возможность пополнять базовый набор функций путем подключения подпрограмм, написанных на языке ассемблера, что позволило повысить гибкость использования интерпретатора и эффективность прикладного программного обеспечения, созданного на его основе. Возможность такого пополнения отсутствует в большинстве других Lisp-систем, которые в являются закрытыми программными продуктами [2; 4].

Среди других особенностей системы можно указать на реализацию специального механизма, который позволяет не беспокоиться о присвоении начальных значений буквальным атомам, которые получают начальное значение, равное «печатному» имени самого атома. Еще одной особенностью диалекта является возможность использования новой синтаксической конструкции «embedded cond», которая значительно сокращает тексты описаний пользовательских функций и используется при написании тел функций и лямбда-выражений [12].

Набор базовых функций интерпретатора MuLISP включает в себя ряд функций, которые обеспечивают доступ практически ко всем функциям компьютерной ОС через соответствующие прерывания [2; 4]. Наконец, указанная система Lisp снабжена библиотеками функций Lisp, которые дополняют базовый набор функций, доступных на диалектах Common Lisp и Interlisp, что облегчает решение проблемы переносимости исходных текстов программных модулей, а также библиотек, позволяющих манипулировать окнами на экране дисплея и обрабатывать управляющие действия с помощью устройства типа мыши. Набор дополнительного программного обеспечения для переводчика включает интерактивный текстовый редактор и простую систему обучения, написанную на диалекте языка муЛИСП.

Интерлисп [2]. Interlisp появился в 1972 году из Lisp. К 1978 году, когда было опубликовано описание Interlisp, язык и система уже достаточно стабили-

зировались. Interlisp больше не был языком в том же смысле, что Maclisp или другие системы Lisp или обычные традиционные системы программирования. Это была интегрированная среда программирования, которая включала в себя множество различных вспомогательных инструментов. Interlisp стал классическим примером хорошо разработанного программного обеспечения и инструментов в системах разделения времени [2].

Этот диалект наряду с Common Lisp является одним из наиболее распространенных, обладает достаточно развитым аппаратом для представления различных структур данных и манипулирования ими, включая массивы. Среди общих черт этого варианта языка следует отметить использование нетрадиционных названий для обозначения встроенных функций, что иногда затрудняет перенос готовых программных продуктов на другие диалекты и другие компьютеры [4].

В 1974 году Xerox начала разработку персональной рабочей станции Lisp под названием alto для Interlisp. При реализации Lisp для alto впервые была использована микропрограммируемая система команд и мини-компьютеров, разработанная специально для языка Lisp, способная интерпретировать программы Lisp с более высокой производительностью, чем мэйнфреймы. На основе этой машины alto впоследствии были разработаны машины Lisp серии 1100 компании Hegoh [2].

На основе версии Interlisp, которая работала в системе разделения времени, была создана восходящая совместимая версия Interlisp-de Lisp, используемая на машинах Lisp серии 1100. Его пользовательский интерфейс включал многооконное взаимодействие, графику с высоким разрешением, инструменты выбора меню и мышь, а также инспектор структуры данных, ориентированный на экран. Идея разделения экрана на множество независимых окон родилась в xlg. Алан Кей уже в конце 60-х годов предложил такую идею подхода к компьютерам будущего и интерфейсу между человеком и машиной. Работа xlg привела к созданию в 70-х годах языка программирования smolltalk и объектного программирования [2].

При создании Интерлиспа работа была проведена очень тщательно. Система хорошо документирована, и новые версии совместимы с более ранними. Таким образом, преимуществом системы стало постоянно пополняющееся большое количество портативного программного обеспечения. С другой стороны, ограничение системы старым диалектом, зафиксированное уже в конце 70-х годов, сделало систему несколько устаревшей и трудной для расширения. В Interlisp, помимо прочего, отсутствуют иерархические типы данных, объекты и замыкания. Кроме того, он основан на динамическом связывании, в то время как новые версии Lisp являются статическими. Однако новая версия происходит от Interlisp − Common Lisp (1986). Для программирования на более высоком уровне Interlisp разработала инструменты, в которых объекты уже присутствовали [2].

Interlisp − это настолько закрытая система, что доступна только ее переведенная версия в машинных кодах. В некоторых других системах, таких как,

например, Zetalisp, поддерживается версия Lisp на исходном языке, которая доступна пользователю и может быть им модифицирована. Разработка закрытых систем, подобных Interlisp, связана с ресурсами, доступными лабораториям, которые их создали. Interlisp использует более 500 функций и большое количество системных имен и флажков, с помощью которых вы можете настраивать систему. Interlisp реализован в системе разделения времени на многих больших компьютерах [2].

В Interlisp основное внимание было уделено удобству системы для пользователя. Главный принцип разработчиков этого диалекта заключается в том, что все, что может происходить в системе, естественно, должно быть выражено в терминах ее входного языка. Таким образом, все доступно программисту в Interlisp. Он может переопределять любые функции, включая встроенные; устанавливать и переопределять ответы на ошибки; работать непосредственно с уровня входного языка с внутренними структурами интерпретатора и т. д. В то же время система сохраняет свою целостность и работоспособность.

ФрансЛисп. МакЛисп стал основой для многих новых диалектов языка Lisp, первым из которых был Франс Лисп. Эта версия Lisp названа в честь известного венгерского композитора. Основным мотивом разработки France Lisp было желание получить современную систему Lisp для новых машин vax, чтобы они могли использовать систему Maxim и другое программное обеспечение Lisp. Франс Лисп в довольно большой степени напоминает Маклисп, на котором изначально была реализована Максима. Однако в диалекте отсутствуют некоторые устаревшие функции Maklisp и содержатся новые системные идеи, заимствованные в то время в машинах mit Lisp для Zetalisp [2].

Новый диалект был реализован в университете Беркли на компьютере VAX 780/11 на языке Си под управлением системы Unix. France Lisp довольно широко используется как в unix, так и в vax / vms и в настоящее время является наиболее часто используемой версией Lisp для систем с разделением времени. Кроме того, он широко используется на 32-разрядных компьютерах и рабочих станциях, работающих под управлением операционной системы Unix. Благодаря своей хорошей переносимости Франс Лисп получил распространение в университетах и исследовательских институтах. Сопровождение системы также разошлось в различных исправлениях системных ошибок, реализациях наиболее эффективных алгоритмов, а также в расширениях языка [2].

Зеталисп также опирается на Маклисп. Он создан в 70-е годы в mit в рамках проекта Лисп-машины, финансированного оборонным агентством. С самого начала его целью было изготовление коммерческого продукта. В 1979 году в связи с проектом родились два предприятия, изготавливающие Лисп-машины: symbolic inc. и lisp machine inc. (lmi). После этого в 80-е годы работа по развитию Зета Лиспа продолжалась в них независимо друг от друга на коммерческой основе. Системы отличаются друг от друга, но в части Зета Лиспа машины почти совместимы [2; 7].

Зета Лисп содержит следующие развитые механизмы и черты:

− широкий выбор типов данных;

− возможность объектно-ориентированного программирования в системе flavor;

− современные управляющие структуры, динамические механизмы передачи управления, сопрограммы и процессы;

− гибкий механизм ключевых слов в лямбда-списке и многозначные функции;

− ввод и вывод, основывающиеся на потоках;

− пространства имен;

− уже готовые функции, в том числе для сортировки, работы с линейными управлениями и матричные вычисления.

Выбор используемых в среде Зеталиспа инструментов и языков программирования зависит от поставщика, предлагаемый набор средств расширяется. Среди других языков предлагаются Фортран, Паскаль, Ада и Пролог. Для этих языков в среде Зеталиспа существуют развитые программные окружения, а разработанные в них программы можно выполнять с программами на Лиспе. Готовые инструменты и прикладные разработки имеются для работы с ЭС, с естественным языком и речью, с реляционными базами данных, машинной графики и машинного проектирования [2; 3; 4].

Коммон Лисп. Этот диалект отличается широким представлением различных структур данных и включает около 800 встроенных функций. В этом диалекте обеспечиваются средства обработки основных классов числовой информации: целых, вещественных и комплексных. Символьные данные (литеры, литеральные атомы, строки) в Коммон Лиспе соответствуют основным возможностям других Лисп-систем. Дополнительно имеются средства обработки непечатных литер в символьных именах [7].

Одним из существенных преимуществ диалекта Коммон Лисп является наличие средств обработки массивов и структур, по своим возможностям не уступающих соответствующим средствам традиционных языков программирования (Фортран, Паскаль). Массивы в Коммон Лиспе могут иметь любое неотрицательное число измерений и индексируются последовательностью целых чисел. Тип компонентов массива может быть произвольным. Выделяется подкласс векторов − одномерных массивов, среди которых отдельно рассматриваются строки и битовые массивы.

Структуры Коммон Лиспа являются типом многокомпонентных записей, определяемых пользователем и имеющих именованные компоненты. Встроенное макроопределение defstruct используется для определения структур новых типов. Для создания данных нового типа в виде структуры предусмотрены средства автоматической генерации набора функций, обеспечивающих средства манипулирования объектами этого класса [2; 7; 12].

Удобным средством контроля доступа к различным переменным и функциям Лисп-программ в Коммон Лиспе являются пакеты. Пакет − множество имен объектов, определенных и доступных в нем. Внутри пакета имена объектов подразделяются на внутренние и внешние. Первые предназначены для использования только внутри данного пакета, а вторые − для обеспечения связи с

другими пакетами. Лисп-интерпретатор представляет широкий спектр средств манипулирования с пакетами. Как правило, Лисп-система имеет в своем составе четыре стандартных пакета: lisp (содержащий примитивы системы), user (умалчиваемый пакет прикладных программ и данных пользователя), keyword (содержащий ключевые слова всех встроенных функций и функций, определяемых пользователем), system (зарезервированный для системных целей) [4].

Переработке и расширению в Коммон Лиспе подверглись средства ввода-вывода и передачи информации. Для эффективной организации различных обменов с внешней средой введена концепция потоков, позволяющих осуществлять одно- и (или) двухстороннюю передачу информации. Для потоков предусмотрен набор базовых функций. В диалекте различают символьные и двоичные потоки. В первом варианте передача осуществляется по байтам, а во втором − целыми числами. Кроме стандартных потоков пользователь имеет возможность создавать и использовать собственные потоки [2; 4; 7].

В дополнение к указанным типам данных Коммон Лисп имеет ряд специфических классов объектов: хэш-таблицы, обеспечивающие эффективный способ доступа к данным по ключу; read-таблицы, обеспечивающие управление обработкой информации, поступающей из входного потока Лисп-системы, и некоторые другие. Такое множество имеющихся в диалекте различных типов данных, с одной стороны, развеивает существующее ошибочное представление о языке Лисп как о средстве обработки только символьной информации, а с другой − наличие мощных средств манипулирования типами данных существенно усложняет его [2; 7]. Этот диалект оставлен открытым: принципиальным является то, что осталась возможность в будущем, когда подойдет время и будет достигнуто согласие, добавить новые средства и методы. Эта идея соответствует духу Лиспа.

Коммон Лисп не является готовой программной системой в том же смысле, что и Интерлисп, поскольку вопросы среды в основном оставлены открытыми. В стандарте не определено, каким должен быть редактор или другие вспомогательные средства. Сказано лишь в самом общем виде, каким образом они вызываются. Для того чтобы обеспечить быстрое развитие, среда и инструментальные средства еще не затронуты стандартизацией, и поэтому есть возможность создавать различные среды для различных целей. Коммон Лисп не определяет также и интерфейс пользователя [2].

Коммон Лисп предназначен не только для работы со списками или для символьной обработки, он является универсальным языком программирования, включающим в себя неплохие средства для численных вычислений, управления данными и коммуникации. На Коммон Лиспе можно писать программы в традиционных операторном, процедурном, фразовом стилях, а также в специфических Лисп-стилях. В языке содержатся предпосылки для использования различных способов и стилей программирования, таких как операторное, функциональное, макропрограммирование, программирование, управляемое данными, продукционное программирование, средства, необходимые для логического и объектного программирования. Коммон Лисп содержит почти все, что могут дать другие известные языки программирования, и он предусматривает средства для расширения языка [2; 4; 7].

3.5 Языки программирования интеллектуальных решателей

Интеллектуальное решение [2; 7; 12]. Среди языков, с появлением которых возникли идеи о реализации интеллектуальных систем, необходимо выделить языки, ориентированные на программирование поисковых задач. Группа языков, которые можно назвать языками интеллектуальных решателей, в основном ориентирована на такую подобласть искусственного интеллекта, как решение проблем, которая характеризуется, с одной стороны, довольно простыми и хорошо формализованными моделями проблем, а с другой стороны, сложными методами поиска их решений. Поэтому основное внимание в этих языках было уделено внедрению мощных структур управления, а не способам представления знаний. Это такие языки, как Planer, Koniver, KuA-4, KuLisp [7; 10; 12].

Планер. Этот язык дал толчок мощному творчеству в области языков искусственного интеллекта. Он был разработан в Массачусетском технологическом институте в 1967–1971 годах. Сначала это было дополнение поверх Lisp, в таком виде язык реализован на Maclisp под названием Micro Planer. В дальнейшем Planer был значительно расширен и превратился в самостоятельный язык. В СССР он был реализован под названиями Планер-БЭСМ и Планер-Эльбрус. Этот язык привнес много новых идей в языки программирования: автоматический поиск с возвратами, поиск данных по выборке, вызов процедур по выборке, дедуктивный метод и т. д. [3; 12].

Planer содержит почти весь Lisp (с отдельными модификациями) и сохраняет его специфические особенности. Структура данных (выражения, атомы и списки), синтаксис программ и правила их вычисления в Planer аналогичны Lisp. Для обработки данных в Planer в основном используются те же инструменты, что и в Lisp: рекурсивные и блочные функции. Почти все функции Lisp, включая функцию eval, включены в планировщик. Аналогично определяются новые функции, так как в Lisp списки свойств могут быть связаны с атомами [10].

Но есть также различия между Lisp и Planer. В Lisp при обращении к переменной указывается только ее имя, например X, сам атом, как заданный, указывается как ‘x. Планировщик использует обратную нотацию: атомы обозначают сами себя, а при обращении к переменным перед их именем ставится префикс. В этом случае префикс указывает, как следует использовать переменную. Синтаксис обращения к функциям отличается от Lisp's, который записан в планировщике в виде списка не с круглыми, а с квадратными скобками.

Выполнение программы в режиме возврата удобно для ее разработчика, потому что язык берет на себя ответственность за запоминание форков и оставшихся в них альтернатив, за возврат к ним и восстановление предыдущего состояния программы − все это делается автоматически. Но такой автоматизм не всегда выгоден, так как он приводит к «слепому» поиску. При вызове теорем наиболее подходящая из них будет названа последней, хотя автор программы заранее знает о ее преимуществах. Принимая это во внимание, Planer обеспечивает управление режимом возврата.

Конивер. Язык Koniver был разработан в 1972 году и реализован как дополнение к языку Maclisp. Авторы языка Koniver раскритиковали некоторые идеи языка Planer. В основном это относилось к режиму автоматического возврата, который в целом приводит к неэффективным и неконтролируемым программам, особенно если он компилируется неквалифицированными пользователями. Авторы Koniver отказались от автоматического режима возвратов, полагая, что не нужно встраивать в язык какие-либо фиксированные управленческие дисциплины (кроме самых простых – циклов, рекурсий) и что автор программы должен сам организовывать нужные ему управленческие дисциплины, а для этого язык должен открывать его структуру управления пользователю и предоставить инструменты для работы с ним. Эта концепция была реализована в Coniver следующим образом [3, 12].

При вызове процедуры в памяти выделяется место, где хранится информация, необходимая для ее работы. Здесь, в частности, есть локальные переменные процедуры, указатели доступа (ссылка на процедуру, переменные которой доступны из этой) и return (ссылка на процедуру, которой должно быть возвращено управление). Обычно эта информация скрыта от пользователя, а на языке Coniver такая часть памяти (фрейм) открыта: пользователь может просматривать и изменять содержимое фрейма. В языке фреймы представляют особый тип данных, доступ к которым осуществляется с помощью указателей.

Недостатком языка является то, что хотя пользователь получает гибкие инструменты управления, в то же время на него ложится сложная и кропотливая работа, требующая высокой квалификации. Язык Coniver хорош не для реализации сложных систем, а как база, на основе которой квалифицированные программисты готовят необходимые механизмы управления для других пользователей.

Учитывая сложность реализации управленческих дисциплин, авторы языка были вынуждены включить в него ряд фиксированных механизмов управления, аналогов процедур-форков и теорем языка Planer. Но в отличие от Planer, где разрыв между выбором альтернативы в fork и ее анализом, а при необходимости и развитием отказа может быть сколь угодно большим, в языке Coniver этот разрыв сведен к минимуму. Делая это, разработчик избавляется от негативных последствий глобальных возвратов из-за сбоя, когда необходимо изменить предыдущую работу почти всей программы.

Языки представления знаний [12]. В рамках каждого базового языка искусственного интеллекта четко различаются его прямое использование, его расширение за счет наборов функций и создание автономного языка представления знаний (IAP) с последующей интерпретацией или компиляцией программ на созданном языке. Но в последнем случае базовый язык, как правило, становится инструментальным средством для реализации IAP [10; 12].

Независимо от реализации IAP должен соответствовать следующим требованиям:

− наличие простых и в то же время достаточно мощных средств представления сложных структурированных и взаимосвязанных объектов;

− возможность отображения описаний объектов на различных типах компьютерной памяти;

− наличие гибких инструментов управления выводом, которые учитывают необходимость структурирования правил решателя;

− «прозрачность» системных механизмов для программиста, т. е. их возможность быть определенными и переопределенными на уровне языка ввода;

− возможность эффективного внедрения как программного, так и аппаратного обеспечения;

Среди ЯПЗ первого поколения (70–80-е годы) лишь немногие сыграли значительную роль в программной поддержке систем представления знаний. Выделим KRL, FRL, KL-1. Характерными особенностями этих языков были: двухуровневое представление данных, представление законов предметной области и взаимосвязей между понятиями в виде прилагаемых процедур, семантический подход к сравнению выборок и поиску по образцу [3; 12; 13].

Язык KRL (Knowledge Relational Language), который был основан на следующих концепциях: организация знаний в виде специально выделенных единиц с прикрепленными описаниями и процедурами; наличие средств представления многомерных и/или неполных знаний об объектах; возможность описания объектов путем сравнения их с другими объектами, принимая во внимание уточняющие описания; наличие гибкого набора базовых инструментов для описания стратегий вывода решений и возможности их динамического изменения [12]. Однако KRL не получил широкого применения в интеллектуальных системах из-за своей несколько громоздкой природы и отсутствия собственных средств описания процессов. Lisp использовался в KRL.

Для всех ЯПЗ по сравнению с традиционными языками программирования характерна большая «активность» данных, что приводит к стиранию граней между декларативной и процедурной компонентами. В области разработки и реализации ЯПЗ можно выделить три круга проблем: определение входных языков ПЗ; выбор выходного языка соответствующего транслятора и собственно проблемы этапа трансляции. Входной язык ПЗ должен быть близок к языку предметной области по лексике, синтаксису, семантике. Выходной язык должен отвечать требованиям: иметь большой набор примитивов работы с образцами; обладать встроенными средствами поддержки рекурсии; иметь гибкие средства описания процессов управления. В рамках выходного языка необходимы средства отображения данных на основную и внешнюю память и удобные процедуры работы с этими данными. Желательно, чтобы в нем имелись средства определения новых типов данных. Выбор целевого языка ЯПЗ-транслятора всегда компромисс. Существуют сотни языков и систем представления знаний. Поэтому рассмотрим лишь некоторые особенности нескольких ЯПЗ [2; 7].

Язык RLL. Это фреймовый язык представления знаний (представитель популярного в 70-х годах подхода «фреймы до конца», является инструментальной средой для создания специализированных ЯПЗ. Он содержит два слоя: базисные примитивы и средства их комбинирования на более высоком уровне, чем Лисп. При этом технология конструирования специальных ЯПЗ в рамках

RLL-среды сводится к редактированию существующих заготовок и последующему конвертированию их в Лисп. Учитывая последовательную ориентацию RLL на концепцию фреймов, структуры (декларативные и процедурные) описываются здесь в виде фрейм-подобных RLL-элементов [10]. В RLL имеется и библиотека удачных управляющих структур, средства конструирования из них решателей, необходимых для конкретной ЭС. Одним из основных стандартных механизмов вывода решений в RLL является agenda (управляющий список с динамической коррекцией элементов) [10].

Язык ART. Это не только язык представления знаний, но и определенное программное окружение, включающее редакторы, отладчики, трансляторы и модули управления. Входной язык системы ART гибкий и обеспечивает использование фактов, схем, комбинаций этих понятий и правил. Декларативную компоненту этого ЯПЗ составляют факты и схемы. Факт включает три основных компонента: утверждение, значение истинности и точку зрения. С каждым утверждением может быть связано одно из трех значений истинности – true, false или unknown, а также определенные сферы его справедливости, которое и называется точкой зрения. Факты описываются экземплярами фреймов. Фреймы-прототипы в art представляются схемами, каждая из которых описывает объекты и/или классы объектов с фиксированными свойствами. Механизмы наследования свойств при этом поддерживаются самой системой [12].

ART погружен в Лисп-среду, так что синтаксически фреймовые и продукционные структуры выражаются как атомы, функции и списки языка Лисп. Такой подход в ART естественен, так как первоначально был реализован на Лисп-машинах. Средства описания фактов в языке ART почти полностью «отданы на откуп» Лиспу, что снижает концептуальную целостность языка, так как средства описания схем и правил здесь хотя и похожи на лисповские, но свои. В ART пользователю дается небольшой набор встроенных стратегий вывода решений и весьма ограниченный выбор из ART-действий, взаимодействующих с модулем вывода. Но в системе имеется возможность выхода в базовый язык Лисп, где программируются любые управляющие стратегии [12].

KIF (Knowledge Interchange Format)) компьютерно-ориентированный язык для обмена знаниями и взаимодействия между программами.

3.6 Описание знаний в Интернете

Язык XML. В результате продолжительных исследований по описанию данных и знаний в конце 90-х годов появился новый язык – XML (eXtensible Markup Language) или, более точно, стек спецификаций языков разметки различного назначения, которые базируются на общих синтаксических правилах. Возникновение XML было обусловлено, в частности, стремлением разработчиков унифицировать форму хранения документов для различных носителей информации. Не последнюю роль сыграла и необходимость развития и унификации формата хранения метаданных, преобразования документов в процессе их отображения и просмотра [5; 11; 13].

XML-документы имеют механизм для самоописания в виде Document Type Definition (DTD). Поскольку использование XML стало более разнообразным и интенсивным, ограничения DTD стали актуальными, особенно в области печатания данных, модульности и повторного использования. Поэтому консорциум W3C начал работу над XML в области создания нового поколения схем для XML. Сеть Интернет способствовала разработкам новых форм описания данных и знаний, среди которых можно выделить прежде всего язык XML со следующими преимуществами:

– текстовый формат представления данных воспринимается как компьютерами, так и людьми;

– структура и семантика XML-документа задается авторами, является гибкой и расширяемой;

– разделение содержания и представления, позволяющее повторно использовать данные при программировании, веб-дизайне и т. д.;

– передача и разбор XML-документов возможен стандартными транспортными протоколами и анализаторами соответственно.

Среди недостатков XML выделяют сложность синтаксиса и неоднозначность семантики, которая вызвана нестрогостью правил выбора атрибутов или элементов, используемых при составлении XML-документов. Но несмотря на это, появление и распространение XML приводит к существенным изменениям в концепциях представления данных и знаний, построения распределенных систем, методах программирования.

Таким образом, основным вкладом XML в развитие средств и методов интеграции данных и знаний является трансформация данных-параметров, зависящих от кода и протоколов транспортировки, в данные-документы, не зависящие от языков программирования, операционных систем, платформ. Обмен данными-документами (XML-сообщениями) лежит в основе нового поколения распределенных архитектур − веб-сервисов, компоненты которых взаимосвязаны стандартами Интернет, а не объектно-ориентированными технологиями, на которых базируются инфраструктуры CORBA, DCOM и др. Алгоритм интеграции приложений может содержаться не в самом интерфейсе взаимодействия компонентов, а в обработчике XML, который принимает сообщение и выполняет заложенные в нем инструкции. Рассмотрим Интернет-стандарты, составляющие основу архитектуры веб-сервисов и преимущества интеграции приложений на базе таких сервисов.

Логическая структура документа. Логическая структура документа определяет отношения объектов между собой. Существует по крайней мере две модели объектов документа: модель JavaScript и Document Object Model (DOM). Первая поддерживается (с некоторыми оговорками) практически всеми наиболее популярными браузерами. Вторая только претендует на роль стандарта и должна в будущем поддерживаться всеми браузерами [3; 13].

Отношения отдельных объектов между собой сводятся главным образом к отношениям типа «часть–целое», а структура документа представляет собой дере-

во. В роли остова выступает дерево блочных элементов разметки документа. Затем на этот остов накладываются строчные элементы и стили. Кроме того, у документа существует поддерево классов объектов документа, определяемое в DTD. Изменение свойств класса приводит к изменению свойств всех объектов данного класса.

Интерфейс прикладного программирования DOM. Объектные модели данных JavaScript, JScript и Java предшествовали появлению DOM-спецификации интерфейса прикладного программирования для HTML и XML. DOM наследует многие свойства этих моделей и творчески развивает их. DOM определяет логическую структуру документа, способы доступа к его элементам и манипулирования ими. При этом термин «документ» понимается широко – это единица информационного описания, которую можно закодировать на XML. Другими словами, XML-описание рассматривается в качестве документа, а DOM определяет способы манипулирования этим описанием [3; 13].

В рамках DOM программист может создавать документы, получать доступ к их составным частям (элементам), добавлять, удалять элементы и изменять их содержание. Главным в DOM был универсальный, независимый от языка программирования, интерфейс прикладного программирования документов Web. В спецификации DOM специально оговаривается вопрос о том, что применение DOM не подразумевает модель данных типа «дерево» или «лес деревьев». Авторы подчеркивают, что они оперируют структурной моделью данных (structured model), избегая формулировки «модель данных типа дерево или леса».

XML, для которого разработана DOM, опирается на представление данных в виде леса деревьев, но для манипулирования структурами такого типа не обязательно использовать именно эту модель данных. Можно обойтись списками или другими моделями, которые однозначно отображаются на деревья. Авторы специально подчеркивают, что если две разных реализации DOM создают один и тот же документ, то они создают одну и ту же модель данных с одинаковыми объектами и отношениями между ними. О таких реализациях DOM говорят, что они изоморфны. Таким образом, DOM определяет [5; 11]:

− интерфейсы и объекты для представления документа и манипулирования этим документом;

− семантику интерфейсов и объектов, включая их поведение и атрибуты;

− взаимоотношения и объединения интерфейсов и объектов.

Спецификация DOM состоит из двух частей: ядра DOM и применения DOM к HTML. Ядро обеспечивает функциональную полноту для обработки XML-документов, а также базис для DOM HTML. Ядро DOM позволяет определить документ как множество узлов, связанных между собой отношением «родитель–потомок», а ключевым объектом является Document. Отношение «родитель–потомок» определено в DOM следующим образом [3; 13]:

Document =: Element, ProcessingInstruction, Comment, DocumentType DocumentFragment =: Element, ProcessingInstruction, Comment,Text, CDATASection, EntityReference DocumentType =: нет потомков EntityReference =: Element, ProcessingInstruction, Comment, Text, CDATASection,EntityReference Element =: Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference Attr =: Text,

EntityReference ProcessingInstruction =: нет потомков Comment =: нет потомков Text =: нет потомков CDATASection =: нет потомков Entity := Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference

С другой стороны, в DOM определен объект Node (узел), у которого определен интерфейс NodeList, устанавливающий порядок узлов потомков объекта Node. Описание интерфейсов дано в обычном алфавитном порядке и разбито на описание ядра DOM и DOM HTML, а в ядре выделены фундаментальные интерфейсы. Следует отметить, что описание интерфейса – это фактически описание класса объектов. Только для методов класса отсутствуют реализации методов, которые перечислены в интерфейсах. В описании интерфейсов есть переменные, но в отличие от Java, где тоже имеется механизм интерфейсов, они могут определять как неизменяемые, так и изменяемые свойства объектов класса. Реализация интерфейсов в конкретных языках программирования − задача разработчиков этих языков в рамках описания соответствующих классов [13].

В DOM рассматриваются два уровня интерфейсов: базовый, оперирующий узлами, ориентирован на применение в рамках Java или других языков разработки приложений; прикладной уровень, который оперирует объектами типа Document или Element, а в рамках DOM HTML конкретными классами объектов HTML. Последний тип интерфейсов ориентирован на скриптовые языки и может применяться более широким кругом программистов [13].

Язык RDF. В усилиях направить дальнейшее воздействие сети на обществе, члены консорциума W3C пришло к тому, чтобы создать платформу для интернетного контекстного выбора (PICS), которая обеспечит пользователей способностью выбрать содержание информации на основании некоторых меток с поддержкой провайдеров и других потребителей. Критический компонент PICS − описание системы оценки, своего рода схема; каждая метка PICS указывает на описание в сети и на поля в метке.

PICS был разработан как первый шаг к обобщенным меткам, которые позволят любую составляющую сети сделать понятной по качеству ресурсов: терминов и условий для использования и так далее. Деятельность Metadata выполняет необходимую работу, чтобы закончить представление: структурированные метки, правила, интеграция с цифровыми подписями. Проект меток PICS был обобщен для информационной модели как направленные помеченные графы (DLGs). Модель известна как модель RDF, а преобразование в последовательную форму было определено в синтаксисе XML. Оценки системы PICS были включены как специальные случаи в проекте RDF-схем.

Начальное намерение состояло в том, что RDF будет просто встроен в XML с минимальным взаимодействием. Но затем проект RDF начал бы включать списки имен для соединения с XML именами элементов к адресам сети, что привело бы к давнишнему обсуждению проекта для XML (и SGML перед этим). Точно так же требования для типов данных RDF были разделены с многими другими XML основанными форматами.

Через какое-то время взаимодействия усложнялись. Модель объекта документа Document Object Model (DOM), которая была создана как усилие со-

гласовывать скрипты HTML для обслуживания в браузерах, была расширена включением XML и стала основополагающим прикладным программирующим интерфейсом (API) для многих платформ программного обеспечения сети и структурированных в репозиториях данных.

Программное обеспечение, основанное на этих платформах, рассматривает RDF не как XML-потоки, но как объекты проекта DOM. Появление компонента преобразования Extensible Style Language (XSL) в качестве полезного компонента позволило по-новому посмотреть на многих из синтаксических проблем проекта в RDF. Преимущество синтаксиса, которое позволяло легко управлять XSL, не было очевидным в ранней стадии проекта RDF. Более подробно язык RDF будет рассмотрен в главе 10.

Требования для общей модели данных сети. Семантическая сеть должна разрешить распределенным пользователям работать независимо, но чтобы взаимопонимание увеличивалось с добавлением новой информации без изменения существующей. Этот подход позволяет пользователям решать двусмысленности и разъяснять несуразности. Поэтому семантическая сеть должна быть основана на средстве, которое может расширяться. Это средство должно иметь возможность захватывать информацию, которая связывает независимые представления накладывающихся областей знаний.

3.7 Язык для моделирования процессов в управлении и технике

Постоянное усложнение производственно-технических и организационно-экономических систем – фирм, предприятий, производств и других субъектов производственно-хозяйственной деятельности и необходимость их анализа с целью совершенствования функционирования и повышения эффективности обусловливают необходимость применения специальных средств описания и анализа таких систем. Эта проблема приобретает особую актуальность в связи с появлением интегрированных компьютеризированных производств и автоматизированных предприятий.

В США это обстоятельство было осознано еще в конце 70-х годов, когда ВВС США предложили и реализовали Программу интегрированной компьютеризации производства ICAM (Integrated Computer Aided Manufacturing), направленную на увеличение эффективности промышленных предприятий посредством широкого внедрения компьютерных (информационных) технологий. Реализация этой программы потребовала создания адекватных методов анализа и проектирования производственных систем и способов обмена информацией между специалистами, занимающимися такими проблемами. Для удовлетворения этой потребности в рамках программы ICAM была разработана методология IDEF (ICAM Definition), позволяющая исследовать структуру, параметры и характеристики производственно-технических и организационно-экономических систем (в дальнейшем там, где это не вызывает недоразумений, – систем). Общая методология IDEF состоит из трех частных методологий моделирования, основанных на графическом представлении систем [4; 11]:

IDEF0 используется для создания *функциональной модели*, отображающей структуру и функции системы, а также потоки информации и материальных объектов, связывающие эти функции.

IDEF1 применяется для построения *информационной модели*, отображающей структуру и содержание информационных потоков, необходимых для поддержки функций системы.

IDEF2 позволяет построить *динамическую модель* меняющихся во времени поведения функций, информации и ресурсов системы.

Наибольшее распространение и применение имеют методологии IDEF0 и IDEF1 (IDEF1X), получившие в США статус федеральных стандартов. Методология IDEF0 основана на подходе, разработанном Дугласом Т. Россом в начале 70-х годов и получившем название SADT (Structured Analysis & Design Technique − метод структурного анализа и проектирования). Основу подхода и, как следствие, методологии IDEF0 составляет графический язык описания (моделирования) систем, обладающий следующими свойствами.

Графический язык − полное и выразительное средство, способное наглядно представлять широкий спектр деловых, производственных и других процессов и операций предприятия на любом уровне детализации. Он обеспечивает точное и лаконичное описание моделируемых объектов, удобство использования и интерпретации этого описания.

Язык облегчает взаимодействие и взаимопонимание системных аналитиков, разработчиков и персонала изучаемого объекта (фирмы, предприятия), т. е. служит средством «информационного общения» большого числа специалистов и рабочих групп, занятых в одном проекте, в процессе обсуждения, рецензирования, критики и утверждения результатов. Язык может генерироваться рядом инструментальных средств машинной графики; известны коммерческие программные продукты, поддерживающие разработку и анализ моделей − диаграмм IDEF0, например, продукт Design/IDEF 3.7 (и более поздние версии) фирмы Meta Software Corporation (США).

Перечисленные свойства языка предопределили выбор методологии IDEF0 в качестве базового средства анализа и синтеза производственно-технических и организационно-экономических систем, что нашло свое отражение в упомянутых федеральных стандартах США. В связи с расширяющимся применением информационных технологий и, в частности, CALS-технологий в народном хозяйстве Российской Федерации ниже приводятся основные сведения о методологии IDEF0 и графическом языке описания моделей, а также некоторые практические рекомендации по разработке таких моделей.

Концепция IDEF0. Методология IDEF0 основана на следующих концептуальных положениях. *Модель* – искусственный объект, представляющий собой отображение (образ) системы и ее компонентов. Согласно М *моделирует* А, *если* М *отвечает на вопросы относительно* А. Здесь М – модель, А – моделируемый объект (оригинал). Модель разрабатывают для понимания, анализа и принятия решений о реконструкции (реинжиниринге) или замене существующей, либо проектировании новой системы.

Система представляет собой совокупность взаимосвязанных и взаимодействующих частей, выполняющих некоторую полезную работу. Частями (элементами) системы могут быть любые комбинации разнообразных сущностей, включающие людей, информацию, программное обеспечение, оборудование, изделия, сырье или энергию (энергоносители). Модель описывает, что происходит в системе, как ею управляют, какие сущности она преобразует, какие средства использует для выполнения своих функций и что производит.

Блочное моделирование и его графическое представление. Основной концептуальный принцип методологии IDEF – представление любой изучаемой системы в виде набора взаимодействующих и взаимосвязанных блоков, отображающих процессы, операции, действия , происходящие в изучаемой системе. В IDEF0 все, что происходит в системе и ее элементах, принято называть *функциями*. Каждой функции ставится в соответствие *блок*. На *IDEF0-диаграмме,* основном документе при анализе и проектировании систем, блок представляет собой прямоугольник. Интерфейсы, посредством которых блок взаимодействует с другими блоками или с внешней по отношению к моделируемой системе средой, представляются *стрелками,* входящими в блок или выходящими из него. Входящие стрелки показывают, какие условия должны быть одновременно выполнены, чтобы функция, описываемая блоком, осуществилась.

Лаконичность и точность. Документация, описывающая систему, должна быть точной и лаконичной. Многословные характеристики, изложенные в форме традиционных текстов, неудовлетворительны. Графический язык позволяет лаконично, однозначно и точно показать все элементы (блоки) системы и все отношения и связи между ними, выявить ошибочные, лишние или дублирующие связи и т. д.

Передача информации. Средства IDEF0 облегчают передачу информации от одного участника разработки модели (отдельного разработчика или рабочей группы) к другому. К числу таких средств относятся:

− диаграммы, основанные на простой графике блоков и стрелок, легко читаемые и понимаемые;

− метки на естественном языке для описания блоков и стрелок;

− глоссарий и сопроводительный текст для уточнения смысла элементов диаграммы;

− последовательная декомпозиция диаграмм, строящаяся по иерархическому принципу, при котором на верхнем уровне отображаются основные функции, а затем происходит их детализация и уточнение;

− древовидные схемы иерархии диаграмм и блоков, обеспечивающие обозримость модели в целом и входящих в нее деталей.

Строгость и формализм. Разработка моделей IDEF0 требует соблюдения ряда строгих формальных правил, обеспечивающих преимущества методологии в отношении однозначности, точности и целостности сложных многоуровневых моделей. Эти правила описываются ниже. Здесь отмечается только основное из них: все стадии и этапы разработки и корректировки

модели должны строго, формально документироваться, с тем чтобы при ее эксплуатации не возникало вопросов, связанных с неполнотой или некорректностью документации.

Итеративное моделирование. Разработка модели в IDEF0 представляет собой пошаговую, итеративную процедуру. На каждом шаге итерации разработчик предлагает вариант модели, который подвергают обсуждению, рецензированию и редактированию, после чего цикл повторяется. Такая организация работы способствует оптимальному использованию знаний системного аналитика, владеющего методологией и техникой IDEF0, и знаний специалистов – экспертов в предметной области, к которой относится объект моделирования.

Отделение «организации» от «функций». При разработке моделей следует избегать изначальной «привязки» функций исследуемой системы к существующей организационной структуре моделируемого объекта (предприятия, фирмы). Это помогает избежать субъективной точки зрения, навязанной организацией и ее руководством. Организационная структура должна явиться результатом использования (применения) модели. Сравнение результата с существующей структурой позволяет, во-первых, оценить адекватность модели, а во-вторых – предложить решения, направленные на совершенствование этой структуры.

Процесс моделирования. Для эффективного моделирования и получения результатов в соответствии со сроками и сметами управление проектом должно представлять собой процесс, в ходе которого координируется работа авторов, экспертов и тех, кто принимает окончательную версию модели системы или ее части. Это должен быть процесс, в полной мере использующий возможности методологии, основанной на разделении функций участников проекта и итеративном характере рецензирования, в ходе которого проверяется корректность диаграмм и/или моделей, а также соответствие их поставленной цели и точке зрения.

IDEF0-модель есть результат скоординированной коллективной работы, при которой авторы создают первоначальные диаграммы, основанные на собранной информации об объекте моделирования, и передают их другим участникам проекта для рассмотрения и формулирования замечаний. Порядок требует, чтобы каждый эксперт, у которого есть замечания к диаграмме, сделал их письменно и передал автору диаграммы. Этот цикл продолжается до тех пор, пока диаграммы, а затем и вся модель не будут приняты.

Выводы

Сформулируем заключение, используя также источники [1−13].

1. Языки программирования разделяются на процедурные, функциональные логические. Большинство используемых языков программирования (Си, и др.) относятся к процедурным языкам, на которых описываются алгоритмы решения задач. Функциональные и логические языки относятся к области ИИ.

2. Функциональная программа состоит из совокупности определений функций, которые представляют собой вызовы других функций и предложений, управляющих последовательностью вызовов. Логическое программирование дает возможность программисту описывать ситуацию при помощи формул логики предикатов, а затем для выполнения выводов из этих формул применить автоматический решатель задач.

3. Сформированные принципы, положенные в основу языка Лисп: использование единого спискового представления для программ и данных; применение выражений для определения функций; скобочный синтаксис языка. После создания Лисп-систем: Маклисп, Интерлисп попытки создания языков ИИ, отличных от Лиспа, сходят на нет. Дальнейшее развитие языка идет по пути его стандартизации и в направлении создания новых языков для представления и манипулирования знаниями в Лисп-среде [7].

4. В начале 70-х годов появился язык, составивший конкуренцию Лиспу при реализации систем, ориентированных на знания, – Пролог, который поддерживает другую модель организации вычислений. Язык базируется на сопоставление образцов, древовидном представлении структур данных и автоматическом возврате при неудаче. Он приспособлен для решения задач, в которых фигурируют объекты и отношения между ними, применяется для представления знаний; в экспертных системах и других задачах ИИ.

5. Группа языков интеллектуальных решателей ориентирована на подобласть ИИ как решение проблем, для которой характерны достаточно простые и хорошо формализуемые модели задач и усложненные методы поиска их решения. Поэтому основное внимание в этих языках было уделено введению мощных структур управления, а не способам представления знаний (языки Плэнер, Конивер, КюА-4, КюЛисп).

6. Язык представления знаний должен отвечать требованиям: наличие простых и мощных средств представления сложноструктурированных и взаимосвязанных объектов; возможность отображения описаний объектов на разные виды памяти компьютера; наличие гибких средств управления выводом; «прозрачность» системных механизмов для программиста, возможность эффективной реализации, как программной, так и аппаратной. Среди первых ЯПЗ можно выделить krl, frl, kl-one [10].

7. Для описания знаний в Интернете применяются языки XML, XML-схемы, модель DOM, платформа PICS, язык RDF и RDF-схемы. Логическая структура документа определяет отношения объектов между собой. Существует две модели объектов документа: модель JavaScript и DOM.

8. DOM определяет логическую структуру документа, способы доступа к его элементам и манипулирования ими. При этом термин «документ» понимается широко – это единица информационного описания, которую можно закодировать на XML. XML-описание рассматривается в качестве документа, а DOM определяет способы манипулирования этим описанием. Ядро DOM позволяет определить документ как множество узлов, связанных между собой отношением «родитель–потомок», а ключевым объектом является Document.

9. Члены консорциума W3C создали платформу для интернетного контентного выбора – PICS, которая обеспечит пользователей способностью выбрать содержание информации на основании некоторых меток с поддержкой провайдеров и других потребителей. Критический компонент PICS − описание системы оценки, схема; каждая метка PICS указывает на описание в сети и на поля в метке. Проект меток PICS был обобщен для информационной модели, содержащей направленные помеченные графы (DLGs). На основе модели получен язык RDF, а преобразование в последовательную форму было определено в синтаксисе. XML. Оценки системы PICS были включены как специальные случаи в проекте RDF схем.

10. Для создания адекватных методов анализа и проектирования производственных систем и способов обмена информацией между специалистами в рамках программы ICAM была разработана методология IDEF, позволяющая исследовать структуру, параметры и характеристики производственно-технических и организационно-экономических систем. Общая методология IDEF состоит из трех частных методологий моделирования, основанных на графическом представлении систем: IDEF0 используется для создания функциональной модели, а также потоки информации и материальных объектов, связывающие эти функции; IDEF1 применяется для построения информационной модели, отображающей структуру и содержание информационных потоков; IDEF2 позволяет построить динамическую модель меняющихся во времени поведения функций, информации и ресурсов системы.

Вопросы для контроля

1. Поясните классификацию декларативных языков.
2. Охарактеризуйте язык Пролог.
3. Приведите пример программ на Прологе.
4. Охарактеризуйте язык ЛИСП.
5. Определите диалекты языка ЛИСП.
6. Перечислите языки интеллектуальных решателей.
7. Поясните языковые средства для представления знаний в Интернете.
8. Охарактеризуйте язык XML.
9. Охарактеризуйте язык IDEF.
10. Поясните общую методологию IDEF.

Литература, используемая в главе

1. Искусственный интеллект : справ. В 3 т. Т. 2. : Модели и методы / под ред. Д. А. Поспелова. – М. : Радио и связь, 1990. – 304 с. ; Т. 3. : Программные и аппаратные средства / под ред. Д. А. Поспелова. – М. : Радио и связь, 1990. – 363 с.

2. Метаданные [Электронный ресурс]. – Режим доступа: http://www. alfaconsultant. ru/ sklad/%&Ovr0/dpc31308.zip. – Дата доступа: 15.01.2023.

3. Вишняков В. А. Интеллектуальные системы в управлении. / В. А. Вишняков. – Минск : Изд-во МИУ, 2010. – 364 с.

4. Системы искусственного интеллекта. Практический курс. / В. А. Чулюков [и др.] ; под ред. И. Ф. Астаховой. – М., 2008. – 293 с.

5. Российский сервер «Информационные технологии в управлении» [Электронный ресурс]. – Режим доступа: http://www.cfin.ru/itm. – Дата доступа: 15.12.2021.

6. Адилов, Р. М. Системы искусственного интеллекта / Р. М. Адилов. – Пенза : ПГТУ. – 95 с.

7. ЛИСП [Электронный ресурс]. – Режим доступа: http://bestreferat.su/Informatika-programmirovanie. – Дата доступа: 15.12.2021.

8. Масленникова, О. Е. Основы искусственного интеллекта : учеб. пособие / О. Е. Масленникова, И. В. Попова. – Магнитогорск : МаГУ, 2008. – 282 с.

9. Костров, Б. В. Искусственный интеллект и роботехника : учеб. пособие / Б. В. Костров, В. Н. Ручкин, В. А. Фулин. – М. : Диалог-МИФИ, 2008. – 224 с.

10. Башлыков, А. А. Решатели интеллектуальных задач для человеко-машинных систем поддержки принятия решений / А. А. Башлыков // Автоматизация, телемеханизация и связь в нефтяной промышленности. – 2013. – № 10. – С. 6–20.

11. Корнеев, И. К. Информационные технологии в работе с документами : учебник / И. К. Корнеев. – М. : Проспект, 2015. – 297 с.

12. Решатели интеллектуальных задач для человеко-машинных систем [Электронный ресурс]. – Режим доступа: http://elibrary.ru/item.asp?id=20368896. – Дата доступа: 15.12.2021.

13. Метаданные [Электронный ресурс]. – Режим доступа: http://www.citforum.ru/internet/search/tips.shtml. – Дата доступа: 15.12.2022.

4 ОСНОВЫ ЭКСПЕРТНЫХ СИСТЕМ

4.1 Экспертные системы − понятия и определения

Особенности ЭС. Основными отличиями ЭС от других программных средств являются использование не только данных, но и знаний, а также специальный механизм вывода решений и новых знаний на основе существующих и объяснение их получений. В ЭС формируется алгоритм обработки знаний и отсутствует алгоритм решения задачи [4; 6; 7].

Алгоритм обработки знаний заранее неизвестен и строится в процессе решения задачи на основе эвристических правил и механизма управления выводом. Решение проблемы в ЭС сопровождается удобными для пользователя пояснениями. Проблемы представляются системе в виде набора фактов, описывающих определенную ситуацию, и система пытается сделать вывод из этих фактов с помощью базы знаний.

ЭС функционирует в таком циклическом режиме: ввод (запрос) данных или результатов анализа задачи, интерпретация результатов, усвоение новой информации, выдвижение временных гипотез с использованием правил, а затем выбор следующей группы знаний или результатов анализа.

Виды знаний. В любой момент времени в экспертной системе существует три разновидности знаний: структурированные знания − статические о предметной области (справочники, таблицы); структурированные динамические знания о предметной области, которые обновляются по мере появления новой информации; рабочие знания, используемые для решения текущей задачи или консультации. Все три вида знания хранятся в базе знаний (БЗ). Чтобы построить эту базу, требуется провести опрос специалистов, которые являются экспертами в определенной предметной области, а затем систематизировать их с помощью инженера-программиста, описать с помощью ЯПЗ (см. главу 3) и снабдить эти знания метками, чтобы их можно было извлечь из базы знаний. На рисунке 4.1 показаны основные компоненты ЭС [5; 6; 7].

Пользователь является начинающим специалистом в предметной области, для которой предназначена ЭС. Его квалификация недостаточно высока, и поэтому он нуждается в помощи и поддержке со стороны экспертной системы. Инженер по знаниям − это программист по искусственному интеллекту, который выступает посредником между экспертом и БЗ ЭС.

Пользовательский интерфейс представляет собой комплекс программ, реализующих диалог пользователя с экспертной системой на этапе как ввода информации, так и получения результатов [3; 4]. База знаний − это ядро ЭС, совокупность знаний предметной области, записанных на машинном носителе в форме, понятной эксперту и пользователю.

Решатель − это программная часть ЭС, которая имитирует ход рассуждений эксперта на основе знаний, доступных из БЗ. Подсистема объяснения − это программная часть ЭС, которая позволяет пользователю получать ответы на вопросы: «Как была получена та или иная информация?» или «Почему экс-

пертная система приняла такое решение?». Ответ на вопрос «Как?» − это объяснение всего процесса получения решения с указанием использованных элементов базы знаний [4].

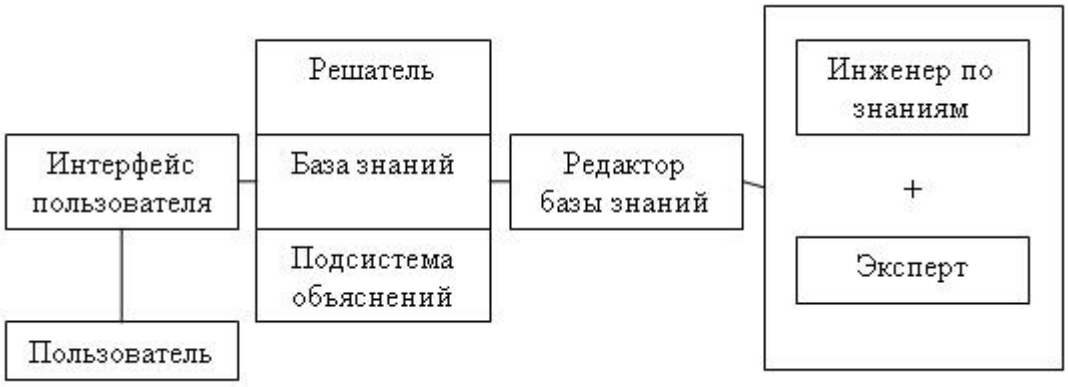

Рисунок 4.1 − Обобщенная структура экспертной системы

Интеллектуальный редактор базы знаний − это программная часть ЭС, которая предоставляет инженеру по знаниям возможность создавать базу знаний в интерактивном режиме. Инструментальные средства построения экспертных систем.

Языки, которые используются для построения ЭС, традиционно Лисп, Пролог. «Оболочки» − это «пустые» версии существующих ЭС-систем (без наполненной БЗ), то есть готовые экспертные системы без сформированной базы знаний [8]. Они, как правило, не требуют работы программистов для создания готовой экспертной системы. Для заполнения базы знаний требуются только специалисты в соответствующей предметной области.

4.2 Отличительные особенности ЭС, области применения

Рассмотрим особенности экспертных систем [6; 7].

1. Обследование может быть проведено только в одной конкретной предметной области. Программа, предназначенная для определения конфигурации инфокоммуникационных систем, не может ставить медицинские диагнозы.

2. База знаний и механизм вывода − это разные компоненты ЭС. Действительно, часто можно комбинировать механизм вывода с другими базами знаний для создания новых предметных ЭС.

3. Наиболее эффективным подходом при применении ЭС является решение задач дедуктивным методом. Например, правила или эвристика выражаются в виде пар посылок и выводов продукционной модели «Если А− То В».

4. ЭС может объяснить ход решения проблемы на естественном языке для пользователя. Обычно мы сомневаемся принимать ответ эксперта, если на вопрос «Почему?» не можем получить внятный ответ. Аналогичным образом мы должны иметь возможность спросить экспертную систему, основанную на знаниях, как был получен тот или иной результат.

5. Выходные результаты являются качественными, а не числовыми.

6. Экспертные системы, основанные на знаниях, построены по модульному принципу, что позволяет им со временем пополнять свои базы знаний и модифицировать механизм вывода.

Экспертные системы, которые могут только повторять логический вывод одного эксперта, являются ЭС первого поколения. Но для специалиста, решающего интеллектуально сложную задачу, недостаточно иметь систему, которая только имитирует человеческую деятельность. ЭС, принадлежащие ко второму поколению, называются партнерскими. Их отличительными чертами являются способность учиться и развиваться, то есть эволюционировать.

В ЭС первого поколения знания представлены таким способом: это только знания эксперта при формировании ЭС, опыт накопления знаний не предусмотрен; методы представления знаний позволили описать только статические предметные области; модели представления знаний ориентированы на простые области [5−7].

Особенности представления знаний в ЭС второго поколения следующие: в них используются не первичные знания, а более глубокие (полученные в процессе решения), есть возможность пополнять предметную область; система может решать задачи с динамической БЗ предметной области.

Области применения ЭС [5−7]. Области применения систем, использующих знания, можно сформировать в несколько предметных классов: медицинская диагностика, контроль и управление в технике, диагностика неисправностей в механических и электрических устройствах, обучение и т. д.

Медицинская диагностика. Диагностические системы используются для установления связи между заболеваниями человека и их возможными причинами. Наиболее ранней такой диагностической системой является MYCIN, которая предназначена для диагностики и мониторинга состояния пациента при менингите и бактериальных инфекциях. Ее первая версия была разработана в Стэнфордском университете в середине 70-х годов. Эта система ставит диагноз на уровне опытного врача-специалиста.

Прогнозирование. Прогностические системы предсказывают возможные исходы или события на основе данных о текущем состоянии объекта. Программная система «Завоевание Уолл-стрит» может анализировать рыночную ситуацию и разрабатывать план капиталовложений на будущее, используя статистические методы алгоритмов. Системы прогнозирования могут предсказывать погоду, урожайность и пассажиропоток.

Планирование. Эти ЭС планирования предназначены для достижения конкретных целей при решении задач с большим количеством изменяемых параметров. Дамасская фирма Informat предоставляет клиентам несколько рабочих станций, установленных в вестибюле ее офиса, где проводятся бесплатные 15-минутные консультации, чтобы помочь клиентам выбрать конфигурацию ноутбука, который наилучшим образом соответствует их запросам, финансам.

Фирма Boeing использует другую ЭС для проектирования космических станций, а также для выявления причин отказов авиационных двигателей и ре-

монта вертолетов. Экспертная система XCON (фирма DEC) используется для определения или изменения конфигурации компьютерных систем типа VAX и в соответствии с требованиями покупателя [8].

Толкование. Интерпретирующие системы обладают способностью делать выводы на основе результатов наблюдения. Система PROSPECTOR, объединившая знания девяти экспертов, позволила обнаружить залежи руды стоимостью в десятки миллионов долларов, и ни один из девяти экспертов не предполагал о наличии этих подобных ископаемых. Другая система интерпретации − HASP/SIAP определяет местоположение и типы судов в Тихом океане по данным акустических систем слежения за движущимися объектами.

Контроль и управление. ЭС этого класса могут принимать решения, анализируя данные, поступающие из многих источников. Эти системы уже работают на тепловых и атомных электростанциях, управляют воздушным движением в аэропорту. Они также могут быть полезны при регулировании финансовой деятельности предприятия и помогать в разработке решений в критических ситуациях.

Диагностика неисправностей в механических и электрических устройствах. В этой области системы незаменимы, как для ремонта механических и электрических машин (турбин, вагонов, тепловозов и т. д.), так и для устранения неполадок и ошибок в компьютерном оборудовании и программном обеспечении сложных систем инфокоммуникаций.

Обучение. Система получает информацию об активности определенного объекта (например, студента) и анализирует его активность, ответы на вопросы и т д. База знаний изменяется в соответствии с поведением объекта. Примером такого обучения является компьютерная игра, сложность которой возрастает по мере повышения уровня мастерства играющего.

4.3 Использования ЭС, ограничения, преимущества

При применении ЭС нужно руководствоваться несколькими положениями [4; 5; 6; 8].

1. Данные и знания правдивы и не меняются со временем.

2. Пространство возможных решений относительно невелико.

3. В процессе решения задачи используются формальные рассуждения.

4. Должен быть хотя бы один эксперт, который при создании ЭС формулировал введенные знания этой предметной области и мог объяснить методы применения этих знаний для решения отдельных задач.

В таблице 4.1 приведены сравнительные свойства прикладных задач, по особенностях которых можно судить о целесообразности использования ЭС для их решения. Исходя из специфики задач ЭС не рекомендуется применять для решения следующих видов:

− математических, решаемых обычным путем формальных преобразований и процедурного анализа;

− вычислительных, поскольку они решаются численными методами;

− задач, для которых невозможно построить базу знаний.

Таблица 4.1 − Критерий применимости ЭС

Применимы	Неприменимы
Не могут быть построены строгие алгоритмы или процедуры, но существуют эвристические методы решения	Имеются эффективные алгоритмические методы
Есть эксперты, которые способны решить задачу	Отсутствуют эксперты или их квалификация не удовлетворяет
Задачи относятся к областям, рассмотренным в п. 4.2	Задачи носят вычислительный характер
Доступные данные неточны или недостоверны	Известны точные факты и строгие процедуры
Задачи решаются методом формальной логики	Задачи решаются известными процедурными методами
Знания статичны (неизменны со временем)	Знания динамичны (меняются со временем)

Ограничения в применении ЭС. Даже лучшие из существующих ЭС имеют ограничения по сравнению с человеком-экспертом [6; 7].

1. Отдельные ЭС не вполне пригодны для применения конечным пользователем. Другие ЭС оказываются доступными только тем экспертам, которые создавали их базы знаний.

2. Вопросно-ответный режим, используемый в системах, замедляет получение решений, а иногда не подходит, поскольку решение требуется в режиме реального времени.

3. Возможности системы не всегда возрастают после сеанса экспертизы.

4. Все еще остается проблемой приведение знаний, полученных от эксперта, к виду, обеспечивающему их лучшую компьютерную реализацию.

5. ЭС не обладают здравым смыслом, в типовых ситуациях путаются.

6. ЭС не применимы в очень больших предметных областях. Их использование ограничивается предметными областями, в которых эксперт может принять решение за время от нескольких минут до нескольких часов.

7. В тех областях, где отсутствуют эксперты (например, в астрологии), применение ЭС оказывается невозможным.

8. Динамические области, теннис, езда на велосипеде не могут являться предметной областью для ЭС, однако такие системы можно использовать при формировании составов спортивных игровых команд.

9. Человек-эксперт при решении задач обращается к своей интуиции или здравому смыслу, если отсутствуют формальные методы решения или аналоги таких задач.

Преимущества ЭС перед человеком-экспертом. Системы, основанные на знаниях, имеют определенные преимущества перед человеком-экспертом [6].

1. У них нет предубеждений.

2. Они не делают поспешных выводов.

3. Эти системы работают систематизированно, рассматривая все детали, выбирая лучшую альтернативу из всех возможных.

4. База знаний может быть очень большой. Будучи введены в машину один раз, знания сохраняются навсегда. Человек же имеет ограниченную базу знаний, и если данные долгое время не используются, то они забываются и теряются.

5. Системы, основанные на знаниях, устойчивы к неполным знаниям или «помехам». Эксперт пользуется побочными знаниями и поддается влиянию внешних факторов, которые не связаны с решаемой задачей. ЭС, не обремененные знаниями из других областей, менее подвержены «шумам».

6. В целом ЭС не заменяют специалиста, а являются дополнительным инструментом в его профессиональной деятельности.

4.4 Методология и этапы разработки ЭС

Пользователи ЭС. Рассмотрим построение ЕС, структура которого была представлена на рисунке 4.1 [5; 6; 7]. Она имеют два отдельных входа, соответствующих разным целям взаимодействия пользователя с ЕС: начинающий пользователь, которому нужна консультация ЕС в процессе диалога с ней, во время которого она решает какую-то экспертную задачу. Существуют две формы диалога с ЕС − использование словаря-меню (в котором система предлагает выбор профессиональной лексики экспертов на каждом этапе диалога) и взаимодействие, основанное на нескольких действиях); экспертная группа работает с помощью специализированного диалогового компонента подсистемы получения знаний ЭС, который позволяет частично автоматизировать этот процесс [6; 7].

Подсистема интеллектуального ввода знаний предназначена для добавления новых фактов, правил в БЗ и изменения существующих. Ее задача − привести правило к форме, позволяющей подсистеме вывода применять его во время обработки. В этой подсистеме существуют также средства для проверки введенных или измененных правил на соответствие с существующими правилам на предмет противоречия.

База знаний. Наиболее распространенный способ представления знаний – в форме конкретных фактов и правил, с помощью которых новые факты могут быть выведены из имеющихся фактов. Факты могут быть представлены, например, в виде троек: (ЗНАЧЕНИЕ АТРИБУТ ОБЪЕКТ). Этот факт означает, что указанный объект имеет указанный атрибут (свойства) с указанным значением. Например, запись (ТЕМПЕРАТУРА ПАЦИЕНТА Иванова равна 37,9) представляет факт «Температура пациента Иванова равна 37,9». В более простых случаях факт выражается каким-либо простым утверждением, которое может быть истинным или ложным, например: «Надежность падает». Правила в БЗ имеют вид продукций:

ЕСЛИ А, ТО Б, где А − условие; Б − действие.

Действие Б выполняется, если А имеет значение истины. Действие Б и условие А представляет собой утверждение, которое может быть выведено системой (то есть оно становится ей известным), если условие правила А верно. Приведем простой пример правила из инфокоммуникаций:

ЕСЛИ сеть не работает, ТО необходимо найти неисправность (правило 1).

Пример этого правила с более сложным условием:

ЕСЛИ сеть не работает и клиенты жалуются, то необходимо найти неисправность (правило 2).

Если факты «Сеть не работает «и клиенты жалуются» уже находятся в рабочем наборе, то после применения вышеуказанного правила в него также включается новый факт «найти неисправность».

Если система не может вывести какой-либо факт, истинность или ложность которого необходимо установить, то система запрашивает пользователя об этом. Например:

ПРАВДА ЛИ, ЧТО сеть не работает ?

При получении положительного ответа от пользователя факт «Сеть не работает» включается в рабочий набор.

Подсистема вывода. Она выполняет две функции: во-первых, просматривает существующие факты из рабочего набора и правила из базы знаний и добавляет новые факты в рабочий набор и, во-вторых, определяет порядок просмотра и применения правил [6; 7].

Целью ЭС является вывод некоторого факта, который называется целевым утверждением, или опровержение этого факта. Целевое утверждение может быть либо заранее «встроено» в базу знаний системы, либо извлечено системой из диалога с пользователем. Цикл заканчивается, когда целевое утверждение выводится или опровергается.

Логический вывод может происходить несколькими способами, из которых более распространенными являются прямой и обратный порядок вывода. Прямой порядок вывода – от фактов, которые находятся в рабочем наборе, к заключению. Если такой (вывод) может быть найден, то он заносится в рабочую часть БЗ. Чтобы проиллюстрировать это, давайте добавим еще одно правило к нашему примеру БЗ:

ЕСЛИ вам нужно найти неисправность, то вам необходимо устранить неисправность (правило 3).

Давайте также предположим, что факты «Сеть не работает», «клиенты жалуются» уже находятся в рабочей области БЗ, и цель системы – ответить на вопрос пользователя:

«Необходимо устранить неисправности?»

При прямом выводе система будет работать следующим образом:

Шаг 1. Правило 1 рассмотрено. Его условие истинно, поскольку оба элемента продукции присутствуют в рабочем наборе, то применяем правило 2; добавляем факт «Необходимо найти неисправность» к рабочему набору.

Шаг 2. Рассматривается правило 3. Его условие истинно, потому что факт из условия этого правила присутствует в рабочей области БЗ. Применяем правило 3; получаем факт «Необходимо устранить неисправности» в рабочий набор. Целевое утверждение является производным.

Обратный порядок вывода в ЭС [6]: правила просматриваются до тех пор, пока факты, подтверждающие один из них, не будут найдены в рабочей памяти

или получены от пользователя. В системах с обратным выводом сначала выдвигается какая-то гипотеза, а затем механизм вывода в процессе работы возвращается назад, переходя от гипотезы к фактам, и пытается найти среди них те, которые подтверждают ее. Если она оказалась правильной, то выбирается следующая гипотеза, детализирующая первую, которая является подцелью по отношению к ней. Далее обнаруживаются факты, подтверждающие истинность подчиненной гипотезы. Обратный поиск используется в тех случаях, когда цели известны и их относительно немного. В некоторых системах вывод основан на комбинации обратного и ограниченного прямого преобразования. Этот комбинированный метод называется циклическим.

4.5 Компоненты вывода

Механизм вывода включает в себя два компонента − один реализует вывод, другой управляет этим процессом вывода. Компонент вывода выполняет первую задачу, рассматривая существующие правила и факты из рабочего набора и добавляя к нему новые факты при срабатывании очередного правила. Управляющий компонент определяет порядок выбора и применения правил [5; 6; 7].

Составляющая вывода, его действия основаны на применении правила дедукции, называемого modus ponens, суть которого заключается в следующем: пусть будет известно, что утверждение А истинно и существует правило вида «Если А, то Б», тогда утверждение Б также верно. Правила срабатывают, когда есть факты, удовлетворяющие истинности их левой части (А): если посылка верна, то и вывод Б должен быть верным.

Хотя в принципе кажется, что такой вывод легко может быть реализован на компьютере, тем не менее на практике человеческий мозг все же оказывается более эффективным при решении задач. Рассмотрим, например, простое предложение: Маша искала ключ. Здесь для слова «ключ» приемлемы, как минимум, несколько значений: «источник воды» и «ключ от дома, дачи, машины». В следующих двух предложениях одно и то же слово имеет совершенно разные значения: «Они заблудились в чаще». «Им нужно чаще ходить в театр».

Становится еще труднее понять факты, если они являются компонентами продуктов, которые используют правило modus ponens для вывода. Вот пример:

ЕСЛИ светлую машину можно заметить ночью, и машина Ивана светлая, то машину Ивана можно заметить ночью.

Этот вывод можно сделать даже школьнику, но некоторым ЭС это оказывается затруднительно.

Выходной компонент должен обладать способностью функционировать при разных условиях. Механизм вывода должен быть способен продолжать рассуждения и в заключение находить решение даже при неполной информации. Это решение может быть неточным, вероятностным, но ЭС не должна останавливаться из-за отсутствия какой-либо части входной информации.

Компонент управления. Этот компонент определяет порядок применения правил, а также вычисляет, есть ли еще факты, которые могут быть использованы, если работа ЭС продолжится. Компонент управления выполняет четыре функции [6]:

1. Сравнение – левая часть очередного правила сравнивается с введенными фактами, далее выбор правила.

2. Выбор − если в конкретной ситуации можно применить сразу несколько правил, то выбирается одно из них, подходящее по введенным фактам (далее запуск).

3. Запуск − если левая часть правила совпала с какими-либо фактами из рабочего набора во время сравнения, то правило запускается (далее действие).

4. Действие − рабочий набор фактов может быть изменен путем добавления к нему заключения (факта из правой части) сработавшего правила.

Интерпретатор правил работает с повторением. В каждом цикле он пересматривает все правила, чтобы выявить среди них те из них, в которых совпадают левые части с известными уже фактами из рабочей области БЗ. Интерпретатор также определяет порядок, в котором применяются правила. После того как правило выбрано, его заключение (полученный факт) вводится в рабочий набор, а затем цикл повторяется с самого начала. Только одно правило может работать в одном цикле. Рабочий цикл схематично можно представить следующим образом [3; 6].

1. Информация (известные факты) из рабочее области БЗ последовательно сравнивается с допущенными правилами для определения успешного сравнения.

2. Набор выбранных правил составляет набор конфликтов.

3. Для разрешения конфликта у интерпретатора есть критерий, по которому он выбирает единственное правило (полное совпадение левой части), после чего оно запускается.

4. Это приводит к включению новых фактов в рабочей области БЗ или в изменении критерия выбора конфликтующих правил.

5. Если заключение включает в себя название действия, то последнее выполняется (например, подается звуковой сигнал, начинает запускаться программа и т. д).

Новые данные, введенные в экспертную систему, обрабатываются правилом, которое, в свою очередь, может изменить критерий выбора правила. Если экспертная система, предназначенная для игры в шахматы, разыгрывает партию для двух игроков, то она может принять решение придерживаться атакующей стратегии посредством хода, т. е. один из партнеров будет атаковать. Если оппонент играет по этой системе, то в какой-то момент ЭС может переключиться на использование оборонительной стратегии (скорее, временно), а затем снова вернуться к наступательной тактике. Изменение критерия основано на выводах, полученных после анализа позиции на доске, которая представлена в рабочем наборе системы, а также правил игры в виде статических и динамических знаний (эвристики).

На самом деле у ЭС нет процедур, которые могли бы построить весь путь решения проблемы в пространстве состояний сразу. Более того, часто даже невозможно определить, есть ли вообще какое-либо решение проблемы. Тем не менее поиск решения осуществляется, поскольку движение в пространстве состояний контролируется заложенными процедурами. Их называют демонами, потому что они пассивны во время работы системы и активируются только тогда, когда их просят о помощи, т. е. они действительно ведут себя как хорошие помощники.

Демоны получили свое название от «демона Максвелла» − действующего лица одного из мысленных экспериментов, предложенных его автором для критики законов термодинамики [3; 6]. Если использовать научную терминологию, то такие процедуры контроля называются недетерминированными. Это означает, что траектория поиска решения в пространстве состояний полностью определяется данными.

При разработке управляющей составляющей механизма (подсистемы) необходимо решить вопрос о том, какой критерий следует использовать для выбора правила, которое будет применяться в конкретном цикле. Уже на ранней стадии разработки ЕС необходимо знать, что будет вводить конечный пользователь. Это необходимо для того, чтобы убедиться, будет ли ЭС достаточно практичной и сможет ли она адаптироваться к среде, в которой она будет функционировать. Участие пользователей может выражаться в следующем [6, 7]:

− выполнение неких задачи, пользователь, столкнувшись с конкретными проблемами, может объяснить возникновение проблем и предложить использовать возможные процедуры для них;

− коммуникация, пользовательский интерфейс должен соответствовать техническому словарному запасу пользователя и уровню профессиональной подготовки;

− установление ЭС с ним связей, познакомить пользователя с причинами и следствиями, которые вызывают то или иное действие в процессе функционирования ЭС, это имеет значение при определении взаимосвязей фактов в базе знаний;

− обратная связь, отличительной особенностью простого в использовании момента работы ЭС является ее способность объяснять конечному пользователю последовательность своих шагов при решении задачи.

Диалог с ЭС, объяснение. Поскольку системы, основанные на знаниях, реализованы на компьютерных технологиях, входная информация воспринимается в форме, принятой компьютером, т. е. в битах и байтах. Однако для того, чтобы неподготовленный пользователь мог взаимодействовать, ЭС должна включать средства коммуникации на языке, близком к естественному.

Некое большинство ЭС, основанных на знаниях, имеют довольно простой интерфейс на естественном языке – допустимые сообщения пользовательского ввода ограничены набором терминов, содержащихся в БЗ. Используя пример простых ЭС и БЗ, диалог пользователя с системой можно представить:

ЭС: Вы хотите знать, необходимо ли принять меры в сети?
Пользователь: Да.
ЭС: Правда ли, что сеть не работает?
Пользователь: Да.
ЭС: Правда ли, что клиенты жалуются?
Пользователь: Да.
ЭС: Необходимо принять меры в сети.

Как видно из этого примера, во время консультации инициатива диалога принадлежит системе, а сама консультация с ЭС выглядит так же, как консультация с экспертом-человеком: задается ряд вопросов и на основе их анализа выдается конечное заключение. Однако, в отличие от беседы с экспертом, диалог с ЭС имеет свои специфические особенности: большинство пользователей склонны доверять «мнению» ЭС меньше, чем мнению «живого» эксперта.

Чтобы проверить «разумность и компетентность» ЕС, пользователь может обратиться к его подсистеме объяснений. Для того чтобы понять, как это работает, нам нужно рассмотреть вопрос о том, в какой форме хранить информацию о процессе наших рассуждений. В ЕС принято представлять процесс логического вывода в виде диаграммы, которая называется деревом вывода. В примере дерево вывода будет выглядеть следующим образом [6; 7]:

Для того чтобы понять, как она работает, нам необходимо рассмотреть вопрос о том, в какой форме ЭС хранит информацию о процессе своих рассуждений. В ЭС принято представлять процесс логического вывода в виде схемы, которая называется *деревом вывода*. В примере дерево вывода будет иметь вид [7]:

Сеть не работает И клиенты жалуются ТО нужно искать неисправность
Правило 1.

Если нужно искать неисправность То Необходимо устранить неисправности
Правило 2.

4.6 Стратегии управления выводом

Разработка стратегии. Важным вопросом, возникающим при проектировании управляющей компоненты вывода ЭС, основанных на БЗ, является выбор метода поиска решения, т. е. стратегии вывода. От выбранного метода поиска будет зависеть порядок и скорость применения и срабатывания правил. Процедура выбора сводится к определению направления поиска и способа его осуществления. Для управления выводом в ЭС может применяться поиск в глубину, в ширину, полный перебор или выборочный по дереву решений. Процедуры, реализующие поиск, обычно включены как программы в подсистему вывода, поэтому в большинстве ЭС инженеры знаний не имеют к ним доступа и, следовательно, не могут в них ничего изменять по своему желанию. При разработке стратегии управления выводом необходимо ответить на два вопроса [4; 5; 6; 7; 9]:

1. Какую позицию в пространстве состояний принять в качестве исходной? Дело в том, что еще до начала поиска решения ЭС должна каким-то образом выбрать исходную позицию поиска в прямом или обратном направлении.

2. Как повысить эффективность поиска решения? Чтобы добиться повышения эффективности поиска решения, необходимо найти эвристики разрешения конфликтов, связанных с существованием нескольких возможных путей для продолжения поиска в пространстве состояний, поскольку требуется отбросить те из них, которые заведомо не ведут или замедляют процесс поиска нужного решению.

Повышение эффективности поиска. В БЗ экспертной системы может быть сотни правил, нужно найти какую-либо стратегию управления выводом, позволяющую минимизировать время поиска решения, и тем самым повысить эффективность вывода. К числу таких стратегий относятся поиск в глубину, поиск в ширину, разбиение на подзадачи и альфа-бета алгоритм [6].

Сопоставление методов поиска в глубину и ширину. Суть поиска в глубину дерева состоит в том, что при выборе очередной подцели в пространстве состояний (дерево решений) предпочтение отдается той ветви дерева, которая соответствует более детальному уровню описания задачи. Пространство состояний − это граф (дерево), вершины которого соответствуют ситуациям, а решение задачи сводится к поиску пути в этом графе.

При поиске в ширину ЭС проанализирует все признаки, находящиеся на одном уровне дерева, и лишь затем перейдет к исследованию следующего уровня. Ученые в узкой области науки оценивают поиск в глубину, поскольку он позволяет собрать воедино все признаки, связанные с выдвинутой гипотезой. Ученые в нескольких областях науки отдают предпочтение поиску в ширину, т. к. в этом случае они не ограничиваются заранее очерченным кругом признаков, т. е. можно найти принципиально новое решение.

Особенности пространства поиска определяют целесообразность применения той или иной стратегии: например, ЭС для игры в шахматы реализуются на основе поиска в ширину, поскольку при использовании поиска в глубину число анализируемых ходов может быть и очень большим [6].

Альфа-бета алгоритм. Задача для этого алгоритма заключается в уменьшении области решений (отсечению ветвей дерева) путем удаления в нем вариантов, заведомо тупиковых для поиска успешного решения. Поэтому просматриваются только те вершины, в которые можно попасть в результате следующего шага, после чего неперспективные узлы и ветви за ними исключаются из просмотра. Например, если цвет предмета, который мы ищем, не синий, то его бессмысленно искать среди синих и голубых предметов. Альфа-бета алгоритм нашел широкое применение в экспертных системах, ориентированных на интеллектуальные игры, например в шахматных программах.

Разбиение на подзадачи. При этой стратегии вывода в исходной задаче выделяются подзадачи, решение которых рассматривается как достижение промежуточных целей на пути к заключительной цели. Если удается правильно понять сущность задачи и оптимально разбить ее структуру на элементы иерар-

хически связанных целей-подцелей, то можно добиться того, что путь к ее решению в пространстве поиска будет меньшим. Однако если задача является плохо структурированной, то сделать это невозможно [6].

При сведении задачи к подзадачам производится исследование исходной задачи с целью выделения такого множества подзадач, чтобы решение некоторого определенного подмножества этих подзадач содержало в себе решение исходной задачи. Рассмотрим, например, задачу о проезде на автомобиле из Орши (Витебская область) в Березу (Брестская область). Эта задача может быть сведена, скажем, к следующим подзадачам:

Подзадача 1. Проехать из Орши в Жодино.
Подзадача 2. Проехать из Жодино в Минск.
Подзадача 3. Проехать из Минска в Столбцы.
Подзадача 4. Проехать из Столбцов в Барановичи.
Подзадача 4. Проехать из Барановичей в Березу.

Решение всех пяти подзадач обеспечило бы некоторое решение первоначальной задачи. Каждая из подзадач может быть решена с применением какого-либо метода: использующего пространство состояний, или анализа с целью выделения для каждой своих подзадач и т. д. Если продолжить процесс разбиения возникающих подзадач на еще более мелкие, то придем к некоторым элементарным задачам, решение которых может считаться тривиальным. Так как некоторые из этих множеств, возможно, не приведут к окончательному решению задачи, то для решения исходной задачи необходим поиск в пространстве множеств подзадач.

Представление задач в пространстве состояний. Чтобы построить описание задачи с использованием пространства состояний, нужно иметь представление о том, что представляют состояния в этой задаче. Важным этапом построения какого-либо описания задачи с использованием пространства состояний является выбор некоторой конкретной *формы* описания состояний этой задачи [6].

В целом любая структура величин может быть использована для описания состояний. Это могут быть строки символов, векторы, двумерные массивы, деревья и списки. Часто выбираемая форма описания имеет сходство с некоторым физическим свойством решаемой задачи. Выбирая форму описания состояний, нужно позаботиться и о том, чтобы применение оператора, преобразующего одно описание в другое, оказалось бы достаточно легким [6].

Операторы [6]. Для обсуждения методов поиска решения оказывается полезным введение понятий *состояний* и *операторов* для данной задачи.

Пространство состояний, достижимых из данного начального состояния, полезно представлять себе в виде графа, вершины которого соответствуют этим состояниям. Вершины такого графа связаны между собой дугами, отвечающими операторам. Шаги переводят одно состояние в другое, их можно рассматривать как функции, определенные на множестве состояний и принимающие значения из этого множества. Так как наши процессы решения задач основаны на работе с описанием состояний, то будем предполагать, что шаги − функции этих описаний, а их значения − новые описания.

В некоторых задачах оптимизации недостаточно найти любой путь, ведущий к цели, а необходимо найти путь, оптимизирующий некоторый критерий (например, минимизирующий число применений шагов или время поиска). С такими задачами надо работать так, чтобы поиск не оканчивался до тех пор, пока не будет найдено некоторое оптимальное решение. Видим, что для полного представления задачи в пространстве состояний необходимо задать:

а) форму описания состояний и описание начального состояния;

б) множество шагов и их воздействий на описания состояний;

в) свойства описания целевого состояния.

Запись в виде графа [6]. Граф состоит из множества (не обязательно конечного) *вершин*. Некоторые пары вершин соединены с помощью *дуг*, и эти дуги направлены от одного узла к другому. Эти графы носят название *направленных графов.* Если дуга направлена от вершины n_i к вершине n_j, то говорят, что вершина n_j является дочерней для вершины n_i, а вершина n_i является родительской вершиной для n_j. Если две вершины будут дочерними друг для друга; в этом случае пара направленных дуг называется иногда ребром графа. В случае когда граф используется для представления пространства состояний, с его вершинами связывают описание состояний, а с его дугами − операторы.

Последовательность вершин $n_{i1},n_{i2},...,n_{ik.}$, в которой каждая вершина n_{ij} дочерняя для $n_{i,j-1}$, $j = 2, k$, называется путем длины k от вершины n_{i1} к вершине n_{ik}. Если существует путь, ведущий от вершины n_i к вершине n_j, то вершину n_j называют достижимой из вершины n_i или потомком вершины n_i. В этом случае вершина n_i называется предком для вершины n_j. Проблема нахождения последовательности операторов, преобразующих одно состояние в другое, эквивалентна задаче поиска пути на графе [6].

4.7 Пример построения ЭС для заключения договоров

Рассмотрим ЭС «Посредник», служащую для заключения договоров между поставщиками и покупателями строительных материалов. Заключение договоров может производиться в обычном либо в интерактивном режиме. В первом случае система заключает контракт на основе имеющейся в БД информации о клиентах, во втором случае система запрашивает данные у пользователя, ищет информацию в БД и, найдя договоры, выводит их либо при отсутствии возможностей для сделки заносит информацию о клиенте в БД [3].

Концептуальная модель предметной области представляется множеством классов объектов с заданными на нем отношениями и операциями. В данном примере классами объектов являются Спрос, Предложение и Договор. Класс, Спрос имеет следующие атрибуты: порядковый номер, название фирмы, название товара, производитель желаемой партии товара, цена за единицу товара и срок поставки. У класса Предложение имеются такие же атрибуты, за исключением того, что вместо желаемой партии товара здесь присутствуют минимальная партия и максимальное количество товара, имеющееся у поставщика. Представим объекты классов Спрос и Предложение в виде таблиц 4.2, 4.3 [3; 4].

Таблица 4.2 − Объекты класса Спрос

№	Фирма	Товар	Производитель	Кол-во	Цена, $	Срок поставки
1	ДСК-3	Кирпич облицовочный	Российская	1000	0.33	6
2	Мосжилстрой	Керамзит	Европейская	30	22	8
3	Геракл	Арматура стальная	СНГ	15	51	10

Таблица 4.3 − Объекты класса Предложение

№	Фирма	Продукция	Производитель	Мин. партия	Макс. партия	Цена, $	Срок поставки
1	Стройсервис	Кирпич облицовочный	2-й кирпичный	120	900	0.25	7
2	Орион	Керамзит	Электроизолит	10	50	20	4
3	Салют	Арматура стальная	ММЗ	1	20	50	8

Представление знаний о клиентах. Данные о клиентах в системе «Посредник» представлены в виде стандартных баз данных (database) языка Пролог. Общий вид записей в базах данных следующий:

поставщик(*N, F, T, P, M, X, C, D*); покупатель(*N, F, T, P, K, C, D*),

где *N* − номер записи; *F* − фирма-поставщик; *T* – наименование товара; *P* – производитель товара; *M* – минимальная партия; *X* – имеющаяся в наличии партия; *C* – цена за единицу товара; *D* – срок поставки; *K* – нужное покупателю количество товара. *N, M, X, K, D* – целочисленные, *F, T, P* – строковые, C – действительная. База данных по поставщикам записана в файле kurs1.dat, по покупателям – в kurs2.dat.

Сетевая модель для представления знаний о клиентах. Она реализуется с помощью семантических сетей. В них имена объектов, процессов, действий, сущностей и их классов ассоциируются с узлами, а отношения между ними ассоциируются с дугами, соединяющими узлы.

Классифицирующие сети, строятся на основе родовидового отношения ***sup***, заданного на множестве классов объектов. Это отношение интерпретируется следующим образом: если K_i ***sup*** K_j, то в любой момент времени t каждый объект класса K_i является объектом класса K_j, т. е. K_j является подклассом K_i (или K_i является подклассом K_j). Конкретные объекты связываются с классами низшего уровня иерархии отношением принадлежности ***isa***. Запись k_i ***isa*** K.

В данном примере классифицирующая сеть используется для определения региональной принадлежности производителя строительных материалов. Благодаря этому при заключении договоров покупателю не обязательно задавать конкретного производителя, а можно задать только регион расположения тех производителей, продукция которых покупателю кажется наиболее предпочтительной. В системе «Посредник» имеется также возможность получения информации о принадлежности какого-либо производителя региону путем просмотра связей в классифицирующей сети, а также имеются режимы корректи-

ровки и дополнения этой сети. Программно классифицирующая сеть реализована в виде стандартных баз данных Пролога (database) и выглядит следующим образом:

sup(K_i,K_j), где K_i и K_j – класс и его подкласс соответственно. Классифицирующая сеть находится в файле web.dat.

Критерии выбора варианта договора. Выгода посредника состоит в получении наибольшей прибыли в кратчайшие сроки. Доход посредника обычно составляет некоторый комиссионный процент от заключенной сделки, поэтому наиболее выгодны контракты, в которых произведение количества товара, необходимое покупателю на цену этого товара, назначенную продавцом, будет максимальным. Это и будет первым критерием выбора наиболее предпочтительного договора. Вторым критерием, как было сказано выше, будет являться кратчайший срок поставки. Вначале программа выбирает предпочтительные договоры по первому критерию. Если таких договоров получится несколько, то программа выбирает из них тот, у которого срок поставки меньше (т. е. реализуется второй критерий). Например, пусть имеется три возможных договора для фирмы «АТС-50», которой требуется партия кабеля АВВГ российского производителя объемом 1000 единиц по цене 2$ за единицу в сроки не более 18 дней (таблица 4.4):

Таблица 4.4 − Договора фирмы АТС-50

№	Фирма	Товар	Производитель	Мин. партия	Кол-во	Цена	Сроки
1	Стройсервис	Кабель АВВГ	Иркутсккабель	900	000	1	1
2	Все для дома	Кабель АВВГ	ММЗ	900	000	1	2
3	Ункомтех	Кабель АВВГ	Иркутсккабель	700	200	0.9	

При применении первого критерия выбираются фирмы «Стройсервис» и «Все для дома», поскольку цена, по которой они предлагают товар, больше, чем у фирмы «Уникомтех» и, следовательно, доход посредника будет больше. На втором этапе из этих двух фирм выбирается «Стройсервис», поскольку у нее меньше срок поставки.

Логическая модель представления знаний на Прологе. При представлении логических моделей на Прологе классы сущностей предметной области интерпретируются как имена сортов. Для примера введем следующие имена сортов: Договор, Поставщик, Покупатель, Продукция, Производитель, Количество, Цена, Срок, Тип_Произв, а также следующие функции и предикаты:

1.Пост: Договор $\rightarrow$ Поставщик, покуп: Договор $\rightarrow$ Покупатель,
продук: Договор $\rightarrow$ Продукция, произв: Договор $\rightarrow$ Производитель,
кол: Договор $\rightarrow$ Количество, цена: Договор $\rightarrow$ Цена, срок: Договор $\rightarrow$ Срок
2.тип_произв: Производитель $\rightarrow$ Тип_Произв
3.российский: $\rightarrow$ Тип_Произв, европейский: $\rightarrow$ Тип_Произв
4. $\leq$:Количество $\times$ Количество $\rightarrow$ Т $\geq$:Количество $\times$ Количество $\rightarrow$ Т
Выражения 1–4 составляют сигнатуру и имеют следующий смысл:

1.Задает несколько функций, например прод (продукция), которые будучи применимы к объекту *e* сорта Договор, дают, например, продукцию прод(*e*), участвующую в операции *e*.

2.Задает функцию, значениями которой служат типы производителя.

3.Задает константы, принадлежащие сорту Тип_Произв.

4.Задает двухместные предикаты на объектах сорта Количество.

Сигнатура – это множество функций вида f: $A_1 \times A_2 \times \ldots \times A_n \rightarrow B$, где A_1, $A_2, \ldots, A_n$ – аргументы, B – значение функции. Множества аргументов и значений функций образуют соответственно сорта A и B. В частном случае, если $B = T$, причем $T = \{1,0\}$ – особый сорт, то сигнатура имеет вид P: $A_1 \times A_2 \times \ldots \times A_n \rightarrow T$, причем P называют предикатом.

Сигнатура задает структурные связи между понятиями предметной области, представленными предикатами и функциями, т. е. формально представляет одну часть знания о предметной области, а формулы, записанные в этой сигнатуре, представляют другую часть знания. Графическое представление сигнатуры показано на рисунке 4.3.

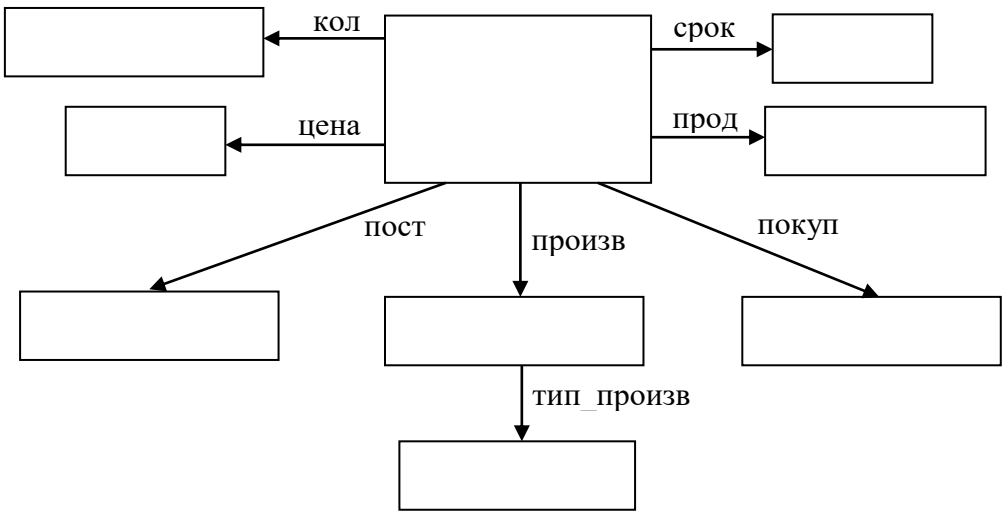

Рисунок 4.3 – Графическое представление сигнатуры

Логическая модель в Прологе представляется в виде предикатов и баз данных database. Например: database – договор

дог (Поставщик, Покупатель)

Функцию пост (поставщик) можно реализовать так:

пост (N):– дог (Покупатель, Поставщик), N=Поставщик.

Функцию $\leq$ можно представить следующим образом:

$\leq$(Количество1,Количество2):–Количество1<= Количество2.

Организация диалога с пользователем**.** При загрузке система «Посредник» выводит главное меню, в котором представлены все режимы работы программы. В программе реализованы режимы просмотра данных, их коррекции и удаления, наряду с обычным имеется интерактивный режим работы. Если в БД были внесены какие-либо изменения, то при выходе из программы система выведет запрос о необходимости сохранения изменений, сопровождаемый сигналом.

Выводы

По материалам главы сформулируем заключение.

1. Экспертная система (ЭС) − это набор программ, выполняющий функции эксперта при решении задач из некоторой предметной области. ЭС выдают советы, проводят анализ, дают консультации, ставят диагноз. Знания в ЭС представляются в такой форме, которая может быть обработана на компьютере. В ЭС известен алгоритм обработки знаний, а не алгоритм решения задачи. Качество ЭС определяется размером и качеством БЗ.

2. В ЭС существуют три типа знаний: структурированные знания − статические о предметной области, они не изменяются; структурированные динамические знания − изменяемые о предметной области, которые обновляются по мере выявления новой информации; рабочие знания, применяемые для решения задачи или проведения консультации. Все знания хранятся в БЗ.

3. Структурно ЭС включает: решатель, БЗ, подсистему объяснений, которые с одной стороны через интерфейс доступны пользователю для решения его задач. Через редактор знаний ЭС доступна эксперту и инженеру знаний для ввода и редактирования правил и фактов. Для построения ЭС используются языки Lisp, Prolog. Оболочка (shell) − «пустая» версия ЭС без наполненной БЗ, требующая работы эксперта и инженера по знаниям для ввода и корректировки знаний в данной предметной области с использованием ЯПЗ.

4. ЭС, которые могут только повторять логическое заключение эксперта, принимаются для отнесения к первому поколению. Если ЭС выступает в роли помощника и консультанта, способна анализировать нечисловые данные, выдвигать и отвергать гипотезы, оценивать достоверность фактов, самостоятельно пополнять базу данных, контролировать согласованность, делать выводы на основе прецедентов, генерировать решения новых, ранее не рассматривавшихся проблем, то она принадлежит к второму поколению.

5. При определении целесообразности использования ЭС следует руководствоваться следующими критериями: данные и знания надежны и не меняются с течением времени; пространство возможных решений невелико; в процессе решения проблемы следует использовать формальные рассуждения; должен быть эксперт, способный сформулировать свои знания и объяснить методы их применения для решения задач.

6. ЭС имеют определенные преимущества перед экспертом: у них нет предрассудков; они не делают поспешных выводов; работают систематически, учитывая все детали, часто выбирая наилучшую альтернативу из всех возможных; они устойчивы к «вмешательству», а эксперт использует необходимые знания и легко его находит под влиянием внешних факторов, которые напрямую не связаны с решаемой задачей; системы не заменяют специалиста, а являются инструментом в его руках.

7. Подсистема сбора знаний предназначена для добавления новых правил в BZ и модификации существующих, приведения правила к форме, позволяющей подсистеме вывода применять это правило в процессе работы, проверки введенных или измененных правил на соответствие существующим правилам.

BZ – это изменяемая часть системы, которая может быть пополнена и модифицирована инженерами знаний с учетом опыта использования ES между консультациями. Знания представлены в символической форме (списки, правила, факты и т. д.).

9. Машина вывода выполняет две функции: во-первых, просматривает существующие факты из рабочего набора и правила из базы знаний и добавляет новые факты в рабочий набор, а во-вторых, определяет порядок просмотра и применения правил. Цель ES состоит в том, чтобы вывести какой-то данный факт, то есть в результате применения правил гарантировать, что этот факт включен в рабочий набор, или опровергнуть этот факт, то есть убедиться, что его невозможно вывести.

10. Логический вывод может происходить многими способами, из которых наиболее распространенными являются прямой порядок вывода и обратный порядок вывода. Прямой порядок вывода – от фактов, которые находятся в рабочем наборе, к заключению. Обратный порядок вывода: выводы пересматриваются до тех пор, пока факты, подтверждающие один из них, не будут найдены в рабочей памяти или получены от пользователя. Эффективность конкретной стратегии вывода зависит от характера задачи и содержания базы знаний.

11. Механизм вывода включает в себя два компонента − один из них реализует вывод, другой управляет этим процессом. Компонент вывода выполняет первую задачу, рассматривая существующие правила и факты из рабочего набора и добавляя к нему новые факты при срабатывании некоторого правила.

12. Компонент управления определяет порядок применения правил и выполняет четыре функции: сравнение − образец правила сравнивается с имеющимися фактами; выбор − если в ситуации можно применить сразу несколько правил, то выбирается одно из них; запуск − если образец правила совпал с фактами из рабочего правила, то правило срабатывает; действие − рабочий набор может быть изменен путем добавления к нему заключения сработавшего правила.

13. Стратегии вывода включают поиск в глубину, поиск в ширину, разбиение на подзадачи и алгоритм альфа-бета. Суть поиска в глубину заключается в том, что при выборе следующей подцели в пространстве состояний уважение всегда по возможности отдается той, которая соответствует следующему, более подробному уровню описания задачи. При поиске по ширине система проанализирует все объекты, находящиеся на одном уровне пространства состояний, и только затем перейдет к объектам следующего уровня детализации.

14. В стратегии разделения на подзадачи подзадачи выделяются в начальной задаче, решение которой рассматривается как достижение промежуточных целей на пути к конечной цели. С помощью альфа-бета-алгоритма задача сводится к уменьшению пространства состояний путем удаления в нем ветвей, которые не являются перспективными для нахождения успешного решения.

15. В методе полной итерации вершины раскрываются в том порядке, в котором они построены. В методах поиска в глубину выявляются те вершины, которые были построены последними. Использование эвристической информации может значительно сократить объем поиска.

Вопросы для контроля

1. Поясните структуру ЭС.
2. Охарактеризуйте специалистов, работающих с ЭС.
3. Назовите особенности ЭС.
4. Поясните области применения ЭС.
5. Охарактеризуйте преимущества и недостатки ЭС по сравнению с человеком.
6. Определите этапы разработки ЭС.
7. Поясните механизм вывода в ЭС на простом примере.
8. Охарактеризуйте четыре стратегии вывода в ЭС.
9. Приведите примеры фактов и правил в БЗ ЭС.
10. Поясните два вида интерфейсов ЭС.

Литература, используемая в главе

1. Искусственный интеллект : справ. В 3 т. Т. 2. : Модели и методы / под ред. Д. А. Поспелова. – М. : Радио и связь, 1990. – 304 с. ; Т. 3. : Программные и аппаратные средства / под ред. Д. А. Поспелова. – М. : Радио и связь, 1990. – 363 с.

2. Рассел С. Искусственный интеллект, современный подход / С. Рассел, П. Норвинг. – М. : Вильямс, 2006. – 1408 с.

3. Вишняков В. А. Интеллектуальные системы в управлении / В. А. Вишняков. – Минск : Изд-во МИУ, 2010. – 364 с.

4. Информатика : учебник. / под ред. проф. Н. В. Макаровой – 3-е изд., перераб. – М. : Финансы и статистика, 2009. – 768 с.

4. Козлов, А. Н. Интеллектуальные информационные системы : учебник /А. Н. Козлов. – Пермь : Изд-во ФГБОУ ВПО Перм. ГСХА, 2013. – 278 с.

5. ДП 150906 ФАИТ Кадыров ША.txt [Электронный ресурс]. – Режим доступа: https://studizba.com/files/show/pdf/69251-1-vyvod-otcheta-na-pechat-prosmotr-po.html. – Дата доступа: 15.12.2021.

6. Сальников, Е. А. Методы формализации и автоматизации маркетинговых задач конкурентного анализа : дис. канд. эконом. наук : 08.00.05, 08.00.13 / Е. А. Сальников. – СПб., 2005. – 248 с.

7. Как работать в Visual Prolog [Электронный ресурс]. – Режим доступа: https://pro-prof.com/forums/topic/visual-prolog-how-to-work. – Дата доступа: 15.06.2024.

5. ОСНОВЫ НЕЙРОННЫХ СЕТЕЙ

5.1 Нейрокомпьютер и основы нейроинформатики

*Программа RWC***.** В начале 90-х годов появилась международная программа «Вычисления в реальном мире» (RWC − Real World Computing). Суть ее в том, чтобы дать вычислительным и управляющим системам возможность самостоятельно, без помощи человека воспринимать воздействия внешнего мира и действовать в нем. Авторы программы большую часть (до 30–40 % ее содержания) отводят исследованию естественных и созданию искусственных нейросетевых систем [4; 7; 9].

Нейробионический подход к проблеме ИИ основывается на использовании принципов работы мозга для конструирования интеллектуальных систем. Его привлекательность и перспективность обусловливаются тем, что на функциональном уровне нервная система обеспечивает недоступную для технических устройств способность живых существ адаптироваться в реальном мире, а на «технологическом» уровне уникальные возможности по быстродействию и надежности [7; 9].

Имитация работы мозга. Имитация затруднена различиями между конструкцией компьютера и организацией мозга. Из-за того, что когда одно устройство моделирует другое, сильно от него отличающееся, процесс моделирования протекает очень медленно. На компьютере не сложно моделируются формально-логические элементы мышления, но сложнее процессы распознавания.

Моделирование способности человека адаптироваться в изменяющихся и слабо формализованных условиях реального мира сопряжено со значительными сложностями. Притом, что уровень технологии в микроэлектронике позволяет превзойти по плотности упаковки вычислительных элементов и по экономичности энергопотребления нервную ткань. Эту возможность адаптироваться к постоянно изменяющимся внешним условиям необходимо обеспечить системам, претендующим на интеллектуальность [9].

Сформировалось новое научно-практическое направление − создание нейрокомпьютера, представляющего собой машину нового поколения, качественно отличающуюся от предыдущих отсутствием заранее созданных алгоритмических программ и способностью к самоорганизации и обучению. Основу нейрокомпьютеров составляют НС − иерархически организованные параллельные соединения адаптивных элементов − нейронов, которые обеспечивают взаимодействие с объектами реального мира. Основные отличия нейрокомпьютера от обычного компьютера [7; 9]:

− *параллельная работа* большого числа простых вычислительных устройств обеспечивает высокое быстродействие;

− НС *способна к обучению*, которое осуществляется путем настройки параметров сети;

− *высокая отказоустойчивость и помехоустойчивость* НС за счет того, что знания как бы «размыты» в ней и обрыв какой-то связи в общем случае не

является достаточным условием отказа, а устранение помех осуществляется за счет «скатывания» поступившего искаженного образа к ближайшему имеющемуся образцу с меньшим энергетическим уровнем;

− простое строение отдельных *нейронов* позволяет использовать новые физические принципы обработки информации для реализаций нейросетей.

Дальнейшее повышение производительности компьютеров связывают с параллельной обработкой. В отличие от микропроцессора, имеющего полный набор команд, каждый нейрон, из которых состоит нейросеть, представляет собой лишь простейший аналоговый преобразующий элемент.

Наиболее хорошие результаты использования нейросетей достигнуты при распознавании образов, построении ассоциативной памяти, при создании самообучающихся ЭС, при решении оптимизационных большой размерности. Возможны несколько *типов классификации* существующих НС [7; 9]:

− модели с учителем, которые для того чтобы удовлетворять поставленным критериям, требуют предварительного обучения перед включением их в реальную обстановку; сети, не требующие предварительного обучения, способные самообучаться в процессе работы;

− *по характеру распространения информации*: *однонаправленные* сети, в которых информация распространяется только в одном направлении от одного слоя элементов к другому; с обратной связью (ОС);

− *по характеру кодирования информации:* обрабатывающие двоичную информацию и сети; оперирующие с действительными числами;

− по способу преобразования входной информации: $X \neq Y$ − гетероассоциативная сеть, осуществляющая отображение входного вектора X в выходной вектор *У*; выходной вектор у представляет искаженный образ эталона $X > 0$; если X = У − такая сеть называется автоассоциативной.

В таблице 5.1 приведены виды НС по двум параметрам классификации.

Таблица 5.1 − Основные типы НС

Тип НС/вид связи	С учителем	Без учителем
Без обратной связи	Многослойные персептроны (аппроксимация функций, классификация)	Карты Кохонена (сжатие данных, выделение признаков)
С обратной связью	Рекуррентные аппроксиматоры (предсказание временных рядов, обучение в режиме онлайн)	Сети Хопфилда (ассоциативная память, классификация данных)

5.2 Нейрон, нейронные сети

Нейрон (искусственный) − элементарный преобразующий элемент, как составная часть НС состоит из элементов трех типов и выполняет две основные функции − взвешенное суммирование и нелинейное преобразование. Элементы

нейрона − умножители (синапсы), сумматор и нелинейный преобразователь. Синапсы осуществляют связь между нейронами, умножают входной сигнал (x_j) на число, характеризующее силу связи, − вес синапса (w_j). Сумматор выполняет сложение сигналов, поступающих по связям от других нейронов и входных сигналов (рисунок 5.1) [8; 9].

Нелинейный преобразователь выполняет нелинейную функцию одного аргумента − выхода сумматора. Таким образом, нейрон в целом реализует скалярную функцию векторного аргумента. Функционирование нейрона можно разбить на два такта (в соответствии с двумя его функциями) [9]:

1. В сумматоре вычисляется величина возбуждения, полученного нейроном, $s = \sum_{j=1}^{n} w_j x_j = (w, x)$ (для простоты смещение w_0 не учитывается).

2. Возбуждение пропускается через преобразующую (активационную) функцию f, в результате чего определяется выходной сигнал $y = f(s) = f(\sum_{j=1}^{n} w_j x_j)$.

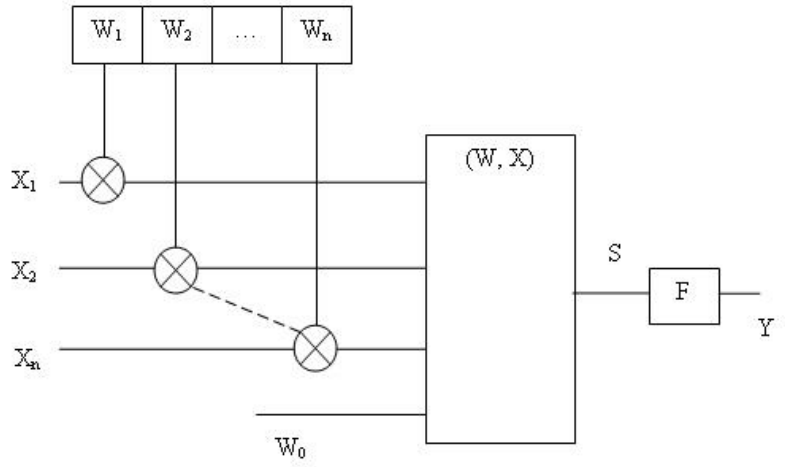

Рисунок 5.1 − Структурная схема нейрона

Наиболее часто используются следующие функции активации [9]:

1. *Пороговая* (ступенчатая) функция $f(s) = \begin{cases} 1, s > \theta, \\ 0, s \le \theta, \end{cases}$ где θ − порог срабатывания нейрона. Концептуально каждый нейрон может рассматриваться как правило «ЕСЛИ (w, x) > θ, ТО выдать импульс». Следует отметить, что, например, «небольшая» НС для распознавания образов может содержать десятки миллионов нейронов. Обработка БЗ, содержащей такое число правил, трудно реализуема даже на современных процессорах.

2. *Сигмоидная (логистическая)* функция. Функция называется *сигмоидной,* если она ограничена по минимальному и максимальному значениям и имеет везде положительную производную. Кроме этого, предполагается, что функция быстро сходится к верхнему пределу при $s \to +\infty$ и к нижнему при $s \to -\infty$.

Например, $f(s) = \frac{1}{1+e^{-as}}$, где a − параметр ($a > 0$), $0 < f(s) < 1$. График сигмоидной функции качественно близок к изображению передаточной характеристики биологического нейрона. Сигмоидная функция приближается к ступенчатой с порогом $\theta = 0$ при $a \to +\infty$.

3. *Гиперболический тангенс* $f(s) = \text{th}(as) = \frac{1-\exp(-as)}{1+\exp(-as)}$, $-1 < f(s) < 1$. Свойство нечетности тангенса, а также то, что $f(0)$, иногда оказывается очень удобным. Таким образом, каждый нейрон характеризуется вектором весовых множителей и параметрами преобразующей функции. Нейрон способен получать сигналы и в зависимости от их интенсивности и собственных характеристик выдавать выходной сигнал. При этом если выходной сигнал нейрона близок к единице, то нейрон возбужден [9].

Не ограничивая общности, можно считать, что нейроны в сети расположены слоями. Обычно выделяют входной слой, на который подается возбуждающий сигнал, выходной слой, с которого снимают переработанный сетью сигнал, а все остальные слои называют скрытыми (поскольку они не видны пользователю). Очевидно, что для адекватного решения задачи функционирования сети нужно правильно выбрать значения весов связей между нейронами – обучить сеть [9].

Выходной вектор, получающийся в результате работы нейросети, характеризует состояние элементов сети. Смысл, который имеют значения этого вектора, определяется в зависимости от решаемой проблемы. Решение задачи на нейрокомпьютере принципиально отличается от решения той же задачи на обычном компьютере. Решение задачи на обычном компьютере заключается в обработке вводимых данных в соответствии с разработанной и записанной в нее программой. Для составления программы необходимо разработать алгоритм.

Нейрокомпьютер же используется как «черный ящик», который можно обучить решению задач из какого-нибудь класса. Нейрокомпьютеру «предъявляются» исходные данные задачи и ответ, который соответствует этим данным. Нейрокомпьютер должен сам построить внутри себя алгоритм решения этой задачи, чтобы выдать ответ, совпадающий с правильным ответом. Естественно ожидать, что чем больше различных пар (исходные данные − ответ) будет предъявлено нейрокомпьютеру, тем более адекватную решаемой задаче модель он сформулирует [9].

Более того, если после этапа обучения нейрокомпьютеру предъявить исходные данные, которых он раньше не встречал, то желательно, чтобы он тем не менее выдал правильное решение. В этом заключается способность нейрокомпьютера к обобщению. Поскольку в основе устройства нейрокомпьютера лежит искусственная НС, то процесс обучения состоит в настройке параметров этой сети. При этом, как правило, топология сети считается неизменной, а к подстраиваемым параметрам обычно относятся параметры нейронов и величины синаптических весов. Обычно в литературе под обучением принято понимать процесс изменения весов связей между нейронами.

Классификация методов обучения НС следующая: по способу *использования обучения* [9]*:*

− с учителем, сети предъявляются примеры входных и выходных данных. Она преобразует входные данные и сравнивает свой выход с желаемым. После проводится коррекция весов с целью получить лучшую согласованность выходов с последовательным подкреплением знаний;

− без учителя − сети не дается желаемое значение выхода, а только ставится оценка, хороший выход или нет, НС вырабатывает правила обучения путем выделения оценочных признаков из набора входных данных;

по использованию элементов случайности:

− детерминированные сети − в них шаг за шагом осуществляется процедура коррекции весов сети, основанная на использовании их текущих значений, входов сети, выходов нейронов и некоторой дополнительной информации, например, значений желаемых выходов сети;

− стохастические сети − они основываются на использовании случайных изменений весов в ходе обучения.

5.3 Многослойные однонаправленные сети

Такие сети называются также сетями прямого распространения, или многослойными персептронами. Сети этого типа состоят из нескольких слоев нейронов: входного слоя, выходного и нескольких «скрытых слоев». На рисунке 5.2 изображена сеть, у которой *K* слоев. Нейроны каждого слоя не связаны между собой. Выходной сигнал с каждого нейрона поступает на входы всех нейронов следующего слоя. Нейроны входного слоя не осуществляют преобразования входных сигналов, их функция заключается в распределении этих сигналов между нейронами первого скрытого слоя.

Функционирование этой сети осуществляется следующим образом: входной сигнал, подаваемый на сеть, поступает на нейроны входного слоя, проходит по очереди через все слои и снимается с выходов нейронов выходного слоя. По мере распространения сигнала по сети он претерпевает ряд преобразований, которые зависят от его начального значения, от преобразующих функций и величин весов связей [7; 8; 9].

Пусть сеть состоит из *K* слоев: одного входного, одного выходного и (*K* >2) скрытых слоев, − каждый слой состоит из *n (k)* нейронов. Тогда прямое функционирование сети описывается следующими соотношениями: $y^1 = x$

$$y_j^k = f\left(\sum_{i=1}^{n(k-1)} w_{ij}^k y_i^{k-1}\right),\ j = 1 : n(k), \text{ где } y_j^k .$$

В основе методов обучения многослойных НС используется дельта-правило (при обучении с учителем) и реализуется следующим образом: $w_{ij}(t=1) = w_{ij}(t) = hx_i(d_j - y_j)$,

где *h* – параметр (шаг обучения); *d* − эталонное (требуемое) значение выхода элемента.

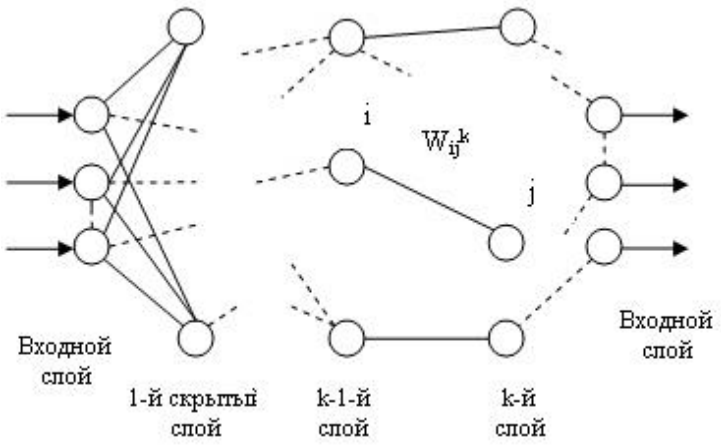

Рисунок 5.2 – Схема многослойной однонаправленной сети

Таким образом, изменение силы связей происходит в соответствии с ошибкой выходного сигнала $S = (d - y)$ и уровнем активности входного элемента x. Обобщение дельта-правила, называемое обратным распространением ошибки (back − propagation), применимо к сетям с любым числом слоев. Обучение сети состоит из следующих шагов:

1. Выбрать очередную обучающую пару (X, d). Подать входной вектор X на вход сети.

2. Вычислить выход сети Y.

3. Вычислить разность между выходом сети Y и требуемым выходом $Y - d$ (ошибку).

4. Подкорректировать веса сети так, чтобы минимизировать ошибку.

5. Повторять шаги с 1-го по 4-й для каждой обучающей пары, пока ошибка не достигнет приемлемого уровня.

Ошибка функционирования сети обычно определяется как

$$E = \frac{1}{2}\sum_{j=1}^{n(K)}(d_j - y_j)^2,$$

где $y_j = y_j^k$ выход сети.

Для уменьшения этой ошибки следует изменить веса сети по правилу

$$w^k(t=1) = w^k(t=1) - h \cdot \frac{\partial E}{\partial w^k}.$$

Если при прямом функционировании входной сигнал распространяется по сети от входного слоя к выходному, то при подстройке весов ошибка сети распространяется от выходного слоя к входному слою.

Область применения многослойных нейросетей обусловлена тем, что они аппроксимируют отображение $F : D \subset R^{n(1)} \to R^{n(K)}$, используя для этого предварительное обучение на наборах тренировочных данных (x_1, d_1), (x_2, d_2), ..., (x_L, d_L), где $d_e = F(x_e)$. Таким образом, сеть можно рассматривать как модель $y = \varphi(x)$ реального объекта $y = F(x)$.

5.4 Полносвязные сети Хопфилда

Сеть Хопфилда (рисунок 5.3) − однослойная сеть. Все нейроны связаны друг с другом связями w_{ij} , при этом сигнал с выхода некого нейрона может подаваться на его же вход и необязательно $w_{ij} = w_{ji}$. Поскольку сигнал с выхода каждого нейрона подается на входы всех остальных, входной вектор начинает циркулировать, преобразуясь по сети до тех пор, пока сеть не придет в устойчивое состояние (когда все нейроны на каждом следующем цикле будут вырабатывать тот же сигнал, что и на предыдущем) [7; 8; 9].

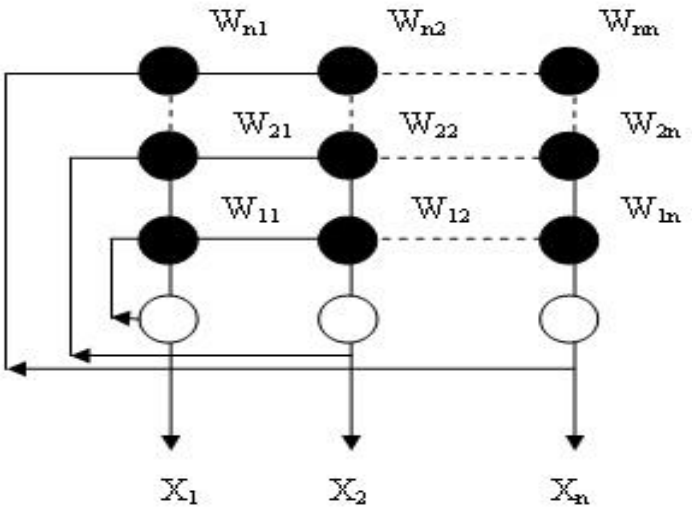

Рисунок 5.3 Полносвязная сеть Хопфилда

Возможны случаи бесконечной циркуляции входного вектора без достижения устойчивого состояния. Возьмем выходную функцию нейрона [9]:

$$x_j(t+1) = \begin{cases} 1, & \sum_{i=1}^{n} w_{ij} x_i(t) > \theta_j, \\ 0(-1), & \sum_{i=1}^{n} w_{ij} x_i(t) < \theta_j, \\ x_j(t), & \sum_{i=1}^{n} w_{ij} x_i(t) = \theta_j, \end{cases}$$

Состояние сети − множество текущих значений сигналов x от всех нейронов. Функционирование такой сети может быть представлено как продвижение вектора x, характеризующего состояние сети, в пространстве куба $[0,1]^n$. Когда поступает новый входной вектор, сеть Хопфилда переходит из вершины в вершину, пока не стабилизируется. Устойчивая вершина определяется сетевыми весами, текущими входами и величиной порога. Если входной вектор частично неполон, то сеть стабилизируется в вершине, ближайшей к нужной.

В целом все возможные состояния сети образуют картину холмистой поверхности, а текущие состояния сети аналогичны положениям металлического шарика, пущенного на эту поверхность, он движется вниз по склону в ближай-

ший нижний уровень. Каждая точка холмистой поверхности соответствует сочетанию активностей нейронов в сети, а высота подъема холмистой поверхности в этой точке характеризует «энергию» данного состояния. Энергия сочетания активностей определяется как сумма весов связей между парами активных нейронов, взятая со знаком минус [9].

Если связь между двумя некими нейронами имеет большой положительный вес, то сочетания, в которых эти нейроны активны, характеризуются низким уровнем энергии − именно к этому состоянию и будет стремиться вся сеть. Иначе, нейроны с отрицательной связью при активации добавляют к энергии сети большую величину, так что сеть стремится избегать подобных состояний. Динамику сети Хопфилда удобно описывать функцией энергии, которая может быть записана

$$E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} x_i x_j + \sum_{j=1}^{n} \theta_j x_j.$$

Функция энергии определяет устойчивость сети, по-другому − это функция Ляпунова сети Хопфилда, то есть функция, которая всегда убывает при изменении состояния сети. Эта функция должна достичь минимума и прекратить изменение, гарантируя тем самым устойчивость сети. Изменение состояния какого-либо элемента сети всегда вызывает уменьшение энергии сети [9].

Сети Хопфилда называются также сетями, минимизирующими свою энергию, они имеют многочисленные применения. Одни из них связаны со способностью запоминать различные образы, а затем восстанавливать их даже по неполной входной информации, другие связаны с возможностью использования этих НС для решения оптимизационных задач.

Двунаправленная ассоциативная память. Спецкомпьютер, построенный на базе нейросетей, обладает ассоциативной памятью и классифицирует поступившие образы со высокой скоростью, которая не зависит от количества поступивших образов, эта память связывает новый образ с ближайшим имеющимся. Как и сеть Хопфилда, двунаправленная ассоциативная память способна к обобщению, вырабатывая правильные реакции, несмотря на возможные искажение входа [8; 9].

В общем случае состояние нейронов можно представать как кратковременную память, поскольку она может изменяться при появлении другого входного вектора. Значения коэффициентов весовой матрицы для нее образуют долговременную память (ассоциации) и могут изменяться на более длительном отрезке времени, используя соответствующий метод обучения. Обучение производится с использованием обучающего набора из пар векторов x и y. Пусть все запомненные образцы представляют собой двоичные векторы. Решение задачи с помощью двунаправленной ассоциативной памяти можно разбить на два шага: режим обучения и распознавания. Рассмотрим эти шаги на примере [9].

Каждый нейрон a в первом слое A имеет синапсы, соединяющие его с нейронами b во втором слое B. Пусть нейроны имеют следующий «смысл»: a_1 − доллары, a_2 − доллары, a_3 − евро, a_4 − рубли, b_1 − США, b_2 – Россия, b_3 –Канада, b_4 – Германия.

Режим обучения бинарным образом. Подадим на нейросеть три бинарных вектора (x_1, y_1), (x_2, y_2), (x_3, y_3). Пусть

$x_1 = (1, 1, 0, 0) \rightarrow y_1 = (1, 1, 1, 0)$;
$x_2 = (1, 0, 1, 0) \rightarrow y_2 = (0, 1, 0, 1)$;
$x_3 = (1, 0, 0, 1) \rightarrow y_3 = (0, 1, 0, 0)$.

Смысл обучающих связей очевиден: если возбуждены нейроны a_1 и a_2 (в нашем распоряжении есть доллары), то по соответствующим синапсам возбудятся нейроны b_1, b_2, b_3 (мы можем ими воспользоваться в США, России и Канаде), и т. д. От бинарных связей перейдем к биполярным (это сделано исключительно для простоты, чтобы не нужно было вводить ненулевой порог срабатывания нейронов) [8]:
$x_1 = (1, 1, -1, -1) \rightarrow y_1 = (1, 1, 1, -1)$;
$x_2 = (1, -1, 1, -1) \rightarrow y_2 = (-1, 1, -1, 1)$;
$x_3 = (1, -1, -1, 1) \rightarrow y_3 = (-1, 1, -1, -1)$.

Составим матрицу весов:

$$W = \sum_{i=1}^{3} x_i^T y_i = \begin{pmatrix} -1 & 3 & -1 & -1 \\ 3 & -1 & 3 & -1 \\ -1 & -1 & -1 & 3 \\ -1 & -1 & -1 & -1 \end{pmatrix}.$$

Режим распознавания. Оценим эффективность запоминания обучающих связей. Убедимся, что матрица W хранит связи (x_1, y_1), (x_2, y_2), (x_3, y_3). Подадим на вход x_1, тогда $x_1 = (2, 2, 2, -2)$ − это означает, что в слое B возбудятся первые три нейрона (порог срабатывания принят равным нулю). Тогда в бинарной форме $y = (1, 1, 1, 0)$, что является требуемой ассоциацией. Это означает, что подача на вход x_1, приводит к y_1, то есть компьютер действительно «запомнил» связь (x_1, y_1). Аналогично проверяется запоминание остальных связей.

Сеть является двунаправленной: $y_1 W^T = (1, 5, -3, -3) \rightarrow (1, 1, 0, 0) \rightarrow x_1$, и т. д. Определим энергию связей в памяти:

$E(x_1, y_1) = -6$, аналогично $E(x_2, y_2) - 4$ и $E(x_3 y_3) = -2$. Следует ожидать, что при ошибке в исходной информации связь (x_1, y_1) будет притягивать к себе больше образов, так как это точка устойчивого равновесия c минимальным энергетическим уровнем. Подадим на вход вектор $x' = (1, 1, 0, 1)$ − искаженный на один бит x_1 и x_3, тогда $x' W = (1, 1, 1, -3) \rightarrow (1, 1, 1, 0) \rightarrow y_1$. Также, если взять $x» = (1, 0, 1, 1)$ − вектор, расположенный «между» x_2 и x_3, то получим $(-3, 1, -3, 1) \rightarrow (0, 1, 0, 1) \rightarrow y_2$ − связь (x_2, y_2), притягивает к себе, так как ее энергия меньше энергии (x_3, y_3).

Работа с неопределенными данными. Рассмотрим случай, когда тип валюты не определен x' − (1, 0, 0, 0), тогда x' W = (−1, 3, −1, −1) → (0, 1, 0, 0) → y_3. Это означает, что она может быть использована только в той стране, где имеет хождение любая валюта. Если валюта может быть любой, например, доллары и евро, то она может использоваться везде:

x' − (1, 1, 1, 0) → x'M = (1, 1, 1, 1) → y'.

Данное исследование означает, что сформированная нейросеть может запомнить нужную информацию на этапе обучения, а на этапе работы решает задачи распознавания, практически реализует функции ассоциативной памяти. Вся полученная при обучении информация сосредоточена в матрице W. За счет параллельной структуры сеть решает задачу «очень быстро» за одно действие – в процессе умножения входного вектора на матрицу памяти. Так как информация как бы интегрирована в матрицу W, сеть способна достаточно эффективно решать задачу и при частичных искажениях входных данных [8].

5.5 Самоорганизующиеся сети Кохонена

Важной особенностью человеческого мозга является то, что его структура отражает организацию внешних раздражителей, которые поступают в него от рецепторов. Так, известно, что положение рецепторов на поверхности кожи и нейронов в головном мозге, которые воспринимают и обрабатывают сигналы от них, находятся в соответствии. В связи с этим участки кожи, которые плотно «заселены» рецепторами (лицо, шея), связаны с пропорционально большим количеством нейронов. Данное положение формирует соматотропную карту, которая отражает поверхность кожи в участок мозга − соматосенсорную кору, которая воспринимает ощущение прикосновения [3; 4; 6; 8; 9].

Системы, построенные на основе многослойных сетей с обратным распределением ошибок, имеют следующий недостаток, который состоит в том, что нужно подготовить входные данные и определить для них правильные ответы для обучения сети, что не всегда возможно.

Принципы функционирования естественной соматотропной карты легли в основу создания самоорганизующихся сетей (maps, grids) Кохонена. Они воспринимают только входные данные и способны определять собственное восприятие внешних раздражителей. Самоорганизующиеся сети Кохонена − это карты или многомерные решетки, с каждым узлом которых связан входной весовой вектор, то есть набор из k входных весовых коэффициентов нейрона рассматривается как вектор в K-мерном пространстве. На рисунке 5.4 показана сеть при k = 2. Входной весовой вектор имеет размерность входа в сеть [8].

Обучение происходит в результате соперничества, возникающего между нейронами сети за право реагировать на поступающий входной сигнал. Сетевой элемент, выигравший в этом соревновании («победитель»), и его ближайшее окружение (кластер) изменяют веса своих входных ссылок. Перед обучением каждый компонент вектора входного веса инициализируется случайным образом. Как правило, каждый вектор нормализуется в единичной вектор в про-

странстве весов. Это выполняется путем деления его веса на корень из суммы квадратов составляющих этого вектора веса. Входные векторы нормализуются аналогичным образом. Сетевое обучение состоит из следующих этапов [8]:

1. Вектор x = $(x_1, x_2, ..., x_k)$подается на вход сети.

2. Определяется расстояние d_{ij} (в k-мерном пространстве) между x_k и весовыми векторами w_{ijk} каждого нейрона, например:

$$d_{ij} = \sqrt{\left[\sum_k (x_k - w_{ijk})^2\right]}.$$

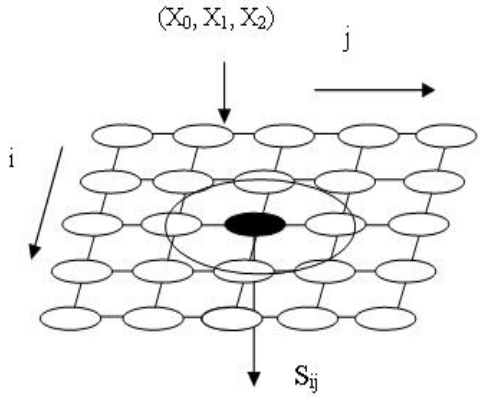

Рисунок 5.4 – Самоорганизующаяся карта Кохонена

3. Нейрон, который имеет весовой вектор, наиболее близкий к x, является «лучшим». Этот весовой вектор $w_{i^*j^*}$ становится главным в группе входных весовых векторов, которые лежат на расстоянии D от $w_{i^*j^*}$. Получаем группу (кластер) для главного вектора.

4. Группа входных весовых векторов модифицируется в соответствии с выражением $w_{ij}(t+1) = w_{ij}(t) + \eta(x - w_{ij}(t))$ для всех весовых векторов в пределах расстояния D от $w_{i^*j^*}$.

5. Шаги 1–4 повторяются для каждого входного вектора.

В процессе обучения значения D и η уменьшаются: $\eta : 1 \rightarrow 0$, D в начале обучения равняется максимальному расстоянию между весовыми векторами, а к концу обучения снизится до величины, при которой будет обучаться только один нейрон.

Из формулы адаптации входного весового вектора следует, что он (для «победителя» и его окружения) сдвигается по направлению к входному вектору. Получим, по мере поступления новых входных векторов весовые векторы сети разделяются на группы, формирующиеся в виде облаков (кластеров) вокруг входных векторов. В процессе обучения плотность весовых векторов будет выше в тех позициях пространства, где входные векторы появляются чаще, и наоборот. В результате сеть Кохонена адаптирует себя так, что плотность весовых векторов будет соответствовать плотности входных векторов. Пусть на вход сети подается поток равномерно распределенных случайных величин, тогда сетевые веса будут приводиться в регулярную структуру [8].

В общем случае $h = h(r, f)$, где r − расстояние между нейронами, f − параметр, характеризующий время обучения. Эта функция традиционно имеет вид «мексиканской шляпы» (зависимость силы связи от расстояния до победителя), которая по мере увеличения параметра f делается более узкой. Обучающий алгоритм настраивает входные весовые векторы в окрестности возбужденного нейрона таким образом, чтобы они были более «похожими» на входной вектор. Если все векторы нормализованы в векторы с единичной длиной, то они могут рассматриваться как точки на поверхности единичной гиперсферы. В процессе обучения группа соседних весовых точек перемещается ближе к точке входного вектора.

Очевидно, так как $S = \sum_k x_k w_k$, то победитель может быть найден так: $(i^*, j^*) = \arg\max(S_{ij})$ − наиболее активный нейрон. Самоорганизующиеся сети не требуют предварительного обучения на примерах и используются в задачах распознавания − классификации образов, представленных векторными величинами, в которых каждая компонента вектора соответствует элементу образа. После обучения подача входного вектора из данного класса будет приводить к возбуждению нейронов; тогда нейрон с максимальным возбуждением и будет представлять классификацию. Так как обучение проводится без указания целевого вектора, то априори невозможно определить, какой нейрон будет соответствовать данному классу входных векторов. Однако после обучения такие соответствия идентифицируются и могут быть использованы в задачах адаптивного управления.

5.6 Машинное обучение

Для машинного обучения программист не создает программу, содержащую решения. Вместо этого в программу подбирают алгоритм нахождения решений путем комбинированного использования статистических данных, из которых выводятся закономерности. Необходимо загрузить исходные данные, на которых нейронные сети будут учиться [7; 8].

Задачи, решаемые с помощью ML, относятся к одной из следующих категорий [8; 9]:

– регрессии – прогноз на основе выборки объектов с различными признаками. На выходе должно получиться вещественное число.

– классификации – получение категориального ответа на основе набора признаков;

– кластеризации – распределение данных на группы: разделение всех клиентов;

– уменьшения размерности – сведение большого числа признаков к меньшему;

выявления аномалий – отделение аномалий от стандартных случаев.

Задачи, решаемые при помощи методов машинного обучения, относятся к двум видам: обучение с учителем (supervised learning) либо без него (unsupervised learning). В обоих видах обучения машине предоставляются ис-

ходные данные, которые ей предстоит проанализировать и найти закономерности. Различие лишь в том, что при обучении с учителем есть ряд гипотез, которые необходимо опровергнуть или подтвердить.

Алгоритмы моделей машинного обучения [8; 9].

1. Дерево принятия решений. Метод поддержки принятия решений, основанный на использовании древовидного графа: модели принятия решений, которая учитывает их последствия (с расчетом вероятности наступления того или иного события), эффективность, ресурсозатратность.

2. Байесовские классификаторы основаны на теореме Байеса, которая рассматривает функции как независимые (строгое предположение).

3. Линейная функция обычно используется при подборе данных для машинного обучения, а метод наименьших квадратов – для сведения к минимуму погрешностей путем создания метрики ошибок.

4. Логистическая регрессия – это способ определения зависимости между переменными, одна из которых категориально зависима, а другие независимы. Для этого применяется логистическая функция.

5. Метод опорных векторов (SVM). Набор алгоритмов, необходимых для решения задач на классификацию и регрессионный анализ. Исходя из того что объект, находящийся в N-мерном пространстве, относится к одному из двух классов, метод опорных векторов строит гиперплоскость с мерностью (N – 1), чтобы все объекты оказались в одной из двух групп.

6. Кластеризация заключается в распределении множества объектов по категориям так, чтобы в каждой категории – кластере – оказались наиболее схожие между собой элементы.

7. Метод главных компонент (PCA) представляет собой статистическую операцию по ортогональному преобразованию, которая имеет своей целью перевод наблюдений за переменными, которые могут быть как-то взаимосвязаны между собой, в набор главных компонент – значений, которые линейно не коррелированы.

8. Сингулярное разложение (SVD) определяется как разложение прямоугольной матрицы, состоящей из комплексных или вещественных чисел. Так, матрицу M размерностью [m*n] можно разложить таким образом, что $M = U\Sigma V$, где U и V будут унитарными матрицами, а Σ – диагональной.

9. Анализ независимых компонент (ICA). Статистический метод, который выявляет скрытые факторы, оказывающие влияние на случайные величины, сигналы и пр. ICA формирует порождающую модель для баз многофакторных данных. Переменные в модели содержат некоторые скрытые переменные, которые являются независимыми компонентами выборки и считаются негауссовскими сигналами.

Диагностика заболеваний. Пациенты в данном случае являются объектами, а признаками – все наблюдающиеся у них симптомы, анамнез, результаты анализов, уже предпринятые лечебные меры (фактически вся история болезни, формализованная и разбитая на отдельные критерии). Некоторые признаки – пол, наличие или отсутствие головной боли, кашля, сыпи и иные – рассматри-

ваются как бинарные. Оценка тяжести состояния (крайне тяжелое, средней тяжести и др.) является порядковым признаком, а многие другие – количественными: объем лекарственного препарата, уровень гемоглобина в крови, показатели артериального давления и пульса, возраст, вес. Собрав информацию о состоянии пациента, содержащую много таких признаков, можно загрузить ее в компьютер и с помощью нейронной сети, способной к машинному обучению, решить следующие задачи:

– провести ИТ-диагностику (определение вида заболевания);
– выбрать наиболее оптимальную стратегию лечения;
– спрогнозировать развитие болезни, ее длительность и исход;
– просчитать риск возможных осложнений;
– выявить наборы симптомов, сопутствующие данному заболеванию.

5.7 Области применения нейроинформатики

Основное место на рынке услуг, оказываемых реальным потребителям средств нейроинформатики, занимают финансовые приложения. Объясняется это тем, что нейросети эффективно решают задачи классификации, моделирования и экстраполирования (прогнозирования, предсказания) на различных рынках, что крайне важно при решении финансово-экономических задач [4; 5; 8; 9].

НС служат основой для создания программных пакетов (имитаторов), плат-акселераторов для персональных компьютеров, нейроБИС, а также специализированных нейрокомпьютеров. Для отработки методологии решения задач в нейросетевой постановке начинающим пользователям оказывается достаточным использование соответствующих программных нейросетевых средств [9].

На мировом рынке представлено более сотни нейросетевых пакетов, преимущественно американских. Объем рынка нейронных сетей превышает 2,5 млрд долларов в год. Кроме того, практически каждый разработчик традиционных аналитических пакетов сегодня стремится включить НС в новые версии своих программ. В США НС применяются в аналитических комплексах каждого крупного банка.

Остановимся на характеристике популярного пакета «The AI Trilogy» («Трилогия ИИ») американской фирмы Ward Systems Group [4]. Пакет − набор из трех программ, каждая из которых может использоваться как самостоятельно, так и в комбинации с остальными: программа: NeuroShell II − набор из 16 типов нейронных сетей; NeuroWindows − нейросетевая библиотека с исходными текстами; GeneHunter − генетическая программа оптимизации.

Появление пакета «The AI Trilogy» на рынок России, Беларуси и СНГ, создание его русской версии соответствуют этапу использования аналитических систем для освоения нового класса пакетов, позволяющих решать комплексные задачи управления финансовыми ресурсами. Приведем только два примера по использованию этого пакета. Во время подготовки презентации русской версии «Трилогии» были взяты реальные данные о портфеле небольшой финансовой

компании из Калифорнии, играющей на рынке так называемых индексных опционов. Проведение средствами «Трилогии» классической последовательности «Анализ, прогноз, оптимизация» позволило в первый день «сделать» 25 тыс. долларов при величине портфеля в 2 млн долларов, во второй − добавить еще 40 тыс. долларов.

Преимущества нейронных сетей. Инновационное свойство НС − способность обучаться на многих примерах, когда неизвестны закономерности развития ситуации, и некие зависимости между входными и входным данными. В таких случаях (к ним можно отнести до 80 % задач финансового анализа) отступают как традиционные математические методы, так и экспертные системы по следующим причинам:

− НС способны успешно решать задачи, опираясь на неполную, искаженную, зашумленную и внутренне противоречивую информацию;

− для использования методов корреляционного, регрессионного и кластерного анализов понадобился бы профессионал-математик. Эксплуатация обученной НС по силам и начинающему пользователю;

− нейросетевые средства позволяют подключиться к БД, электронной почте, автоматизировать ввод и начальную обработку данных;

− внутренний параллелизм нейронных средств позволяет многкратно наращивать мощность нейрокомпьютерной системы. Можно начать с простого и дешевого пакета, потом перейти на профессиональную версию, потом добавить платы-ускорители, затем перейти на специализированный нейрокомпьютер с гарантией полной преемственности всего ранее созданное ПО.

В мире экономики НС широко применяются для двух основных задач − *прогнозирования котировок* основных инструментов (курсов валют, ценных бумаг, ГКО и др.) и *распознавания определенных ситуаций* (например, подозрительных операций с кредитной картой).

В России и Беларуси наиболее известными приложениями нейросетевых ИТ в области экономики и техники являются следующие: прогнозирование котировок фьючерсов; краткосрочная динамика курсов валют; прогноз оптовых цен на продукты питания; оценка кредитных рисков; оценка объектов недвижимости; ряд задач медицинской и промышленной диагностики; построение высокодоходных тотализаторов; прогноз развития чрезвычайных ситуаций; авторизация доступа по индивидуальному «почерку» работы за клавиатуре компьютера, системы защиты информации.

GPT. Это нейронная сеть для генерации текста, языковая модель, основанная на архитектуре трансформер и обученная в self-supervised режиме на многих текстовых данных. Имеется русскоязычная модель ruGPT3 от Сбербанка [11]. Языковое моделирование — это предсказание следующего слова (или куска слова) с учетом предыдущего контекста. Подбор модификаций текста называется «Prompt Engineering». Такая идея позволяет решать практически неограниченное количество задач.

Машинное обучение лучше справляется с числами, чем с текстом, поэтому необходима процедура токенизации – преобразование текста в последовательность

чисел. Проблема: слов и их форм очень много (миллионы) и поэтому словарь таких слов-чисел получится чересчур большим. Чтобы сжать словарь ещё сильнее, для обучения GPT OpenAI использовали byte-level BPE токенизацию. Эта модификация BPE работает не с текстом, а напрямую с его байтовым представлением. Использование такого трюка позволило сжать словарь до всего лишь ~50k токенов при том, что с его помощью все еще можно выразить любое слово на любом языке мира. Языковая модель выдает распределение вероятностей следующего токена, а эту информацию можно по-разному использовать для генерации текста.

Выводы

На основе материала главы с учетом источников [1–11] резюмируем.

1. Нейробионический подход к проблеме ИИ основан на использовании принципов работы мозга для конструирования ИС. Его перспективность обусловливается тем, что на функциональном уровне нервная система обеспечивает недоступную для технических устройств способность адаптироваться в реальном мире, а на «технологическом» уровне − большие возможности по быстродействию и надежности [9].

2. Основные отличия нейрокомпьютера от обычного компьютера: параллельная работа большого числа простых вычислительных устройств обеспечивает высокое быстродействие; сеть способна к обучению, которое осуществляется путем настройки ее параметров; высокая отказоустойчивость и помехоустойчивость сети за счет того, что знания как бы «размыты» в ней и обрыв какой-то связи не является условием отказа; простое строение отдельных нейронов позволяет использовать новые физические принципы обработки информации для аппаратных реализаций нейросетей [9].

3. Основными видами НС являются только три различных типа, большинство остальных распространенных нейросетей состоят из элементов, характерных для этих трех сетей прямого распространения (персептронов); полносвязных сетей Хопфилда; карт (решеток) Кохонена.

4. Нейрон − преобразующий элемент нейросети, состоит из элементов трех типов и выполняет две основные функции − взвешенное суммирование и нелинейное преобразование. Элементы нейрона − умножители (синапсы), сумматор и нелинейный преобразователь. Синапсы осуществляют связь между нейронами, умножают входной сигнал (x) на число, характеризующее силу связи, − вес синапса (w_j). Сумматор выполняет сложение сигналов, поступающих по синоптическим связям от других нейронов и внешних входных сигналов [9].

5. Возможны несколько типов классификации существующих нейросетей: по типу входной информации: сети, анализирующие двоичную и недвоичную информацию; по методу обучения: модели с учителем и самообучающиеся; по характеру распространения информации: однонаправленные и рекуррентные сети; по способу преобразования входной информации: ассоциативные; гетероассоциативные; по использованию элементов случайности: детерминированные и стохастические.

6. Многослойные сети состоят из нескольких слоев нейронов: входного, выходного и нескольких «скрытых слоев». Нейроны каждого слоя не связаны между собой. Выходной сигнал с каждого нейрона поступает на входы всех нейронов следующего слоя. Нейроны входного слоя не осуществляют преобразования входных сигналов, их функция заключается в распределении этих сигналов между нейронами первого скрытого слоя. Входной сигнал поступает на нейроны входного слоя, проходит по очереди через все слои и снимается с выходов нейронов выходного слоя.

7. Сеть Хопфилда − однослойная сеть. Все нейроны связаны друг с другом связями w_{ij} , причем сигнал с выхода нейрона может подаваться на его же вход и необязательно $w_i=w_{ji.}$ Поскольку сигнал с выхода каждого нейрона подается на входы всех остальных, входной вектор начинает циркулировать, преобразуясь по сети до тех пор, пока сеть не придет в устойчивое состояние (то есть когда все нейроны на каждом последующем цикле будут вырабатывать тот же сигнал, что и на предыдущем) [8].

8. Сеть Кохонена воспринимает только вход и способна вырабатывать собственное восприятие внешних стимулов − это карты или многомерные решетки, с каждым узлом которой ассоциирован входной весовой вектор, то есть набор из *k* входных весов нейрона трактуется как вектор в n-мерном пространстве. Входной весовой вектор имеет ту же размерность, что и вход в сеть. Обучение происходит в результате конкуренции, возникающей между узлами сети за право отклика на полученный входной сигнал [8].

9. С помощью нейронной сети, способной к машинному обучению, решить задачи: провести ИТ-диагностику (определение вида заболевания); выбрать наиболее оптимальную стратегию лечения; спрогнозировать развитие болезни, ее длительность и исход; выявить наборы симптомов, сопутствующие данному заболеванию.

10. Основное место на рынке услуг, оказываемых потребителям нейроинформатикой, занимают финансовые приложения. В финансовом мире НС широко применяются для двух основных задач − прогнозирования котировок основных инструментов (курсов валют, ценных бумаг и др.) и распознавания определенных ситуаций (подозрительных операций с кредитной картой). Также прогнозирование котировок фьючерсов; динамика курсов валют; прогноз оптовых цен на ресурсы, защита информации.

11. Преимущества НС: они способны успешно решать задачи, опираясь на неполную, искаженную, зашумленную и внутренне противоречивую информацию; эксплуатация обученной НС по силам неподготовленному пользователю; нейросетевые пакеты позволяют сравнительно легко подключиться к БД, электронной почте и т. д., автоматизировать процесс ввода и первичной обработки данных; внутренний параллелизм, присущий НС, позволяет наращивать мощность нейросистемы. Рассмотрены элементы машинного обучения и приведены краткие сведения по НС GPT.

Вопросы для контроля

1. Поясните понятие и классификацию НС.
2. Назовите отличия нейрокомпьютера от обычного компьютера.
3. Охарактеризуйте структуру и работу нейрона.
4. Поясните структуру персептрона.
5. Перечислите этапы обучения по методу Дельта.
6. Поясните особенности НС Хопфилда.
7. В чем суть работы ассоциативной памяти на НС.
8. Охарактеризуйте НС Кохонена.
9. Определите направления использования НС в технике.
10. Какие задачи решают НС в экономике?
11. Охарактеризуйте пакет AI Trilogy.
12. Поясните достоинства и недостатки НС.
13. Поясните суть машинного обучения.
14. Назовите алгоритмы машинного обучения.

Литература, используемая в главе

1. Рудковская, Д. Нейронные сети, генетические алгоритмы и нечеткие множества / Д. Рудковская, П. Пилинский, Л. Рудковский. – М. : Горячая линия, 2006. – 452 с.

2. Рассел, С. Искусственный интеллект, современный подход / С. Рассел, П. Норвинг. – М. : Вильямс, 2006. – 1408 с.

3. Вишняков, В. А. Интеллектуальные системы в управлении / В. А. Вишняков. – Минск : Изд-во МИУ, 2010. – 364 с.

4. Информатика : учебник для эконом. специальностей вузов / под ред. Н. В. Макаровой. – М. : Финансы и статистика, 1999. – 440 с.

5. Ежов, А. А. Нейрокомпьютеринг и его приложения в экономике и бизнесе / А. А. Ежов, С. А. Шумский. – М. : Изд-во МИФИ, 2005. – 440 с.

6. Нейронные сети, полный курс. – 2-е изд. – М. : Вильямс, 2006. – 1106 с.

7. Интеллектуальные технологии и системы [Электронный ресурс]. – Режим доступа: https://cyberpedia.su/7x1872.html. – Дата доступа: 12.01.2023.

8. Лекции по нейронным сетям [Электронный ресурс]. – Режим доступа: https://elar.urfu.ru/bitstream/10995/1404/7/1331983_lectures.pdf. Дата доступа: 12.01.2023.

9. Нейрокомпьютер и основы нейроинформатики [Электронный ресурс] / КиберПедия – Режим доступа: https://cyberpedia.su/7x1874.html. – Дата доступа: 12.01.2023.

10. GPT: от токенизации до файнтюнинга [Электронный ресурс]. – Режим доступа: https://habr.com/ru/articles/599673/. – Дата доступа: 12.05.2024.

6 ОСНОВЫ МНОГОАГЕНТНЫХ СИСТЕМ

6.1 Понятия многоагентных систем

Под многоагентными технологиями понимаются технологии разработки и использования многоагентных систем (МАС) и многоагентного управления (МАУ). Мультиагентный подход основан на введении и использовании мобильного программного агента, который реализуется и функционирует как самостоятельная специализированная программа или элемент системы искусственного интеллекта [1; 5].

В отличие от классического метода, когда осуществляется поиск определенного (детерминированного) алгоритма, позволяющего найти наилучшее решение задачи, в мультиагентных технологиях решение получается автоматически в результате взаимодействия множества независимых целевых программных модулей или агентов.

На рисунке 6.1 показаны две схемы построения программного обеспечения в сравнении: традиционная и основанная на многоагентной системе [5].

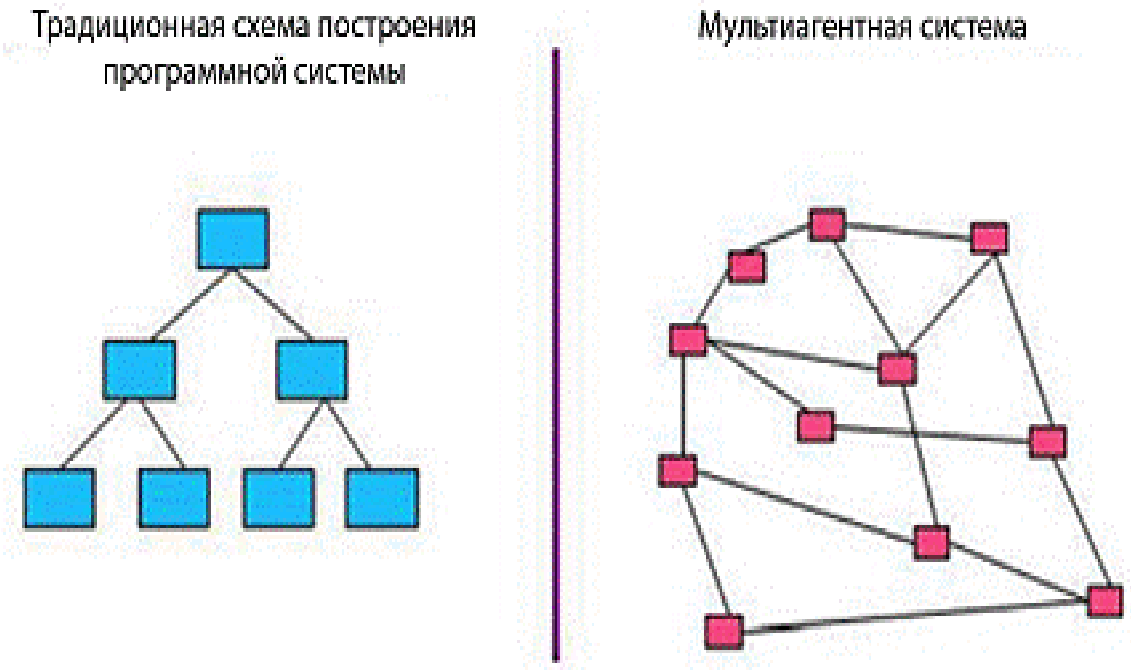

Рисунок 6.1 – Схемы построения программного обеспечения

Идея построения моделей и применения искусственных *МАС* зародилась в 1960-х годах. В качестве основы были взяты достижения таких областей науки, как теория искусственного интеллекта (*Artificial Intelligence*), параллельные вычисления (Parallel *Computing*), распределенное решение задач (*Distributed Problem* Solving) [2].

Примерами событий, вызывающих необходимость выявления особенностей и возможностей, являются: появление выгодного заказа, для выполнения которого своих ресурсов компании недостаточно, отказ части имеющихся ресурсов, а также изменение критериев принятия решений.

Чем выше неопределенность, чем более распределены процессы принятия решений и чем чаще происходят неожиданные события, тем ниже эффективность существующих систем, которые не способны самостоятельно принимать решения и автоматически приспосабливаться к изменениям в окружающей среде [2; 5].

Необходимость изменения схемы принятия решений в традиционных системах оказывается сложной и трудоемкой задачей, требующей высококвалифицированных исполнителей. Другой актуальной проблемой является рост объема информации и степени сложности распределенных систем систем, которые должны функционировать без человеческого управления [2; 5].

Агент – некая разумная сущность, помещенная во внешнюю среду, способная взаимодействовать с ней, выполняя автономные рациональные действия для достижения целей. Интеллектуальный агент – это агент со следующими свойствами [2; 5]:

– реактивность – агент воспринимает внешнюю среду и реагирует на изменения в ней, выполняя действия, направленные на достижение целей;

– проактивность (proactiveness) – агент демонстрирует целенаправленное поведение, проявляя инициативу, выполняя действия, направленные на достижение целей;

– социальность (способность) – агент взаимодействует с другими субъектами внешней среды (другими агентами, людьми и т. д.) для достижения некоторых целей;

– адаптивность – способность автоматически адаптироваться к неопределенным и изменяющимся условиям в динамичной среде.

Условия, которые влияют на построение и функционирование МАС [2; 5].

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

– агенты должны быть способны реагировать на изменяющиеся условия среды, в которой они работают, и изменять свое поведение на основе полученной информации;

– системы характеризуются возникающими структурами – логически связанными схемами, которые формируются в результате взаимодействия между агентами;

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

– при создании сложных систем на основе агентов имеет смысл учитывать биологические аналогии, такие как паразитизм, симбиоз, размножение, генетика и естественный отбор.

Концепция агентов, разработанная в рамках мультиагентных технологий и мультиагентных систем, предполагает наличие активного поведения агентов, т. е. способность компьютерной программы самостоятельно реагировать на внешние события и выбирать соответствующие действия.

6.2 Основные компоненты интеллектуального агента

Интеллектуальный агент должен иметь следующие свойства [1; 5; 6]:

– автономия – способность самому контролировать внутреннее состояние и свои действия;

– адаптивность – агент обладает способностью учиться при выполнении очередных действий;

– рассуждение – агенты могут обладать знаниями и механизмами вывода в определенной предметной области;

– коммуникация – агенты могут взаимодействовать с другими агентами;

– мобильность – возможность переноса кода агента с одного сервера на другой;

– реактивность – правильное восприятие окружающей среды и соответствующие реакции на ее изменения;

– активность – способность генерировать цели и действовать оптимальным образом для их достижения;

– интеллектуальность – знаний агента о себе, окружающей среде, включая других агентов;

– убежденность – переменная часть базовых знаний, которая может меняться и пополняться со временем;

– целенаправленность – набор состояний, на достижение которых ориентировано текущее поведение агента.

Основой архитектуры ИА является серверная среда, в которой каждый агент имеет постоянный идентификатор – его имя. В серверной среде может быть создан не только начальный агент, но и его копия. Агенты могут создавать свои собственные копии, отправляя их на различные серверы для выполнения дополнительной работы [2; 5].

Когда агент прибывает на очередной сервер, его код и данные переносятся в новый контекст и стираются в предыдущем месте. По завершении работы в контексте агент может перенаправить себя в другой контекст или на исходящий адрес отправителя. Агенты также могут завершать работу самостоятельно или по команде сервера, который затем переносит их из контекста в место, предназначенное для хранения [2].

На рисунке 6.2 показана усредненная структура типичного ИА. Входными данными являются внутренние параметры агента и данные о состоянии окружающей среды. Выходные данные – параметры, которые влияют на среду и информируют пользователя (или программу, выполняющую роль менеджера в системе) о состоянии среды и принятых решениях [5].

Решатель агента – это процедура принятия решений. Решателем может быть простой алгоритм или усложненный элемент системы искусственного интеллекта.

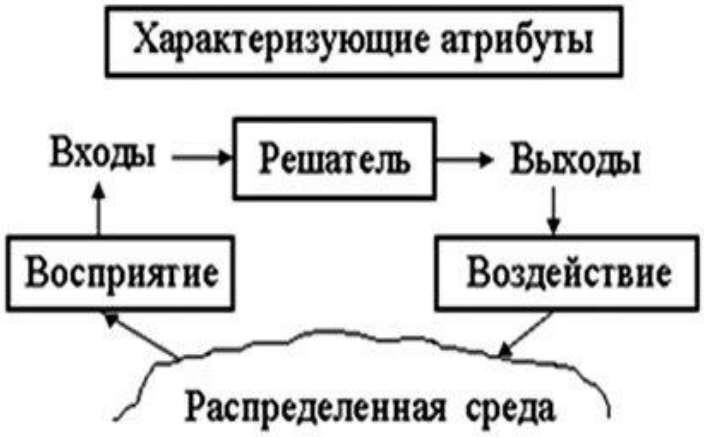

Рисунок 6.2 – Укрупненная структура агента

6.3 Архитектура МАС

Основной формой организации взаимодействия в МАС между агентами является взаимное целенаправленное сотрудничество, которое характеризуется объединением их действий для достижения общей цели при разделении функций, ролей и ответственности между ними, которое записывается в виде [2]:

сотрудничество = сотрудничество + координация действий + разрешение конфликтов.

Координация понимается как управление синхронизацией взаимдействия агентов. Связь между ИА зависит от выбранного протокола, который представляет собой набор правил, определяющих, как формировать нужные и корректные сообщения. Основными характеристиками группы, состоящей из агентов, сотрудничающих для достижения общей цели, являются социальная структура и распределение ролей между агентами. В МАС центр состоит из доменно-независимого ядра, включающего компоненты (рисунок 6.3) [2; 5]:

– служба прямого доступа обеспечивает прямой доступ к атрибутам агента;

– служба сообщений отвечает за передачу сообщений между самими агентами, а также между агентами и дополнительными системами ядра;

– библиотека классов агентов (часть базы знаний) содержит информацию о классификации агентов в этом МАС;

– сообщество агентов – серверная часть, где расположены агенты; этот блок, помимо жизнедеятельности агентов, также предоставляет функции для загрузки/записи агентов и их свойств и для оптимизации работы агентов с ресурсами;

Рисунок 6.3 – Архитектура ядра мультиагентной системы

– онтология – предметная база знаний, содержащая конкретные знания об объектах и среде функционирования, представленная в виде соответствующей семантической сети.

Методология восходящего проектирования МАС может быть представлена цепочкой [5]:

<окружающая среда – функции МАС – роли агентов – отношения агентов – базовые структуры МАС – модификации>.

Она включает в себя шаги [5]:

– формулирование цели (целей развития) МАС;

– определение основных и вспомогательных функций агентов в МАС;

– уточнение состава агентов и распределения функций между агентами, выбор архитектуры агентов;

– выделение основных взаимосвязей (relations) между агентами в МАС;

– определение возможных действий (операций) агентов;

– анализ реальных текущих или ожидаемых изменений во внешней среде.

Методология проектирования МАС по принципу снизу вверх требует предварительного распределения ролей агентов, определения круга их задач друг перед другом, формирования начальных и промежуточных структур на основе выбранных функций. Также исследования соответствия этих структур характеру решаемых задач в заданных проблемных областях [5].

Основная идея нисходящего проектирования состоит в том, чтобы определить общие социальные характеристики МАС в соответствии с определенным набором критериев, построить основные типы их организаций, а затем определить требования к архитектуре агентов. Когда речь заходит о проектировании искусственных социальных систем и сообществ, на первый план выходит нисходящий подход к организационному проектированию [5; 7].

6.4 Мультиагентное управление

Управление распределенными динамическими системами (задача на графах) относится к ситуации, в которой каждый узел может получать

информацию для формирования управления только для себя и от своих соседей. График может указывать топологию сети соединений, которая ограничивает соединения между узлами, – это называется многоагентным управлением.

Допустимые решения были получены только для ограниченного класса практических задач, поскольку решение таких задач достаточно усложняется, с одной стороны, из-за обмена неполной информацией, которая обычно поступает с помехами, с другой стороны, из-за эффектов дискретности выборки, что характерно для всех цифровых систем. Цели управления, которые решаются системами МАУ, включают синхронизацию и формирование [2–5].

Синхронизация – это совпадение или конвергенция переменных состояния двух или более систем или скоординированное изменение некоторых количественных характеристик систем. Задача синхронизации отличается от задачи управления с эталонной моделью, поскольку она допускает совпадение различных переменных, взятых в разные моменты времени. Временные сдвиги могут быть либо постоянными, либо иметь тенденцию к этому. Кроме того, во многих задачах синхронизации связь между системами является двусторонней (двунаправленной) [5].

Это означает, что предельный режим в системе (синхронное поведение) заранее неизвестен. Общей особенностью задач управления синхронизацией является то, что нужное поведение не фиксируется точно, а его характеристики задаются лишь частично.

В задачах синхронизации основным требованием не редко является совпадение или согласованность поведения всех подсистем, в то время как характеристики движения каждой подсистемы могут сильно различаться. В контексте МАУ синхронизация понимается как скоординированное поведение агентов, например, полное или частичное сходство с течением времени для состояний агентов или похожесть их выходных данных.

В МАУ цель управления скорее формулируется как согласованная задача, в которой агент стремится уменьшить отклонение своей целевой переменной от соответствующих переменных своих собратиев. Если речь идет об отклонении от среднего арифметического соседей, то задача состоит в том, чтобы достичь среднего консенсуса [5].

Другим классом задач МАУ является управление распределенным формированием – для локально взаимодействующих агентов, формирующим некоторые структурные конфигурации. Эти задачи решаются применительно к мобильным роботам, беспилотным летательным аппаратам, подводным автономным устройствам и т. д. Задачи управления группой сводятся к задачам частичной синхронизации координат с использованием метода центрального виртуального агента (лидера).

При большом количестве агентов (в отдельных задачах число агентов достигает тысячи и более) требование заданного поведения агентов оказывается излишне жестким и трудновыполнимым. В таких случаях

выделяется характерная точка во множестве состояний агента (центр, лидер, центр тяжести) и желаемое поведение – это заданное поведение центра при условии, что состояния всех агентов ограничены отклонениями от него [5; 6].

6.5 Создание агентов и платформ мультиагентных систем

Наиболее известные из международных стандартов в области создания МАС следующие [1–5]:

– OMG MASIF, созданный группой управления объектами, который основан на концепции мобильного агента;

– спецификации FIPA (Основы интеллектуальных физических агентов), основанные на предположении об интеллекте агента;

– стандарты, разработанные исследовательским подразделением Пентагона – Агентством перспективных оборонных исследовательских проектов (DARPA), Управление агентскими системами – CAS.

Что касается мобильности и интеллекта агентов, (по высказываниям большинства экспертов), то первым требованием является центральная характеристика агента, интеллект желателен, но не всегда точно необходим. Различия в подходах к определению агента в стандартах FIPA и OMG показаны на рисунке 6.4.

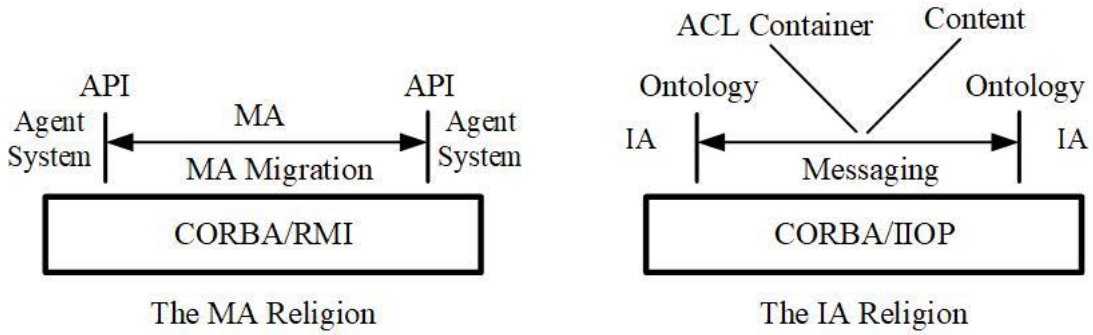

Рисунок 6.4 – Сравнение мобильных и интеллектуальных агентов

Деятельность FIPA заключается в совместных исследованиях и разработке членами организации согласованных на международном уровне спецификаций, которые позволят по-максимому сделать простым взаимодействие между агентскими приложениями, сервисами и оборудованием. Членами FIPA являются известные компании: Alca-tel, Boeing, British Telecom, Deutsche Telekom, France Telecom, Fujitsu, Hitatchi, HP, IBM, Fujitsu, Hewlett Packard, IBM, Intel, NEC, NHK, NTT, Nortel, Siemens, SUN, Telia, Toshiba, а также различные университеты, правительственные организации [5].

Спецификации FIPA ориентированы на обеспечение возможности взаимодействия интеллектуальных агентов посредством стандартизированной коммуникации агентов и языков контента. Наряду с общими основами

коммуникации FIPA также специализируется на онтологии и протоколах согласования для поддержки взаимодействия в конкретных прикладных областях сети [5].

Стандарт OMG MASIF направлен на создание условий для миграции мобильных агентов между компьютерами MAC через стандартизированные интерфейсы CORBA IDL.

Организация DARPA инициировала работу по обмену знаниями, в результате которой языки агентного программирования были разделены на синтаксис, семантику и прагматику, например [2]:

– KIF – формат обмена знаниями (синтаксис);

– Ontolingua – язык для определения совместно используемых онтологий (семантика);

– KQML (Knowledge Query and Manipulation Language) – язык взаимодействия высокого уровня (прагматика).

Определяющим элементом при создании многоагентных систем является язык взаимодействия агентов – Agent Communication Language (ACL), который определяет типы сообщений, которыми агенты могут обмениваться. В рамках парадигмы коммуникации между агентами сотрудничество между ними достигается с помощью ACL, языка контента и онтологии, которые определяют набор базовых понятий, используемых в сообщениях о сотрудничестве [2].

Онтология является синонимом понятия API (интерфейс прикладного программирования), т. е. определяет конкретный интерфейс интеллектуальных агентов. На техническом уровне связь между агентами происходит посредством передачи сообщений с использованием одного из транспортных протоколов более низкого уровня (SMTP, TCP/IP, HTTP, IIOP) [2; 5; 7].

Альтернативами использованию ACL является ряд других языков, таких как языки БД (SQL), распределенные объектные системы (CORBA и т. д.), языки обслуживания (e-speak от Hewlett Packard, BizTalk от Microsoft и т. д.), веб-языки описания данных и знаний (XML, RDF, DAML) [2; 5].

Альтернативой ACL также является CORBA ORB, разработанная Object Management Group. Вся функциональность, предоставляемая CORBA, также доступна на JAVA, благодаря комбинации Java RMI, Java RMI servers, Jini, Java eventservers и других [2].

Языки общения агентов развиваются. Поскольку совместимость является определяющей характеристикой агентов, стандартизированная коммуникация необходима при разработке MAC. Основными объектами для стандартизации агентов являются: архитектура, языки взаимодействия, протоколы взаимодействия, знания, языки программирования агентов [5].

Технология создания агентов включает функции [2; 5]:

– разработка семантики языков коммуникации агентов (ACL) (языки общего контента и онтологии; языки для описания действий, намерений и стремлений агентов);

– разработка онтологии агентов (свойства агентов и поведение);

– улучшенное использование метаданных (абстрактных и совместимых со многими языками контента);

– декларативные протоколы (языки для определения протоколов высокого уровня на основе примитивных);

– практический обмен знаниями между агентами (рассматривая обмен знаниями как мобильный код);

– разработка схем и методов управления агентными системами (искусственные рынки, естественный отбор и т. д.).

Агентные платформы – это инструментарий для построения распределенных систем, они позволяют описывать и предоставлять доступ ко всем приложениям, распределению агентов, контролю их функционирования и к управлению. Существует несколько агентских платформ, ориентированных на использование спецификации FIPA-2000 [5].

Платформа агента (согласно стандарту FIPA) имеет следующую структуру (рисунок 6.5). Система управления агентами (СУА) также является агентом, который контролирует доступ и использование платформы агента.

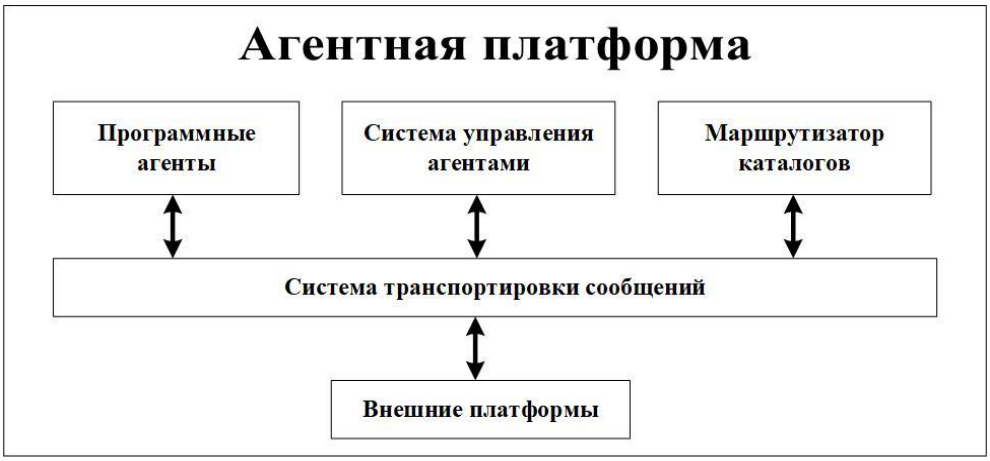

Рисунок 6.5 – FIPA-модель агентной платформы

В каждой агентной платформе присутствует одна СУА, которая предоставляет сервис жизненного *цикла* программных агентов и их реестр с идентификаторами, а также содержит состояния каждого программного агента. Маршрутизатором каталога является программный *агент*, который обеспечивает направление запросов в другие агентные платформы. Система транспортировки сообщений, это канал коммуникации агентов, является программным компонентом для управления потоками сообщений, приходящих на агентную платформу [2; 5].

6.6 Разработка мультиагентных систем и их применение

Применяемыми инструментами разработки для МАС являются следующие инстументальные средства [1–5]:

– JADE (Java Agent Development Framework) – программная среда для создания мультиагентных систем и приложений, поддерживающая стандарты FIPA построения интеллектуальных агентов. Среда включает в себя область выполнения агента, библиотеку классов, которые используются для разработки агентных систем, набор графических утилит для администрирования и мониторинга деятельности агентов. JADE подключается к проекту на языке Java.

– JACK Intelligent Agents – Java-платформа для создания мультиагентных систем. Расширяет средства Java с помощью встроенных классов. JACK – одна из платформ, использующая модель агентской логики, основанную на убеждениях–желаниях–намерениях (Belief–desire–intention – BDI), и встроенные формальные логические инструменты для планирования работы агентов.

– MadKit – модульная и масштабируемая платформа Java. Поддерживает агенты на языках: Java, Python, Jess, Scheme, BeanSchell. Визуализирует и позволяет вам управлять агентами.

– AgentBuilder – рыночный продукт, который доступен в академическом издании. Агенты интеллектуальны, общаются на языке KQML (язык запросов знаний и манипуляций ими) и имеют ментальную модель. Платформа ориентирована на язык Java.

– Cougaar (Cognitive Agent Architecture) – Java-ориентированная платформа для создания распределенных компонентов МАС. Она включает в себя исполняющую систему (run-time engine) и инструменты для визуализации, управления данными и т. д.

– NetLogo – кроссплатформенная программируемая среда для программирования для создания и моделирования МАС.

– VisualBots – это бесплатный мультиагентный симулятор в Microsoft Excel с синтаксисом Visual Basic.

– MASON – Java-библиотека для моделирования мультиагентных систем.

– REPAST – набор инструментов для создания агентных систем.

МАС может использоваться как для проектирования и моделирования гибких производственных систем, так и для управления реальными производственными или транспорными системами (логистика), для продажи продуктов различного назначения (электронная коммерция), интеграции и управления знаниями, а также научной работы [5].

Большое значение в мультиагентном подходе имеет социальный аспект решения современных проблем как его концептуальная основа. Такие системы должны постоянно находиться на корпоративном сервере и длительно участвовать в решении задач, а не запускаться время от времени. Для этой возможности – предоставлять пользователю периодически внедрять новые данные и компоненты.

Данные системы должны накапливать информацию, извлекать из нее новые знания и в зависимости от ситуации изменять свое поведение с течением времени. МАС позволяют автоматизировать полный цикл управления мобильными ресурсами в режиме реального времени, включая функции [5; 6; 7]:

– оперативное реагирование на важные события;

– динамическое планирование и адаптивное перераспределение заказов/ресурсов;

– взаимодействие с клиентами, менеджерами для координации решений через сеть и сотовый телефон;

– мониторинг исполнения построенных планов и бизнес-процессов заказчика;

– перепланировка графиков в случае несоответствия между планом и фактом.

В таких системах агенты способны взаимодействовать друг с другом в виде переговоров и демонстрировать коллективный разум (как в социуме пчел).

При разработке МАС используются базовые принципы самоорганизации и эволюции, присущие живым системам, характеризующимся способностью решать сложные задачи в режиме реального времени, открытостью к изменениям, высокой эффективностью, надежностью и живучестью.

Для МАС использовалась инструментальная платформа, состоящая из основных модулей (рисунок 6.6), включающая следующие модули [5; 7].

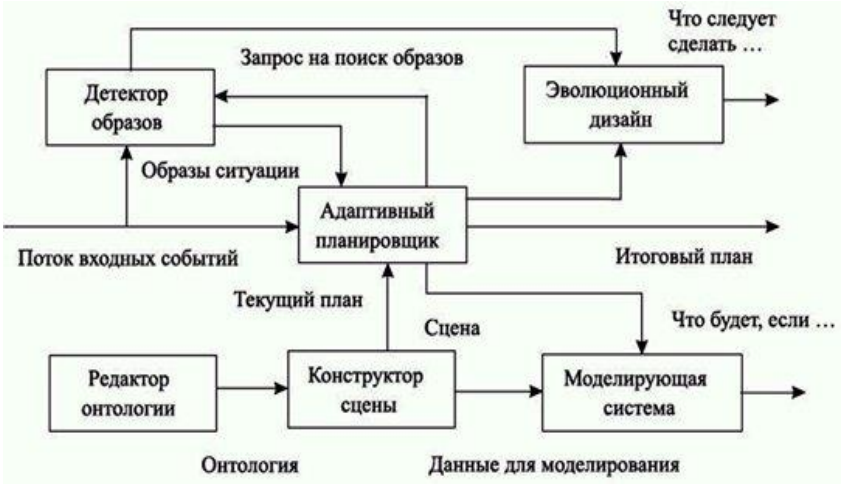

Рисунок 6.6 – Схема взаимодействия модулей инструментальной МАС

– Детектор изображений – распознает типовые ситуации во время приема заявок и вырабатывает прогноз заявок и рекомендации по планированию.

– Адаптивный планировщик – обрабатывает поток входящих событий (ошибки, ввод ресурсов, сбой ресурсов и т. д.).

– Конструктор сцен, позволяет редактировать входную конфигурацию сети и определять все параметры ресурсов организации. Он основывается на общей базе знаний (онтологии), описывающей деятельность компании, в которой существуют основные понятия и взаимосвязи между ними.

– Редактор онтологий, позволяет вводить и изменять онтологию компании, описывающую модель знаний предметной области, которая используется в сетевом редакторе для описания бизнес-конфигурации.

Онтология содержит базовые знания и взаимосвязи между ними, представленные в виде семантической сети.

– Система моделирования – программный модуль, позволяющий моделировать ситуации по принципу «Что, если? ».

– Модуль эволюционного проектирования, который предлагает улучшения конфигурации сети: увеличение или уменьшение количества ресурсов, изменение географии ресурсов и т. д.

МАС в настоящее время используются при создании интеллектуальных систем управления движением без водителя (машины, автобусы и т. д.).

6.7 Агенты в управлении инфокоммуникациями

В рамках модели взаимодействия открытых систем в ISO была разработана модель управления распределенными информационными системами. Модель предлагает архитектуру управления сетью, построенную на принципе «менеджер – агент», и определяет пять концептуальных областей управления [1–5].

Менеджер – это программно-аппаратный комплекс, который выдает команды управления и принимает сообщения об их выполнении. Программно-аппаратный комплекс или установленное на управляемом сетевом элементе программное приложение, которое выполняет команды и посылает сообщения о результатах операций, называется агентом. Команды, которые менеджер может отдавать агенту, а также ответные сообщения определяются посредством протокола управления. Вид архитектуры управления сетью с использованием агентов представлен на рисунке 6.7. Концептуальные области определяют задачи управления: конфигурацией сети, неисправностями, расчетами за услугами связи, безопасностью, производительностью.

Рисунок 6.7 – Архитектура управления сетью

В сетевых объектах управления реализован специальный программный модуль, который называется агентом. Агенты собирают информацию об управляемых устройствах, в которых они работают, и делают эту информацию доступной для систем управления сетями (network management systems – NMS) с помощью протокола SNMP.

Выводы

На основе материала главы с учетом источников [1–7] резюмируем.

1. Под многоагентными технологиями понимаются технологии разработки и использования многоагентных систем (МАС) и многоагентного управления (МАУ). Мультиагентный подход основан на концепции мобильного программного агента, который реализуется и функционирует как самостоятельная специализированная программа или элемент системы искусственного интеллекта.

2. Концепция агентов, разработанная в рамках мультиагентных технологий и мультиагентных систем, предполагает наличие активного поведения агентов, т. е. способность программы (устройства) самостоятельно реагировать на внешние события и выбирать соответствующие действия, принимать решения и взаимодействовать.

3. Наиболее известные из международных стандартов в области МАС: OMG MASIF, созданный группой управления объектами, который основан на концепции мобильного агента; спецификации FIPA (основы для интеллектуальных физических агентов), основанные на предположении об интеллекте агента.

4. Организация DARPA инициировала работу по распространению знаний (Knowledge Sharing Effort), в результате которой языки программирования агентов были разделены на синтаксис, семантику и прагматику: KIF – формат обмена знаниями (синтаксис); Ontolingua – язык для определения совместно используемых онтологий (семантика); KQML (Knowledge Query and Manipulation Language) – язык взаимодействия высокого уровня (прагматика).

5. Популярные инструментальные средства разработки МАС: JADE (Java Agent Development Framework) – программная среда для создания мультиагентных систем и приложений, поддерживающая стандарты FIPA для интеллектуальных агентов; JACK Intelligent Agents – платформа Java для создания мультиагентных систем, расширяющая Java своими классами. JACK – это одна из платформ, использующая модель агентской логики, основанную на убеждениях–желаниях–намерениях и встроенных формально-логических средствах планирования работы агентов.

6. МАС могут использоваться для проектирования и моделирования гибких производственных систем, для управления реальными системами в сетях, продажи продуктов различного назначения (электронная коммерция), управления движением без водителя.

7. В МАУ цель управления чаще формулируется как задача *достижения консенсуса*, в которой *агент* стремится уменьшить отклонение своей целевой переменной от соответствующих переменных своих соседей. Если речь идет об отклонении от среднего арифметического состояний соседей, то ставится задача достижения *усредненного консенсуса*.

8. В рамках модели взаимодействия открытых систем в ISO была разработана модель управления распределенными системами. Модель предлагает архитектуру управления сетью, построенную на принципе «менеджер – агент», и определяет пять концептуальных областей управления.

Вопросы для контроля

1. Поясните понятие агента.
2. Назовите свойства агента.
3. Охарактеризуйте структуру и работу агента.
4. Поясните структуру ядра МАС.
5. Что включает методология восходящего проектирования МАС?
6. Поясните особенности интеллектуального агента.
7. В чем суть спецификации FIFA об интеллектуальности агента?
8. Языки программирования агентов были разделены на … ?
9. Определите направления использования МАС.
10. Что присутствует в каждой агентной платформе?
11. Охарактеризуйте мультиагентное управление.
12. Поясните достоинства и недостатки МАС.
13. Поясните использование агентов для управления в инфокоммуникационных сетях.

Литература, используемая в главе

1. Ватсон, Б. Мультиагентные системы на примерах / Б. Ватсон. – СПб. : БХВ – Санкт-Петербург, 2011. – 608 с.

2. Кияев, В. ИТ в современном менеджменте / В. Кияев, О. Граничин. – М. : ИНТУИТ, 2016. – 171 с.

3. Башлыков, А. А. Решатели интеллектуальных задач для человеко-машинных систем поддержки принятия решений / А. А. Башлыков // Автоматизация, телемеханизация и связь в нефтяной промышленности. – 2013. – № 10. – С. 6–20.

4. Гольдштейн, Б. С. Сети связи пост-NGN / Б. С. Гольдштейн, А. Е. Кучерявый. – СПб. : БХВ-Петербург, 2013. – 160 с.

5. Разработка приложений для мобильных интеллектуальных систем на платформе Intel Atom / К. С. Амелин [и др.]. – Киев, 2016. – 95 с.

6. Голенков, В. В. Семантическая технология компонентного проектирования систем, управляемых знаниями / В. В. Голенков, Н. А. Гулякина // ОСТИС-15, материалы 5-й НТК Минск, БГУИР, 2015. – С. 57–78.

7. Разработка WEB-представительств для систем электронной коммерции [Электронный ресурс]. – Режим доступа: http://elibrary.ru/item.asp?id=_23427691. – Дата доступа: 15.12.2022.

ЧАСТЬ 2
ПРИМЕНЕНИЕ ИНТЕЛЛЕКТУАЛЬНЫХ ТЕХНОЛОГИЙ В ИНФОКОММУНИКАЦИОННЫХ СИСТЕМАХ

7 ИНТЕЛЛЕКТУАЛИЗАЦИЯ БИЗНЕСА В ИНФОКОММУНИКАЦИОННЫХ КОМПАНИЯХ

7.1 Понятие и назначение интеллектуальных проектов

Под интеллектуализацией бизнеса (Business Intelligence − BI) понимается изменение информации о состоянии бизнеса, носящее не оперативный, а аналитический (стратегический) характер. Термином BI принято также обозначать ИТ (программные средства) подготовки и доставки такой информации аналитикам (лицам, принимающим решения). BI − это глобальная система интеграции прикладных корпоративных информационных систем (КИС), основанная на общем хранилище данных. Задачи, которые решают системы BI [1; 3; 5]:

− добыча, структуризация содержащихся в оперативных системах данных и их преобразование в полезную информацию и знания;

− построение единого информационного пространства, проведение анализа и извлечение полезной информации;

− мониторинг деятельности и построение корпоративной отчетности;

− исследование данных, прогнозирование и выработка решений;

− управление рисками и улучшение качества бизнес-процессов;

Мировой рынок систем BI оценивается в 65–70 млрд долларов. Лидирующие на рынке компании используют BI-продукты, чтобы получить более четкое представление о выполнении внешних и внутренних операций (о клиентах, о цепочке поставок и финансовых результатах деятельности). Использование BI-средств позволяет разрабатывать более удачные планы и тактики; быстрее реагировать на любые непредвиденные ситуации; извлекать выгоды из появляющихся новых возможностей.

Рассмотрим примеры об успешных BI-решениях. По оценкам одной крупной авиакомпании, ей удалось повысить доход на 40 млн долларов и сократить расходы на 31 млн за счет использования только четырех из 35 приложений, входящих в состав корпоративной BI-среды. Крупный дистрибьютор электроприборов сэкономил 9 млн долларов благодаря сокращению расходов и получению дополнительной прибыли в результате использования корпоративного BI-проекта. Далее ожидается окупаемость порядка 90 млн долларов.

BI-среда. В BI-среду поступают данные, которые затем перерабатываются в действия и планы, знания, информационные ресурсы (рисунок 7.1) [5].

От данных к информации. Хранилище данных получает данные из многих источников, интегрирует их и хранит данные в специализированных базах. Так, в хранилище могут приводиться в соответствие и объединяться в один файл пользовательские записи из пяти оперативных систем (обработки заказов, обслуживания, продаж, поставок и вкусов потребителя).

От информации к знаниям. Специалисты, работающие с аналитическими инструментами обращаются к информации из хранилища и анализируют ее. Удается выявить тенденции и узкие места. Аналитические инструменты помогают пользователям преобразовать информацию в знания.

Рисунок 7.1 − BI-среда

От знаний к правилам. Получившие знания специалисты могут разрабатывать правила на основе обнаруженных тенденций и новых структур. Правила могут быть простыми («Заказать 50 новых единиц, если на складе осталось менее 25»). Кроме того, могут применяться правила продукции «Если А, То В», опирающиеся на тенденции и предположения [5].

От правил к планам и действиям. Специалисты по маркетингу разрабатывают специализированные действия, сформированные на основе анализа потребительских сегментов и моделей, прогнозирующих реакцию клиентов на конкретные предложения, и результатов предыдущих акций. В действиях указывается, какие предложения делать конкретным клиентам по различным каналам (через почтовую, электронную рассылку, социальные сети) [5].

Цикл обратной связи. После выполнения плана цикл повторяется. Оперативные системы собирают информацию о реакции клиентов на предложения или план, а также данные о последующих транзакциях (например, о продажах, акциях и т. д.). Эти данные извлекаются из хранилища, интегрируются с другой соответствующей информацией и анализируются бизнес-пользователями, которые оценивают эффективность своих планов и дорабатывают их нужным образом. Затем цикл снова повторяется [5].

Обучающиеся» компании. Пятиэтапный цикл обучения, состоящий из «сбора», «анализа», «планирования», «действия» и «пересмотра», создает так называемую «Обучающуюся» компанию», которая способна гибко и оперативно реагировать на различные изменения рынка (рисунок 7.2).

Среда хранилища данных. На представленной ниже диаграмме показана базовая BI-среда в виде двух пересекающихся овалов (рисунок 6.3). Левый овал называется «средой Хранилища». Это тот участок работы, на который ИТ-специалисты тратят от 60 до 80 процентов времени. Их задача состоит

в извлечении, очистке, моделировании, преобразовании, передачи и загрузке транзакционных данных из одной или нескольких оперативных систем в Хранилище.

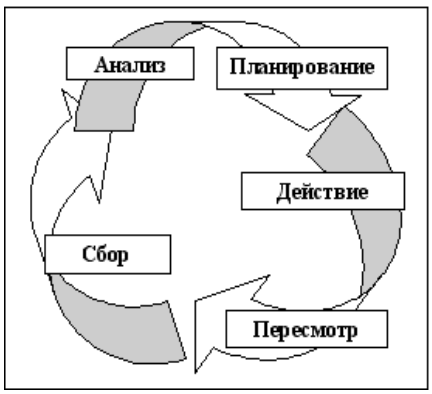

Рисунок 7.2 − Цикл обучения BI

Рисунок 7.3 − Базовая среда Business Intelligence

После очистки и анализа данных ИТ-персонал загружает их в хранилище, которое обычно представляет собой набор реляционных БД.

Аналитическая среда. Правый овал на рисунке 6.3. описывает аналитическую среду бизнес-пользователей, которые применяют инструменты для выполнения запросов, отчетов, а также анализа, исследования, визуализации. Отчеты можно просматривать в виде статических документов, устанавливать фильтры по соответствующим критериям или перемещать в разных направлениях (переход от одного измерения к другому и т. д) с тем, чтобы изменить представление или уровень детализации [5].

*Переход к реальному времени***.** Компании стараются добиться эффективного принятия оперативных решений, анализируя интегрированные данные в реальном времени. Например, менеджерам магазина нужно рассмотреть доходы за вчерашний день, а не за прошлый месяц, сравнить эффективность работы в тот же день в прошлом году, в других магазинах и т. п. [5].

Для поддержки такого оперативного принятия решений BI-системы приближаются по своим характеристикам к транзакционным. Команды разработчиков используют «активные хранилища», «оперативные склады данных» и промежуточное ПО (применяя интеграцию прикладных систем предприятия − EAI (Enterprise Application Integration) и веб-сервисы) для сбора данных.

7.2 Многообразие аналитических продуктов

Концептуально и архитектурно BI гораздо шире, чем простое формирование запросов и отчетов и другие аналитические средства. Системы BI образуют среду обучения, которая позволяет организациям более оптимально вести свой бизнес [5; 6].

Эволюция аналитики. Одни ИТ-специалисты предлагают программные комплекты, разработанные для решения любых аналитических задач в рамках организации. Другие встраивают эти инструменты в пакеты приложений, т. е. предлагают подготовленные решения, направленные на удовлетворение аналитических потребностей в конкретной отрасли, например, для оценки эффективности снабжения или поставок. Третьи сосредоточили свои усилия на создании вертикально-интегрированных пакетов, которые комбинируют ПО для интеграции данных с аналитическими инструментами и приложениями [5]. Выше показан «аналитический ландшафт» менеджеров в их компании (рисунок 7.4). Он отражает четыре основные категории аналитических инструментов, представленных в пересекающихся овалах, и три уровня управления: стратегический, тактический и операционный. Для принятия стратегических и тактических решений первые два используются три основных метода: *отчет, анализ и прогноз* [1; 2].

Стратегические решения предполагают анализ данных с целью долгосрочного планирования (на следующий квартал или следующий год) или управление деятельностью компании по реализации ее стратегических целей. *Тактические решения* нацелены на те действия, которые должны быть выполнены в ближайшее время (на следующей неделе или в следующем месяце). *Операционные решения* должны выполняться немедленно [5].

Рисунок 7.4 −Многообразие аналитических средств

Пользователи формировали запросы к OLTP-системе (On-Line Transaction Processing) и объединяли результаты. Появление «активных хранилищ данных» и аналитических инструментов, работающих в режиме реального времени (таких как панели инструментов, предупреждения, агенты и т. п.), позволяет анализировать в BI-системе данные, приближенные к реальным. За счет этого пользователи получают более полное и точное представление (сравнение по годам или сезонам) для принятия важных оперативных решений [5].

Отчетность и реагирование. Большинство пользователей (75 %) применяют инструменты для создания отчетов или выполнения оперативного контроля. В этом случае просто просматриваются «отчеты», которые могут быть статическими, параметризуемыми (т. е. отражающими определенный набор выделенных переменных) или интерактивными (допускающими поиск, детализацию, навигацию по определенной отчетной форме).

Анализ и прогнозирование. Анализ − компетенция бизнес-аналитиков, которые тратят довольно много времени, обрабатывая данные, создавая прогнозы, выясняя коренные причины различных проблем и тенденций в отрасли. Одни компании используют средства data mining для создания прогнозирующих и других моделей, которые управляют критически важными приложениями, например: для выявления случаев мошенничества с кредитными картами; для прогнозирования сбоев в работе частей конвейера; для предварительного поиска клиентов, которые могут откликнуться на конкретное предложение [5].

Глубина. Организации добиваются более серьезных результатов от внедрения аналитической среды тогда, когда пользователи переходят от простой отчетности («Что произошло?») к анализу («Почему это произошло?»), затем к прогнозирующему анализу («Что произойдет завтра?») и к оперативному контролю («Что произошло только что?») [5].

Ширина. Важно предоставить каждому пользователю соответствующие его деятельности инструменты. Неопытный пользователь, которому необходимо только раз в неделю просматривать стандартные отчеты, будет сбит с толку, если ему предложить сложный OLAP-инструмент или средство data mining. Все аналитические продукты были рассчитаны на продвинутых пользователей [5].

Ценность Business Intelligence. В начале главы были перечислены примеры организаций, которые извлекли существенную выгоду из использования BI. Несмотря на наличие таких примеров, многие руководители до сих пор сомневаются, стоит ли вкладывать средства в эту технологию. Потерпев неудачу в прошлом, они неохотно вкладывают с трудом заработанные средства в очередное рискованное предприятия.

Оправдание вложений в BI-проекты. Организации, внедрившие BI-проекты, приводят много явных и неявных преимуществ таких решений (рисунок 7.5). Хотя трудно задать точную величину возврата инвестиций (ROI) или денежный эквивалент, получаемый в результате этих преимуществ, большинство передовых руководителей придают большое значение тому факту, что у них теперь есть «единственно верная картина текущей ситуации», более каче-

ственная информация для стратегического и тактического принятия решений, а также более эффективные бизнес-процессы [5].

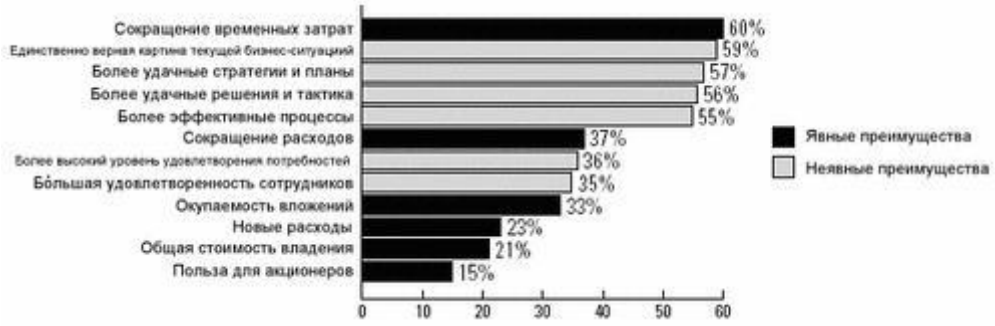

Рисунок 6.5 − Ценность явных и неявных преимуществ

Работа в несколько этапов. Многие специалисты в области BI рекомендуют поэтапную работу технических специалистов, занятых развертыванием BI-проекта. Например, BI-проект для руководителя департамента продаж на первом этапе будет представлять собой настольную систему, с помощью которой можно отслеживать доходы и вознаграждение по продавцам и продуктам. На следующем шаге эта информация станет доступна простым распространителям (продавцам) через веб-интерфейс в корпоративной сети. На третьем этапе добавляются данные подразделений и клиентов [5].

7.3 BI-проект: критерии успешной разработки

Полезный для бизнеса продукт, вложения в который быстро окупаются, можно определить по критериям, которые сформулированы так [2; 5; 6]:

1. Поддерживает ли BI-средство основной процесс, управляющий бизнесом?

2. Считают ли пользователи BI-проект важным для решения основных задач? (Например, «возмущаются» ли они, если система вдруг дает сбой?)

3. До какой степени BI-проект удовлетворяет требования пользователей?

Бизнес-инвестор. Каждому BI-проекту необходим преданный и активный бизнес-инвестор, который может продвигать проект, поддерживать и сохранять финансирование, решать политические вопросы, определять приоритеты проектов [6].

Бизнес-управляющий − бизнес-менеджер (или ИТ-менеджер со знаниями бизнеса). Любой бизнес-руководитель должен быть хорошо знаком с бизнесом и иметь глубокие знания в области ИТ [6].

1. Бизнес-инвесторы должны назначать бизнес-управляющих, которые будут от их имени руководить проектом.

2. Бизнес-управляющие должны посвящать по меньшей мере половину своего времени BI-проекту.

3. ИТ-отдел не должен заказывать BI-проект.

Взаимодействие. Чтобы гарантировать успех BI-проекта, необходимо добиться полного взаимодействия между менеджерами и ИТ-специалистами группы внедрения. Кроме того [6]:

1. Компании необходимо гарантировать слаженную работу ИТ-специалистов (внедряющей BI-проект) и представителей бизнеса.

2. Группы внедрения BI-проекта должны сформировать управляющий комитет для руководства, финансирования и назначения приоритетов для BI-проектов.

3. BI-команды должны сформировать рабочий комитет для управления проектом на операционном уровне.

BI-бюджеты, культура [6]:

1. Новые BI-проекты, которые оправдывают себя поэтапно, имеют больше шансов на успех, чем те, что начинаются с крупных бюджетов.

2. BI-проекты нуждаются в постоянном вливании денег и ресурсов для роста и адаптации к бизнесу.

3. Организации, применяющие ИТ, имеют высокие шансы на успех.

4. Организации, чьи сотрудники используют данные для принятия решений вероятнее всего достигнут поставленных целей.

Аналитика. Аналитические инструменты − путь к BI-решению для бизнес-пользователей. Реакция пользователей на эти инструменты и данные, которые они предоставляют, определяет успех BI-проекта [6]:

1. Успешные BI-команды разрабатывают стратегии, поощряющие применение внедренного BI-решения.

2. Удачные BI-проекты обеспечивают пользователей доступом к данным на более высоком уровне, а также множеством типов анализа.

3. Успех решения не зависит от того, был ли продукт куплен или разработан самостоятельно.

Задачи, которые необходимо решить для успешного развития BI-проекта, собранные на основе мнения BI-профессионалов и экспертов [5; 6].

1. Проект не будет иметь успеха, если руководство им не занимается.

2. Управляющий должен разработать маркетинговый план, используя КИС, годовые отчеты, цитаты в средствах массовой информации.

3. Расстановка приоритетов. Необходимо установить приоритеты приложений, согласно их стратегической ценности, и затем выпускать их поэтапно в течение разумного периода времени.

4. Выделение достаточного количества ресурсов. Бизнес-инвесторы обязаны сохранять исходное финансирование для запуска проекта. Еще важнее то, что они должны поддерживать субсидирование в течение всего цикла жизни BI-портфеля.

5.Укрепление доверия к системе. К сожалению, много возможностей дискредитировать BI-систему [6]:

− данные не точные, не согласовываются с операционными;

− данные выглядят иначе, даже если точны;

− нельзя найти источники показателей или данных в системе;

− интерфейс запутан и аналитический инструмент сложен;

− пользователям трудно находить нужный отчет;

− нет доступа к данным или отчетам из корпоративного Интернета;

− сложно создавать пользовательские формы данных;

− пользователь не обучен работе с аналитическими инструментами;

− пользователи не могут использовать BI-данные в приложениях;

− при работе с BI-системой нет простой и доступной помощи;

− замечания пользователей по усовершенствованию BI-проекта не принимаются во внимание;

− проект не соответствует инфраструктуре организации;

− BI-продукт требует много времени на поддержку;

− система работаем медленно, недоступна ночью и в выходные;

− по сравнению с другими ресурсами новый проект мало чего дает.

Техническая ИТ-команда должна [6]:

− разработать план, чтобы убедиться в адекватном качестве данных;

− создать словарь данных и показателей для бизнес-пользователей;

− в итеративном режиме создавать прототипы пользовательского интерфейса и учитывать все замечания;

− сделать BI-проект или соответствующие отчеты доступными в корпоративном Интернете;

− добиться такой производительности системы, чтобы время отклика было удовлетворительным;

− разработать такую архитектуру, чтобы по мере расширения проекта масштабирование выполнялось незаметно и было недорогим;

− используя реальные бизнес-сценарии и данные, создать обучающую программу, которая обеспечила бы поддержку пользователя;

− обучить и поддерживать квалифицированных пользователей, которые могли бы создавать отчеты для и отвечать на вопросы по данным;

− создать «справочный стол» для ответа на технические вопросы;

− разработать такую архитектуру BI-проекта, чтобы его можно было легко обновлять и изменять в соответствии с требованиями пользователей;

− иметь процедуры резервирования и восстановления после сбоев;

− обеспечить надежное и производительное решение.

Руководителям надо сосредоточиться на задачах, которые минимизируют риск и повышают вероятность успеха: выбор концепции, обеспечение признания концепции, расстановка приоритетов, выделение достаточных ресурсов, обеспечение взаимодействия бизнеса и ИТ, укрепление доверия к системе.

7.4 Фазы разработки BI-проектов

Многие компании готовы к внедрению различных BI-приложений, например карт сбалансированных показателей, управляющих панелей, модулей контроля ключевых показателей эффективности, информационных Хранилищ данных, инструментов формирования запросов и отчетов. Иногда отсутствует четкое представление о различиях между традиционными системами и BI-приложениями. В связи с этим необходимы действия [1; 2; 5; 7]:

− четко уяснить, чем BI-приложения отличаются от традиционных;

− разработать архитектуру BI-приложения как для общеорганизационной, многомерной системы отчетности;

− следовать всем фазам разработки, описанным в этой статье, либо применять уже опробованную в организации методику.

Фазы разработки. Почти все проекты разработки BI-приложения от начала и до полной реализации проходят через шесть фаз [7]:

1. Обоснование − создание BI-концепции и стратегии, анализ эффективности вложений.

2. Планирование, дизайн и построение BI-инфраструктуры.

3. Дизайн, разработка и управление долговременными и операционными информационными Хранилищами данных.

4. Извлечение информации и подготовка отчетности с использованием готового корпоративного ПО.

5. Визуализация данных, прогнозирование и их представление с использованием специальных приложений.

6. Управление BI-приложениями и инфраструктурой и их расширение.

Фаза 1: Обоснование создания BI-концепции и стратегии, анализ эффективности вложений. На данном этапе основная задача − оценка бизнес-ситуации. Так как BI-проекты стоят дорого, организации, стремящиеся к их созданию, должны не только разработать и стратегию, но и четко определить уровень окупаемости. На первом шаге определяются стратегические бизнес-цели. Проводится опрос бизнес-пользователей из различных подразделений на предмет требований к системе; при этом ИТ-персонал занимается исключительно координацией и технической поддержкой. Оценка бизнес-ситуации обычно состоит из четырех процессов [7].

1. *Анализ бизнеса и рынка.* Важно знать, каково текущее состояние вашего бизнеса и в какую сторону он развивается.

2. *Выбор бизнес-управляющих и инвесторов*. Без грамотных бизнес-управляющих и инвесторов, а также без утвержденного стратегического направления развития BI-проект скорее всего провалится.

3. *Оценка и уменьшение рисков*. Общие риски BI-проекта (насколько прибыльны выбранные технологии на рынке? Насколько они устоялись в компании? Сколько будут сосуществовать различные технологии? Есть ли в наличии несовместимые операционные системы? Есть ли несовместимые СУБД?) [7].

Риски сложности. (Насколько сложна ИТ-среда в целом? Насколько сложно BI-приложение само по себе? Насколько изменится технологический процесс? Сколько рабочих мест будет поддерживаться? Насколько они удалены друг от друга? Каков уровень распределенности данных, процессов и элементов управления?) [7].

Риски интеграции. (Сколько будет интерфейсов у приложения? Внешние ли это интерфейсы? Насколько велика избыточность источников данных? Насколько ошибочны данные? Есть ли несовместимые стандарты? Отсутствуют ли стандарты?) [7].

Организационные риски. (Какой риск допустим для руководства IT-отдела? На какую финансовую и моральную поддержку можно рассчитывать, если проект столкнется с трудностями?) [7].

Риски, связанные с командой разработчиков. (Каким опытом обладает ИТ-команда? Есть ли у сотрудников достаточная квалификация в основных областях? Каков моральный климат? Насколько квалифицирован менеджер проекта?)

Финансовые риски. (Как быстро проект должен окупиться? Какова вероятность того, что расходы превысят прибыль? Можно ли снизить риск за счет использования только проверенных технологий? Определив риски, перейдем к технологиям и планам) [7].

4. Анализ расходов/прибылей, сокращение расходов и повышение прибылей. Общий выигрыш от внедрения BI, как правило, труднее поддается количественной оценке, нежели расходы. Один из эффективных способов оценить BI-затраты − привязать их к конкретной, поддающейся измерению проблеме.

Следующий шаг − оценка и сравнение расходов, необходимых для достижения предполагаемой окупаемости. Выполнив все задачи первой фазы, компания должна получить следующее: BI-концепцию; сформулированные цели проекта; сравнительный анализ BI-целей и организационных целей; сравнение расходов и прибылей, оценку эффективности вложений; шаги, определяющие, как проект будет планироваться, разрабатываться и управляться [7].

Фаза 2: Планирование, дизайн и построение BI-инфраструктуры.

Инфраструктура состоит из технической и нетехнической частей. В первую очередь необходимо рассмотреть техническую инфраструктуру, сотрудникам ИТ-отдела необходимо: оценить существующие платформы; оценить и выбрать новые продукты, оборудование, ПО и сетевые компоненты; написать отчет о технической инфраструктуре; расширить существующую платформу.

Далее производится оценка нетехнической инфраструктуры, для чего необходимо: оценить политику, процедуры, руководства и стандарты, которые предназначены для того, чтобы поддерживать координацию и управление BI-среды. По окончании второй фазы должно быть выполнено: проектирование и реализация необходимой инфраструктуры; создание и расширение существующей сети; внедрение необходимых требований по безопасности; выгрузка корпоративных порталов, приложений для управления контентом и систем управления знаниями [7].

Фаза 3: Проектирование, построение и управление оперативными информационными складами данных. BI дает компаниям возможность принимать обоснованные решения за счет выполнения трендового анализа, прогнозирования, data mining и статистического исследования информационных складов данных. Для эффективного использования этой информации: определить требования к данным со стороны специалистов и лиц, принимающих решения; оценить операционные источники и процедуры; выполнить моделирование

данных и информации; создать репозиторий метаданных, который хранит не сами бизнес-данные для приложения, а только контекстуальную информацию о них; извлечь, очистить, преобразовать и загрузить данные в Хранилища и операционные склады данных.

Управлять расширением Хранилища [7]. По самым осторожным оценкам, объем данных в BI-среде удваивается каждые два года. По мере их роста возникает необходимость в создании плана агрегирования и объединения стареющих данных. Результаты выполнения третьей фазы следующие: логическая и физическая модель данных; целевая база данных BI; репозитории метаданных для бизнес-данных и данных приложения; очистка данных; выпущены ETL-инструменты; реализовано корпоративное Хранилище; интеграция с внешними источниками данных.

Фаза 4. Извлечение информации и создание отчетности с использованием готового ПО компании. Для этого необходимо: использовать встроенную функциональность существующих оперативных систем; развернуть готовое ПО для формирования отчетности и запросов. По завершении четвертого этапа должны быть получены следующие результаты: аналитические и статистические отчеты для руководителей и менеджеров; отчеты об очевидных недостатках и возможностях операционных процессов [7].

Фаза 5: Визуализация данных, прогнозирование и их представление с помощью специальных приложений. Для этого приходится использовать и настраивать различные типы ПО; разработка приложения; OLAP-инструменты и многомерные OLAP-инструменты; среда разработки; веб-среда; Data mining [7].

На этапе пятой фазы необходимо выполнить следующие основные операции: определить конечные требования к внедряемому проекту; спроектировать, разработать и протестировать специальные BI-приложения; подготовить BI-приложение для функционирования; обеспечить обучение пользователей. Выполнив эти действия, можно получить следующее: системы динамического создания отчетов и запросов; инструментальные панели BI для руководителей, где можно выполнять нерегламентируемые запросы и многомерное представление ситуации в компании в реальном времени; внедрение мобильных приложений для оптимизации процессов и эффективного сбора данных.

Фаза 6: Управление и расширение BI-приложений и инфраструктуры. Управление развитием системы. Количество данных в BI-среде удваивается каждые два года.

Три ключевые области развития, за которыми необходимо следить, – это рост объема данных, рост интенсивности использования системы и рост количества оборудования. Если выполнить все операции фазы 6, то можно получить следующие результаты: функциональную справочную службу; полностью настроенные средства обучения; функциональные депозитарии метаданных и программные библиотеки; автоматизированные ETL-процессы с автоматизированным механизмом оповещения; процедуры системного копирования и восстановления из архива; плановую оценку уровней и показателей производительности; задокументированные процессы для отчетов и принятия решений [7].

7.5 BI и сбалансированная система показателей

Всю концепцию управления эффективностью компании (Business Performance Management − BPM) можно свести к одному предложению: вы можете управлять лишь тем, что вы можете измерить. Одной из наиболее известных управленческих методик является сбалансированная система показателей (ССП). С помощью ССП руководство может оценивать эффективность работы компании как по финансовым, так и по другим показателям, в том числе степени удовлетворенности клиента, эффективности внутренних процессов и числу инноваций (рисунок 7.6) [4].

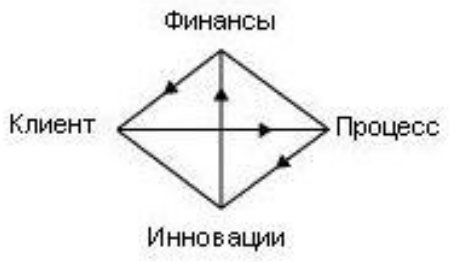

Рисунок 7.6 − Взаимосвязь между показателями эффективности работы компании

С помощью ССП можно определить показатели, необходимые для осуществления контроля за реализацией стратегии, и данные, необходимые для анализа деятельности. Интеграция аналитических ССП-приложений с Хранилищем данных дает преимущества: связь стратегии со всеми корпоративными показателями эффективности; полное понимание информационных потребностей пользователя в соответствии с используемыми им элементами ССП; повышение эффективности деятельности как отдельных работников, так и компании в целом,

Архитектура корпоративной ССП. Для эффективной поддержки ССП-приложений корпоративная информационная архитектура должна отвечать следующим требованиям: включать в себя хранилище данных, которое загружает, согласовывает и обобщает данные из соответствующих источников; реализовывать открытый модульный принцип построения системы (с использованием открытых стандартов − XML), что позволяет при необходимости расширить хранилище данных и ССП-приложения.

Связывая ССП-приложения с хранилищами данных, повышается эффективность тех и других. Данные для ССП поступают напрямую из хранилища данных, что обеспечивает их актуальность и фактическую точность. Хранилище данных более полно и точно отражает показатели ССП, что облегчает выполнение стратегических задач и повышает эффективность на всем предприятии.

Выводы

На основе материала главы с учетом источников [1–7] резюмируем.

1. Business Intelligence (BI)-продукты, построенные на использовании технологий ИИ в бизнесе, принципиально отличаются от других составляющих

корпоративных систем. Хорошо спроектированные BI-системы адаптивны по своей природе, со временем они изменяются и дают ответы на все новые и новые бизнес-вопросы, лучшие из них формируются годами, разрастаясь со временем в ширину и глубину.

2. Для успешных BI-решениях характерны следующие признаки: финансирующая сторона глубоко заинтересована в проекте и активно участвует в его реализации; бизнес-пользователи и разработчики BI-проекта тесно взаимодействуют; BI-система рассматривается как корпоративный ресурс, на который выделяется адекватное финансирование; компания обеспечивает пользователей как статическими, так и интерактивными формами данных; команда разработчиков имеет достаточный опыт в области BI, ее поддерживает поставщик и независимые консультанты, организационная культура компании укрепляет BI-проект.

3. В BI-среду поступает первичный материал − данные, которые затем перерабатываются во множество информационных продуктов по схеме: от данных к информации, затем к знаниям, от них к правилам, затем к действиям, потом цикл повторяется. Пятиэтапный цикл обучения состоит из «сбора», «анализа», «планирования», «действия» и «пересмотра» и создает так называемую «обучающуюся» компанию, которая способна гибко и легко реагировать на любые изменения на рынке.

4. Для принятия стратегических и тактических решений используются три основных метода: отчет, анализ и прогноз. Стратегические решения предполагают анализ данных с целью долгосрочного планирования или управление деятельностью компании по реализации ее стратегических целей. Тактические решения нацелены на те действия, которые должны быть выполнены в ближайшее время, они больше управляются процессом, чем стратегические.

5. Компании используют средства data mining для создания прогнозирующих и других моделей, которые управляют критически важными приложениями, например: для выявления случаев мошенничества с кредитными картами, для прогнозирования сбоев в работе частей конвейера, для предварительного поиска клиентов, которые могут откликнуться на конкретное предложение.

6. Выводы при инвестировании BI-проектов: бизнес-инвесторами должны быть руководители, обладающие большим влиянием в организации, имеющие четкое представление, как можно использовать BI-технологию для реализации основных бизнес-стратегий; заказчики должны быть полностью преданы BI-проекту, иначе его шансы на успех будут неуклонно падать; бизнес инвесторы должны назначать бизнес-управляющих, которые будут от их имени руководить проектом; бизнес-управляющие должны посвящать по меньшей мере половину своего времени BI-проекту; IT-отдел не должен заказывать BI-проект [6].

7. Организации добиваются успеха, если: она активно применяет информационные технологии и при этом уверена, что информация обеспечивает конкурентные преимущества; в ней сотрудники используют данные для принятия решений (а не свою интуицию) и могут свободно делиться информацией [6].

8. Аналитические инструменты − путь к BI-решению для бизнес-пользователей, они определяют успех BI-проекта, если: организации проводят анализ типов пользователей и обеспечивают те инструменты или приложения, которые лучшим образом обеспечивают их аналитические нужды: BI-команды разрабатывают стратегии, поощряющие применение внедренного BI-решения; удачные BI-проекты обеспечивают пользователей доступом к данным на более высоком уровне, а также множеством типов анализа; успех решения не зависит от того, был ли продукт куплен или разработан самостоятельно [6].

9. Важнейшие задачи, которые необходимо решить для успешного развития BI-проекта, собранные на основе мнения BI-профессионалов и экспертов: выбор концепции; ее распространение; расстановка приоритетов; выделение достаточного количества ресурсов; обеспечение взаимодействия бизнеса и IT; укрепление доверия к системе [6].

10. Есть много возможностей дискредитировать BI-систему: данные неточные, не согласовываются с операционными; данные выглядят иначе, даже если точны; нельзя найти источники показателей или данных в системе; пользовательский интерфейс запутан и аналитический инструмент сложен в использовании; пользователям трудно находить нужный отчет; нет доступа к данным или отчетам из корпоративного Интернета; сложно создавать пользовательские формы данных; пользователь не обучен работе с аналитическими инструментами; пользователи не могут использовать BI-данные в других приложениях; при работе с BI-системой нет простой и доступной помощи; замечания пользователей по усовершенствованию BI-проекта не принимаются во внимание; проект не соответствует существующей инфраструктуре организации; BI-продукт требует много времени на поддержку и сложен в администрировании; система работаем медленно, ненадежно, недоступна ночью и в выходные [6].

11. Все проекты разработки BI-приложения от начала и до полной реализации проходят через шесть фаз: обоснование – создание BI-концепции и стратегии, анализ эффективности вложений; планирование, дизайн и построение BI-инфраструктуры; дизайн, разработка и управление историческими и операционными информационными Хранилищами данных; извлечение информации и подготовка отчетности с использованием готового корпоративного ПО; визуализация данных, прогнозирование и их представление с использованием специальных приложений; управление BI-приложениями и инфраструктурой и их расширение [7].

12. Риски BI-проекта: насколько прибыльны выбранные технологии на рынке? насколько они устоялись в компании? сколько будут сосуществовать различных технологий? есть ли в наличии несовместимые операционные системы? есть ли несовместимые СУБД? Риски сложности: насколько сложна IT-среда в целом? насколько сложно BI-приложение само по себе? насколько изменится технологический процесс? сколько рабочих мест будет поддерживаться? насколько они удалены друг от друга? каков уровень распределенности данных, процессов и элементов управления? [7].

13. Интеграция аналитических ССП-приложений с хранилищем данных дает следующие преимущества: связь стратегии со всеми корпоративными показателями эффективности; полное осознание информационных потребностей пользователя в соответствии с используемыми им элементами ССП; повышение эффективности деятельности как отдельных работников, так и компании в целом посредством соотнесения количественных показателей эффективности с конкретными задачами.

Вопросы для контроля

1. Определите назначение интеллектуализации бизнеса.
2. Опишите BI-среду.
3. Определите условия для успешных BI-решений.
4. Из чего состоит пятиэтапный цикл обучения?
5. Охарактеризуйте стратегические и тактические решения.
6. Перечислите условия выполнения BI-проекта.
7. Укажите явные и неявные преимущества BI-программ.
8. Перечислите риски BI-проекта.
9. Охарактеризуйте 1-ю фазу BI-проекта.
10. Перечислите задачи 2-й и 3-й фаз.
11. Что решается во время выполнения фаз 4–6 BI-проекта?
12. В чем суть сбалансированной системы показателей?
13. Что решается во время выполнения фаз 1–5 BI-проекта?

Литература, используемая в главе

1. Business intelligence (BI) [Электронный ресурс]. – Режим доступа : https://skillbox.ru/media/management/business-intelligence-bi-chto-eto-takoe-zachem-biznesu-bisistemy-i-kak-oni-rabotayut/. – Дата доступа : 15.11.2024.
2. Российский сервер «Информационные технологии в управлении» [Электронный ресурс]. – Режим доступа: http://www.cfin.ru/itm. – Дата доступа: 15.12.2021.
3. Вишняков, В. А. Интеллектуальные системы в управлении / В. А. Вишняков. – Минск : Изд-во МИУ, 2010. – 364 с.
4. Информатика: учебник для эконом. специальностей вузов / под ред. Н. В. Макаровой. – М. : Финансы и статистика, 1999. – 440 с.
5. Градобоев, В. В. Использование средств бизнес-аналитики в управлении предприятием [Электронный ресурс] / В. В. Градобоев, В. Г. Карчик. – Режим доступа: https://cyberleninka.ru/article/n/konsolidatsiya-rashodov-zheleznoy-dorogi. – Дата доступа: 15.12.2021.
6. BI проект: критерии успешной разработки / Журнал BPM World. [Электронный ресурс]. – Режим доступа : https://iso.ru/ru/press-center/journal/1979.phtml. Дата доступа : 15.12.2022
7. Метаданные [Электронный ресурс]. – Режим доступа: http://www.citforum.ru/consulting/BI/never_casual/. – Дата доступа: 15.12.2021.

8 ИНТЕЛЛЕКТУАЛЬНЫЕ ТЕХНОЛОГИИ В ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ И СЕТЯХ

8.1 Основы машинного перевода

Разработка естественно-языковых интерфейсов и машинного перевода (МП) [1]. В 50-х годах одной из популярных тем исследований в области искусственного интеллекта являлась тема машинного перевода. Мысль использовать ЭВМ для перевода была высказана в 1947 году в США сразу после появления первых ЭВМ. Первая публичная демонстрация машинного перевода (МП). Несмотря на примитивность той системы (словарь в 250 слов, грамматика из 6 правил, перевод нескольких простых фраз), этот эксперимент получил широкий резонанс: начались исследования в разных странах; в том же 1954 году и в СССР. К середине 1960-х годов в США для практического использования были предоставлены две системы русско-английского перевода: MARK (в Департаменте иностранной техники ВВС); GAT (разработка Джорджтаунского университета, использовалась в Национальной лаборатории атомной энергии в Окридже и в центре Евратома в г. Испра, Италия) [1].

Существенным фактором стало появление мини- и персональных машин, а с ними все более сложных словарных, поисковых и т. п. систем, ориентированных на работу с естественно-языковыми данными. Росла и необходимость в переводе из-за роста международных связей. Все это привело к подъему этой области, наступившему примерно с середины 1970-х годов. В 1980-х наступило время широкого практического использования переводческих систем, сложился рынок коммерческих разработок МП.

Распространенными способами использования ИС при письменном переводе является работа со словарями и глоссариями (*Translation Memory, TM*), содержащими примеры раньше переведенных текстов, терминологическими базами, использование больших коллекций текстов на одном или нескольких языках, что дает сжатое описание того, как слова и выражения реально; используются в языке в примерах или в конкретной предметной области [1; 2].

Статистический машинный перевод – это разновидность машинного перевода текста, основанная на сравнении больших объемов языковых групп − текстов, содержащих предложения на одном языке и соответствующие им предложения на другом языке. Эти пары могут быть как вариантами написания двух предложений человеком − носителем двух языков, так и набором предложений и их переводов, выполненных разными людьми.

Статистический МП обладает свойством «самообучения». Чем больше в распоряжении имеется языковых пар и чем точнее они соответствуют друг другу, тем лучше результат статистического машинного перевода. Основанный на правилах МП использует более сложную модель, включающую анализ и синтез естественно-языковых сообщений, которая состоит из нескольких блоков.

Анализ текста включает пять этапов [10]:

− лексический анализ – анализ символов в словах и предложениях;

– морфологический анализ − анализ слов в тексте;

− синтаксический анализ − анализ предложений, грамматики и связей между словами;

− семантический анализ − анализ смысла каждого предложения на основе некоторой предметно-ориентированной базы знаний;

− прагматический анализ − анализ смысла предложений в окружающем контексте на основе собственной базы знаний.

Существуют два разных подхода к построению алгоритмов машинного перевода: основанный на правилах (rule-based); статистический, или основанный на статистике (statistical-based) [1]. Первый подход является традиционным и используется большинством разработчиков систем машинного перевода (ПРОМТ в России, SYSTRAN во Франции, Linguatec в Германии и др.) Ко второму типу относится популярный сервис Яндекс-переводчик, переводчик Google, а также новый сервис от ABBYY [1].

Синтез текста включает аналогичные этапы, но в обратном порядке. В таких системах реализован путь от синтаксического представления до цепочки слов выходного предложения.

– лексический синтез символов в словах и предложениях;

– морфологический синтез – слов в тексте;

– синтаксический синтез – предложений, грамматики и связей между словами;

– семантический синтез – смысла отдельного предложения на основе некоторой предметно-ориентированной базы знаний;

– прагматический синтез – смысла предложений в окружающем контексте на основе собственной базы знаний.

Формы организации взаимодействия компьютера и переводчика при МП [1; 2]:

1. С постредактированием: исходный текст перерабатывается машиной, а переводчик исправляет результат.

2. С предредактированием: переводчик приспосабливает текст к обработке машиной (устраняет возможные неоднозначные прочтения, упрощает и размечает текст), после чего начинается программная обработка.

3. С интерредактированием: переводчик вмешивается в работу системы перевода, разрешая трудные случаи.

4. Смешанные системы (например, одновременно с пред- и постредактированием переводчиком).

Рассмотрим элементы лингвистического обеспечения системы МП. Словари анализа, как правило, одноязычные. Разделяются словари основ и словосочетаний. Также разделяют словари общеупотребительной и терминологической лексики. Грамматика и словарь задают лингвистическую модель. Различают следующие виды грамматик синтаксического уровня (наиболее разработанные с точки зрения формализма и лингвистики):

− цепочечная (фиксирует порядок следования элементов, т. е линейных структур предложения);

− составляющих (фиксирует лингвистическую информацию о группировке грамматических элементов – именная, предложная, дополнений и т. д.);

− зависимостей (задает иерархию элементов предложения, при этом главным считается глагол в личной форме, который определяет число и характер зависимых предложений);

− категориальная (верхняя грамматика составляющих, в ней 2 категории – предложения и имени, остальные члены определяются в терминах способности комбинироваться с этими главными в структуре текста), она дополняется набором правил работы с составляющими элементами;

− обобщенных составляющих (в ней введены метаправила, являющиеся обобщением закономерностей правил категорийной грамматики.

Для любой системы анализа языка необходимо выбрать варианты разбора. Для ее решения грамматики синтаксического уровня дополняются вспомогательными грамматиками и методами разбора сложных ситуаций. Применяют фильтровый и эвристический методы. Первый основан на том, что получают все варианты анализа предложения, а потом часть отфильтровывается, те, которые не удовлетворяют системе фильтров. Эвристический строит лишь часть вариантов, правдоподобных с точки зрения заданных критериев, например неких весов.

Семантический уровень анализа в меньшей степени обеспечен теорией и практическими разработками. Традиционным считается снятие неоднозначностей после лексического и синтаксического анализов. Используется аппарат селективных ограничений, который вписывается в синтаксическую модель

К математико-алгоритмическому обеспечению СМП относятся: формализмы, используемые для задания лингвистических данных (словарей и грамматик); специальные алгоритмические языки, ориентированные на МП; языки программирования. Формальными являются также языки описания словарей. Программное обеспечение СМП дополняется периферийными модулями: ввода данных, коррекции ошибок, управление данными, редактирование текста, вывод текста, передачу данных, пополнения и модификации словарей и грамматик. МП используется в следующих сферах:

− отраслевых службах информации при поиске иноязычных источников;

− в международных организациях, имеющих дело с многоязычным потоком документов;

− в службах, осуществляющих перевод технической документации.

8.2 Особенности машинного перевода

*Автоматический перевод устной речи***.** МП устной речи с одного языка на другой реализуется с помощью специальных программных и технических средств [1]. Также называется направление исследований, связанных с построением подобных систем. В отличие от печатного текста или искусственных сигналов, естественная речь не допускает простого и однозначного членения на элементы (лексемы, слова, фразы), поскольку они не имеют явных физических

границ. Границы слов в потоке речи автоматически могут быть определены лишь в ходе распознавания посредством подбора оптимальной последовательности слов, наилучшим образом согласующейся с входным потоком речи по акустическим, лингвистическим, семантическим и иным критериям [1; 4].

Октябрь 2012 года − автоматический, почти синхронный голосовой перевод с английского на путунхуа. Разработчик − фирма Microsoft. Система обучения на основе искусственных нейронных сетей (Deep Neural Networks), которая сокращает непонимание до каждого седьмого, восьмого слова. Но самое большое достижение − это генерация речи с сохранением модуляций голоса говорящего. Ноябрь 2012 года − открывшийся сервис японского мобильного оператора NTT Docomo позволяет абонентам, говорящим на разных языках, общаться в режиме реального времени. Языки, поддерживаемые сервисом: (японский ↔ английский), (японский ↔ корейский), (японский ↔ китайский) [4].

Процесс электронного перевода речи (S2S Real-Time Translation), как правило, включает следующие три этапа): автоматическое распознавание речи (ASR − automatic speech recognition) − преобразование речи в текст; МП (MAT − Machine-Assisted Translation); − автоматический перевод текста с одного языка на другой; синтез речи (TTS − text-to-speech) − технология, которая дает возможность произнести текст голосом, приближенным к естественному.

Говорящий на языке А использует микрофон, а модуль распознавание речи вводит произнесенное. Происходит сравнение входных данных с фонологическими моделями, состоящими из большого количества речевых библиотек. Отфильтрованное таким образом, используя словарь и грамматику языка А, преобразуется в строку слов, основанную на массиве фразы языка А. Модуль автоматического перевода преобразует эту строку. Ранние системы заменяли каждое слово соответствующим словом в языке В. Более совершенные системы не используют дословный перевод, а принимают во внимание весь контекст фразы, чтобы произвести соответствующий перевод [4].

Созданный перевод передается в модуль синтеза речи, который оценивает произношение и интонацию, соответствующую ряду слов из массива речевых данных языка В. Данные, соответствующие фразе, отбираются, соединяются и выводятся в необходимой потребителю форме на языке В. Системы перевода речи (ST – Speech Translation) состоят из двух основных компонентов: автоматическое распознавание речи (ASR – automatic speech recognition) и МП (MAT – Machine-Assisted Translation) и различаются: работающие «на клиенте» (client-based); по принципу «клиент-сервер» (client-server) (OnLine service) [1; 4].

Распознавание слитной спонтанной речи − конечная цель всех усилий по распознаванию речи. Автоматическое распознавание речи разделяют на привязку и ее отсутствие, к голосу конкретного человека.

Виды систем машинного перевода. Системы машинного перевода делятся на категории: Rule-Based Machine Translation RBMT) − системы, основанные на правилах, которые описывают языковые структуры и их преобразования; Example-Based MT (EBMT) − системы на примерах двух текстов, один из которых является переводом другого; Statistical Machine Translation (SMT) − стати-

стический машинный перевод − разновидность МП текста, основанная на сравнении больших объемов языковых пар; Hybrid Machine Translation (SMT + RBMT) − гибридные модели «… где ожидается прорыв в качестве перевода» [4; 3]. Границы между системами Example-based и Rule-based не очень четкие, поскольку и те и другие используют словари и правила работы со словарями. Типичная архитектура статистических систем автоматической обработки речи:

Акустическая модель – позволяет оценить распознавание речевого сегмента с точки зрения схожести на звуковом уровне. Для каждого звука изначально строится сложная статистическая модель, которая описывает произнесение этого звука в речи.

Языковая модель. Чем больше в распоряжении имеется языковых групп (пар) и чем точнее они соответствуют друг другу, тем качественней результат статистического машинного перевода.

Статистический декодер – программный компонент системы распознавания, который совмещает данные, получаемые в ходе распознавания от акустических и языковых моделей, и на основании их объединения, определяет наиболее вероятную последовательность слов, которая и является конечным результатом распознавания.

Технология МП на основе правил – (Rule-Based Machine Translation - RBMT), подразделяется: системы пословного перевода; трансферные системы (Transfer) – преобразуют структуры входного языка в грамматические конструкции выходного языка [4];

– интерлингвистические системы (Interlingua) – промежуточный язык описания смысла. Компоненты типичной RBMT:

– лингвистические базы данных: (двуязычные словари; файлы имен, транслитерации; морфологические таблицы);

– модуль перевода: (грамматические правила; алгоритмы перевода).

Особенности RBMT-систем следующие.

Преимущества: синтаксическая и морфологическая точность; стабильность и предсказуемость результата; возможность настройки на предметную область. Недостатки: трудоемкость и длительность разработки, необходимость поддерживать и актуализировать лингвистические БД; «*машинный акцент» при переводе.*

*Гибридные модели SMT + RBMT***.** Разработчики систем машинного перевода для улучшения качества вводят некоторые «сквозные» правила, тем самым превращая чисто статистические системы в гибридный машинный перевод. Добавление некоторых правил, то есть создание гибридных систем, несколько улучшает качество переводов, особенно при недостаточном объеме входных данных, используемых при построении индекса машинного переводчика [3].

Гибридная технология «SMT и RBMT. Объединение RBMT и статистических технологий: лингвистический анализ входного предложения; порождение вариантов перевода; использование статистических технологий; оценка и выбор лучшего варианта перевода с использованием модели языка. Этапы ги-

бридной технологии: обучение RBMT на основе параллельного корпуса с использованием статистических технологий; эксплуатация на основе натренированной системы.

*Системы синтеза речи***.** Типичная архитектура «Text-to-Speech» System. Анализ текста: Определение структуры текста; Нормализация текста; Лингвистический анализ. Фонетический анализ: Графо-Фонетическое преобразование. Анализ просодики: Шаг & Длительность словосочетаний. Синтез речи (Speech Synthesis): Рендеринг голоса. Синтез речи разделяют на группы: параметрический синтез; конкатенативный, или компиляционный (компилятивный) синтез; синтез по правилам; предметно-ориентированный синтез [1; 4].

Помимо проблем, связанных с переводом текста, синхронный перевод речи имеет дело с проблемами, включая бессвязность разговорного языка, меньше ограничений грамматики разговорного языка, неясной границы слова разговорного языка и коррекции ошибок распознавания речи. У синхронного перевода есть свои преимущества по сравнению с переводом текста, менее сложная структура разговорного языка и меньше лексики в разговорном языке.

Ситуация хуже в случае принадлежности говорящих к разным языковым группам. Так, английский язык относится к германской группе индоевропейской семьи языков, а китайский – к китайско-тибетской языковой большой семье. Различия между ними очень велики, сделать правильный перевод затруднительно, причем же одно и то же слово может означать два и более разных по смыслу вариантов перевода в другом языке. По этим причинам процентное количество ошибок при переводе далеких друг от друга языков остается все еще высоким.

*Стандарты МП***.** Международное объединенное исследование создается речевыми консорциумами перевода: (C-STAR) Consortium for Speech Translation Advanced Research – международный консорциум по переводу речи для объединенного исследования речевого перевода; (A-STAR) Asia-Pacific – для азиатского региона. Они были основаны как международная объединенная исследовательская организация по проектированию форматов двуязычных стандартов, которые важны для продвижения научных исследований этой технологии и стандартизации интерфейсов и форматов данных, чтобы соединить речевой модуль перевода на международном уровне [4].

BLEU (Bilingual Evaluation Understudy) – алгоритм автоматической оценки качества машинного перевода по сравнению с человеческим на основе совпадения n-грамм. WER (Word Error Rate) – алгоритм оценки-оптимизации качества текста, МП на основе вероятности ошибки в кодовом слове. Классификатор «Речь/не речь» (speech/non-speech) – определяющий вероятность правильного распознавания речи. Компромисс между определением, голос как шум или шум как голос (Type I and type II errors) [4].

8.3 Построение алгоритмов решения задач

Рассмотрим известные результаты по синтезу алгоритмов для конструирования объектов. При этом важную роль играют языковые средства, необходимые для описания постановки решаемых задач. К таким средствам относятся языки спецификаций, под которыми понимают метасредства для описания не алгоритма решения, а постановки задачи. Среди отечественных систем такого назначения можно выделить ПРИЗ и СПОРА, языками в которых соответственно являются УТОПИСТ и ДЕКАРТ [6].

Несмотря на многообразие средств и приемов, положенных в основу пакетов синтеза программ, большинство из них используют два подхода: либо генерируют программу на машинном языке, либо составляют резидентный набор из заранее отлаженных модулей на время выполнения задачи. В большинстве систем автоматизации программирования описание исходной задачи составляется в диалоговом режиме, причем вначале выбирается этап описания, а затем форма заполняемого кадра. В отдельных системах при выборе этапа или кадра выдаются подсказки. В первом подходе пользователь должен более глубоко представлять технологию проектирования, но сам процесс синтеза выполняется более быстро. Во втором случае система проще для менее подготовленного пользователя.

Рассмотрим вопросы конструирования алгоритмов решения, взяв за основу идеологию системы ПРИЗ, но добавив в нее некоторые положения: систему логического вывода для получения новых фактов и новых отношений вычислимости; систему аналитического решения уравнений для получения новых отношений и тождеств; при создании языковых средств использовать понятия языка УТОПИСТ и нового УТОПИСТа – НУТ, упростив описательную часть и усилив препроцессорную обработку [3; 6].

Заметим, что в основе синтеза объектов могут быть использованы три подхода: дедуктивный, индуктивный и трансформаторный. Первый заключается в построении объекта на основе доказательства утверждения, что решение задачи существует. Во втором подходе синтез основывается на использовании частных примеров, задающих ответ для отдельных исходных данных. В третьем искомый объект (программа) получается постепенно путем преобразования исходного описания по правилам, совокупность которых представляет знания о решаемой задаче. Наиболее формально разработан первый подход, который и положим в основу модели синтеза искомого объекта. При этом процесс синтеза включает три этапа:

– описание условий задачи в виде отношения $R(x, y)$, связывающего исходные данные X с результатом решения Y;

– получение конструктивного доказательства теоремы существования решения задачи (объекта);

– перевод полученного доказательства в форму последовательности программы.

Основные трудности при практическом решении проблемы синтеза объектов (программ) лежат в двух направлениях: в описании условий задачи на удобном языке спецификаций, который применим для представления о данной предметной области; в построении и автоматизации процесса доказательства решения для данной постановки задачи. Рассмотрим формулировку теоремы существования решения задачи (объекта), введя следующие обозначения [6]:

$$(M: X_1, X_2, ...X_n \rightarrow Y_1, Y_2, ...Y_m),$$

где M – условие задачи; X_i – входные переменные (объекты); Y_j – выходные переменные (объекты). В данном случае каждой задаче (конечному объекту) можно сопоставить следующее утверждение. При заданных условиях задачи (синтезируемого объекта) для любых значений X_i, удовлетворяющих условиям задачи, найдутся такие значения Y_j, которые также удовлетворяют этим условиям. Для того, чтобы задача была разрешима (объект синтезируем), необходимо, чтобы это утверждение было истинным. В последнем случае это утверждение является теоремой существования решения задачи (объекта), для доказательства данной теоремы существенным является понятие отношения вычислимости, связывающее два объекта: из X_i вычислимо Y_j применением функции F, которое обозначается $X_i \rightarrow Y_j$. При этом под X, Y могут пониматься не только множества объектов, но и их компонентов, участвующих в решении задачи.

Состоянием на i-м шаге доказательства теоремы назовем множество объектов V, обладающих свойством V ∈ X для всех X, вычислимость которых доказана после завершения i-го шага решения. Основные правила вывода для отношений вычислимости, на основании которых строится доказательство теоремы существования решения задачи (объекта), записывается в виде

$$A_1, A_2, ...A_n \rightarrow B,$$

где A_1, A_2,...A_n – посылки (условия), а B – следствие. Далее рассмотрим представление элементов знаний для синтеза объектов, и языковые конструкции для описания такого синтеза, обобщенный алгоритм синтеза и, наконец, отдельные алгоритмы и модели его составляющих.

Будем рассматривать n-объекты при $n = 1$, называя их просто объектами. К тому же в этом рассмотрении объект $<A, f>$, где $A = \{a_1, a_2, ...a_n\}$ – множество элементов и $f(i) = a_i$ $(1 = I <= m)$ – функция их выбора (доступа), будет представляться в виде $<B>$, где $B = \{(a_i, f(i), g(i)) | I = 1,...,m\}$, а $g(i)$ элемент, следующий за a_i. При этом соседние элементы будут комбинироваться с правилами выбора. Представление знаний для синтеза объектов будем основывать на семантической модели. Положим, что семантическая память в этом случае имеет структуру дерева, в вершинах которого лежат типы (классы). При этом на входном языке системы синтеза объектов необходимо задавать условия задачи в виде совокупности описаний исходных объектов, указывая их типы. Тип – основная структурная единица в базе знаний, которая в общем случае задается следующим образом: $T = (S, O, U, P)$, где T – тип объекта; S – структура типа; O – множество отношений вычислимости, определенных на структуре; U – множество уравнений, связывающих элементы структуры; P – множество теорем для данного типа.

Структура S задается в виде множества кортежей вида S = (У, И, С), где У – уровень описания элемента структуры, представляет число от 1 до N; И – имя, характеризующее компоненту объекта; С – ссылка на описание компоненты объекта. Определение типа через другие типы означает иерархию типов, а уровень – ступенька в этой иерархии. Если ссылка имеет значение ВЕЩ (вещественный), СТР (строковый), то тип является первичным и компонента объекта с таким типом может входить в выражения и уравнения, участвующие в описании и формировании отношений вычислимости.

Отметим, что понятие первичности типа является относительным. Если в какой-либо задаче структурные свойства какого-либо компонента не являются существенными с точки зрения решения, то «этот тип может быть объявлен первичным, обозначив тип-ссылку символом?» Наличие непервичного типа-ссылки одной из компонент объекта говорит о том, что он является составным. Вычислительная модель такого объекта строится путем включения в его описание всех компонент, его составляющих, со своими вычислительными моделями. Заметим, что кортеж с единичным уровнем ссылки типа-ссылки не имеет. Сначала определим выражения следующим образом.

1. Каждый идентификатор (имя объекта) является выражением.

2. Если V и W являются выражениями, есть одна из операций E [+, −, *, /], то запись ($V\,E\,W$) также является выражением.

Отношение вычислимости имеет вид $A \rightarrow B$, где A, B – список идентификаторов. Отношения могут быть безусловными (ОТН) и условными (ОТНУ). Последнее имеет меньший приоритет. Если в качестве ответа на запрос генерируется программа или объект-коммуникатор, то условному отношению ставится в соответствие условный оператор. Теоремы (продукционные правила) образуют одно из основных отличий системы микроПРИЗ от ССО. Общий вид продукции $A_1, A_2, \ldots A_n \rightarrow B$, где A_i – предикат, B – заключение, в качестве которого могут выступать отношение, уравнение, факт.

Опишем более формально представление информации в базе знаний ССО, используя метаязык Бэкуса – Наура:

<Структура> ::= <уровень><имя><ссылка>

<Уровень> ::= число <Имя>::=<идентификатор>

<Ссылка> ::= число <идентификатор>

<Уравнение> ::= {УР:}<выражение> = <выражение>

<Выражение> ::= <имя>< операция><имя> ...

<Отношение> ::= <ОТН> I <ОТНУ>

<ОТН> ::= <список идентификаторов1>→<список идентификаторов 2> <реализация> <комментарий>

<ОТНУ> ::= <список идентификаторов1><условие>-->

<список идентификаторов 2><реализация> <комментарий>

<Реализация> ::= способ вычисления списка идентификаторов 2

Список идентификаторов> ::= <идентификатор>[.<идентификатор>]

<Комментарий> ::= <текст>

<Текст> ::= любая последовательность символов

<Идентификатор> ::= буква [,букваIцифра]
<Теорема> ::= <предикат> [,<предикат>] → <заключение>
<Предикат> ::=<имя предиката>(список переменных)I<имя предиката>
(список констант)I<имя предиката>(список переменных и констант)
<Список переменных> ::= переменная [,переменная]
<Список констант> ::= константа [,константа]
<Переменная> ::= $<идентификатор>
<Константа> ::= <идентификатор>
<Заключение> ::= <отношение>I<уравнение>I<предикат>
<Факт> ::= <имя предиката>(список констант)

Продукционная переменная, начинающаяся с символа «$», используется для обозначения того, что теорема относится к типу, а не к конкретному объекту. Использование такой переменной позволяет формировать теоремы, касающиеся конкретных объектов. Применение продукционных правил, записанных таким образом, представляет возможность препроцессорного описания знаний, позволяет задавать необходимые уравнения и отношения, описывающие объект. Это производится путем указания фактов в постановке задачи и заставляет срабатывать определенные правила. Тем самым упрощается входной язык по сравнению с системой ПРИЗ, а описание типов становится более гибким по отношению к возможным условиям задачи. Например, для прямоугольного треугольника можно записать теорему:

ТРЕУГОЛЬНИК ($) & ПРЯМОУГОЛЬНЫЙ ($)
→ УРАВ: $\$.AB^2 + \$.BC^2 = \$.AC^2$

Если теперь в постановке задачи имеется факт ПРЯМОУГОЛЬНЫЙ ($T1$), что свидетельствует о том, что треугольник $T1$ – прямоугольный, то в описании этого конкретного объекта будет включено уравнение

УРАВ: $T1.AB^2 + T1.BC^2 = T1.AC^2$

Имена компонентов являются составными, что позволяет идентифицировать их на принадлежность конкретным объектом. Все описания знаний хранятся в виде набора библиотек, каждой из которых соответствует определенная предметная область. Имя последней совпадает с именем библиотеки. Описание типов при этом может храниться в разделе библиотеки, причем имя раздела будет совпадать с именем типа.

Для решения задачи прежде всего необходимо построить ее вычислительную модель. Для построения последней необходимы знания, хранящиеся в семантической памяти. В нее заносится описание задачи, которое задает условия и обстановку решения. Для этой цели пользователь имеет в своем распоряжении язык запросов (входной язык ССО). Этот входной язык проще языка УТОПИСТ, поскольку он не содержит процедурной части, а основные трудности описания перенесены на предпроцессорные возможности. Основными конструкциями данного языка являются следующие операторы: опций, объектов, предметной области, рабочего объекта, фактов, продукционных правил, уравнений, отношений, исходных данных и выходных результатов. Признаком

начала оператора является символ «#», после которого следует номер (идентификатор) оператора. Рассмотрим назначение и описание каждого оператора, используя форму Бэкуса – Наура:

<Опция> :: [<тип опции>]=<значение опции i>...
<Тип опции> ::= REZ │TABL│OUT
<Значение опции1 > ::= OTL │ RAB
<Значение опции2 > ::= BAS │ PAS
<Значение опции3 > ::= D │ PR│EK

Оператор опции задает режим работы системы и является необязательным. Опция REZ со значением OTL позволяет выводить все промежуточные варианты синтеза, а со значением RAB – только конечные значения. Тип опции TABL задает таблицу операций, используемых в выражениях, при этом таблица PAS имеет более широкий круг операций, чем BAS. Тип опции OUT задает вид отображения результатов: D – на диск, PR – на печать, EK – на экран дисплея. Если не указаны значения опций, то по умолчанию принимается вывод на экран, использование расширенной таблицы операций и вывод только конечных результатов.

Для описания используемых объектов применяется оператор
<Объекты> ::= [<тип>(<имя>[,<имя>])...[<тип>(<имя>[,<имя])
<Тип> ::= буква {буква/цифра}

Данный оператор является необязательным, если в описании постановки присутствует оператор рабочего объекта, в противном случае зафиксирована ошибка. Тип является именем раздела библиотеки, куда заносится описание объекта данного типа, а имя – имя конкретного объекта, участвующего в задаче, например ТРЕУГ (*Т*1, *Т*2). Для задания предметной области, для которой решается задача, используется оператор предметной области, имеющий описание

<Предметная область> ::= <имя области><тип>[,<тип>]...

Данный оператор является обязательным, при его отсутствии выдается ошибка и работа завершается. Имя области – это имя библиотеки, в которой хранится описание типов, указанных в скобках. Далее следует оператор рабочего объекта, т.е. типа, описание которого нет в базе знаний

<Рабочий объект> :: <уровень><имя><тип>

Данный оператор обязателен, если отсутствует соответствующий тип, описываемый оператором объекта. Оператор факта имеет вид

<Факт> ::= <факт>[,<факт>]...

Оператор является необязательным. Оператор продукционных правил включает теоремы, относящиеся к рабочему объекту, или теоремы, отсутствующие в базе знаний

<Продукционное правило>::=<предикат>&[<предикат>...-><заключение>

Далее следуют операторы уравнений и отношений, которые являются необязательными. Их синтаксис рассмотрен выше. В операторе Дано описываются исходные объекты:

<Исходные данные> ::= <имя>(константа)I<имя>(константа)...

<Константа> ::= цифровая константа I символьная константа

Последним следует оператор, содержащий имена компонент (объектов) которые необходимо получить:

<Выходные результаты> ::= <имя>[,<имя>]...<имя>!!

Этот оператор является обязательным. Запрос после его формирования заносится в один из разделов библиотеки постановок, куда в первую очередь обращается система, начиная решение задачи.

8.4 Методы интеллектуального поиска информации в Интернете

Проблемы интеллектуального поиска и дальнейшей обработки релевантной информации находятся в центре внимания как исследователей, так и разработчиков ИИС. При этом важнейшими вопросами становятся автоматические методы эффективного поиска и классификации документов.

Автоматическая классификация документов текста особенно в Интернете очень важна для фильтрования информации, поиска и гипертекста. Современные инструменты распознавания текста основаны на статистическом распознавании изображений, работающем от основных особенностей документа типа гистограмм частоты термина.

Алгоритмы поиска. Поисковые системы используют различные алгоритмы ранжирования, но основные принципы определения соответствия с образцом поиска следующие [2; 11]:

− количество слов запроса в текстовом содержимом документа (т. е. в html-коде);

− тэги, в которых эти слова располагаются;

− местоположение искомых слов в документе;

− удельный вес слов, относительно которых определяется релевантность, в общем количестве слов документа.

Скрытая семантическая индексация. Метод LSA (Latent Semantic Analysis), опирающийся на векторное представление документов, может быть эффективно применен для распознавания и классификации веб-документов. В его основе лежат статистические методы анализа частоты появления слов. При достаточно оптимизированной реализации его вычислительная емкость растет почти линейно с повышением числа обрабатываемых документов. Latent Semantic Indexing (LSI) – основа для модели линейного пространства. Главная цель состоит в том, чтобы раскрыть скрытые линейные отношения между гистограммами появления слов.

Индексация документов. Проиндексировать отдельный документ, находящийся в Интернете, достаточно сложно. Первые поисковые роботы только сохраняли название документа и якори (anchor) в самом тексте, но продвинутые поисковые роботы уже используют более сложные механизмы и в целом рассматривают полное содержание документа [11].

Различие между логическими и статистическими методами анализа текстовых данных во многом повторяет разницу между векторными и растровыми графическими форматами. В первую очередь они отличаются уровнем абстракции, примененной к данным: если в первом случае имитируется восприятие информации человеком, то во втором используется подход, более удобный для обработки компьютером.

Статистические методы могут автоматически применяться для всех страниц, но не могут быть так же эффективны, как индексация страницы самим ее человеком автором. Язык HTML обеспечивает автора документа средством для того, чтобы присоединить к нему общую информацию. Это средство заключается в определении элемента <META>, например:

«<meta title = «keywords» value = «Обслуживание автомобиля БМВ» >.

Однако в этой записи не определяется никакая семантика для специфических значений атрибутов данного HTML-тэга, что значительно ограничивает его применение, а поэтому и его ценность. Это ведет к низкой «точности» относительно общего количества запрошенных документов, которые являются релевантными для конкретного запроса.

Включение механизма в виде применения булевских операторов, нахождение весов слов, как это делается в WAIS или обратной связи для релевантности, может улучшить точность поиска документов, но учитывая, что информация, находящаяся в данный момент в Интернете, очень разнообразна, эта проблема продолжает быть серьезной и наиболее эффективные пути ее решения пока не найдены. По этим причинам большинство поисковых систем ограничивается анализом метаданных – небольшой части заголовка HTML-документа, включающий описание документа и набор ключевых слов. Поскольку эти метаданные заполняются вручную создателем документа, часто они не соответствуют реальному содержанию, а то и вовсе отсутствуют [11].

Если анализ текстовой компоненты документов HTML еще допускает различные эвристические методики анализа, то сопутствующие его обогащенные ресурсы – мультимедиа, потоки, динамические страницы в отсутствие семантического описания и вовсе остаются за пределами зоны видимости автоматических средств извлечения информации из веб-пространства. То, что справедливо для веб-страниц, по которым осуществляется поиск информации, в той же мере справедливо и для результатов поиска представление результатов поиска в стандартном для веб-формате.

Таким образом, логическое представление веб-ресурсов предполагает непосредственное участие человека в создании документов и обуславливает высокую достоверность и релевантность информации, а статистическое

представление допускает автоматическую генерацию и обработку текстовых данных, но сопряжено с неизбежной (и часто недопустимо высокой) погрешностью.

Методика интеллектуальной обработки документов в Интернете может заключаться в следующем. Документ в формате HTML разбирается при помощи лексического анализатора, после чего из полученного текстового потока выбрасываются шумовые символы и нулевые слова (знаки препинания, спецсимволы и слова из одной буквы), а остальные слова подаются на вход программы морфологического анализа, которая возвращает нормализованную форму слова и набор гипотез о его морфологической природе.

Подсчитывается относительная частота полученных таким образом слов в нормальной форме, относящихся к основным частям речи (существительные, прилагательные, глаголы и наречия). Полученное представление документа в виде вектора весов включается в матрицу, в которой строки представляют документы, а столбцы – ключевые слова. После этого можно определить значимость данного слова для содержащего его документа как отношение его относительной частоты встречаемости в данном документа к сумме частот появления во всех документах рассматриваемого набора. В случае непрерывного потока документов следует подсчитывать не относительные, а абсолютные частоты слов и нормализовать эти значения по размерам документов и сумме обработанных слов.

Индексация документов скрытым семантическим анализом. Рассмотрим еще один метод для автоматической индексации документов в вебпространстве. Подход состоит в том, чтобы улучшить обнаружение требуемых документов на основе терминов, найденных в запросах. Специфическая используемая техника – «разложение с сингулярным значением, в котором большой объем матрицы документов расчленен в набор нескольких десятков ортогональных коэффициентов. Запросы представлены как векторы псевдодокумента, сформированные из взвешенных комбинаций терминов. Сопоставление векторов документов и запросов в пространстве может служить основанием для их релевантности (чем ближе они расположены друг к другу, тем больше степень подобия запроса и документа)» [5].

Представление линейного пространства. Главное преимущество использования метода ICA (Indexing Complex Analysis) то, что представление запроса является лучше выровненным к структуре группы, чем в аналогичном подходе. Модель линейного пространства включает выполнение трех шагов: индексация, взвешивание термина и мера подобия. Метод основан на отдельной статистике слова, следовательно, сначала создается набор терминов всех слов, встречающихся в базе данных. Этот набор терминов освобождается от слов, которые не помогают дифференцировать документы в терминах смысла. Это называется *индексация.* Для этого находятся наиболее часто встречающиеся слова и вносятся в список для удаления. Устраняем редкие слова, которые встречаются только в нескольких документах [5].

*Вычисление количеств сравнения***.** Имеются в основном три вида сравнений: для двух терминов («Насколько похожи термины *i* и *j* ?»); двух документов («Насколько похожи документы *i* и *j* ?»); сравнение термина и документа («Как связаны термин *i* и документ *j* ?»). В стандартных подходах информационного поиска это приводит соответственно к сравнению двух рядов, сравнению двух столбцов или исследованию отдельных ячеек исходной матрицы терминов данного документа − *X*. Рассмотрим подобные сравнения, но используем матрицу *Xhihat*, так как предполагается представить важные и надежные объекты, лежащие в основе данных в *X*. Как только $Xhihat = TSD'$, нужные количества могут быть вычислены, только используя меньшие матрицы, *T, D* и *S*.

Пример [19]. Пусть выбраны заголовки 9 документов. Сначала из этих заголовков были исключены стоп-символы. Это слова, которые встречаются в каждом тексте и не несут в себе смысловой нагрузки, это прежде всего все союзы, частицы, предлоги и множество других слов. Дальше были исключены слова, встречающиеся в единственном экземпляре. Это тоже необязательный шаг, он не влияет на конечный результат, но упрощает математические вычисления. Остались индексируемые слова, выделенные жирным шрифтом:

1. Британская **полици**я знает о местонахождении **основателя WikiLeaks.**
2. В **суде США** начинается процесс **против** россиянина, рассылавшего спам.
3. **Церемонию вручения Нобелевской премии** мира бойкотируют 19 **стран.**
4. В **Великобритании арестован основатель** сайта **Wikileaks** Джулиан Ассандж.
5. Украина игнорирует **церемонию вручения Нобелевской премии**.
6. Шведский **суд** отказался рассматривать апелляцию **основателя Wikileaks.**
7. НАТО и **США** разработали планы обороны **стран** Балтии **против** России.
8. **Полиция Великобритании** нашла **основателя WikiLeaks**, но не **арестовала**.
9. В Стокгольме и Осло сегодня состоится **вручение Нобелевских премий**.

Латентно-семантический анализ. На первом шаге составим частотную матрицу *Xhihat* индексируемых слов. В этой матрице строки соответствуют индексированным словам, а столбцы – документам. В каждой ячейке матрицы указано, какое количество раз слово встречается в соответствующем документе (рисунок 8.1).

Следующим шагом проводим сингулярное разложение полученной матрицы. Математическая операция, раскладывающая матрицу на три составляющих. То есть исходную матрицу представляем в виде *Xhihat = T*D*S*, где *T* и *D* – ортогональные матрицы; *S* – диагональная матрица (рисунок 8.2). Причем диагональные элементы матрицы S упорядочены в порядке убывания. Диагональные элементы матрицы W называются сингулярными числами.

	T1	T2	T3	T4	T5	T6	T7	T8	T9
wikileaks	1	0	0	1	0	1	0	1	0
арестова	0	0	0	1	0	0	0	1	0
великобритан	0	0	0	1	0	0	0	1	0
вручен	0	0	1	0	1	0	0	0	1
нобелевск	0	0	1	0	1	0	0	0	1
основател	1	0	0	1	0	1	0	1	0
полиц	1	0	0	0	0	0	0	1	0
прем	0	0	1	0	1	0	0	0	1
прот	0	1	0	0	0	0	1	0	0
стран	0	0	1	0	0	0	1	0	0
суд	0	1	0	0	0	1	0	0	0
сша	0	1	0	0	0	0	1	0	0
церемон	0	0	1	0	1	0	0	0	0

Рисунок 8.1 – Матрица *Xhihat*

wikileaks	0.57	-0.01	0.01	-0.2	0.13	0.16	-0.16	-0.25	-0.64
арестова	0.34	-0	0.07	0.41	-0.42	-0.02	0.1	0.17	0.01
великобритан	0.34	-0	0.07	0.41	-0.42	-0.02	0.1	0.17	-0.01
вручен	0	0.52	0.07	-0.06	-0.08	-0.15	-0.17	0.02	-0.07
нобелевск	0	0.52	0.07	-0.06	-0.08	-0.15	-0.17	0.02	0.32
основател	0.57	-0.01	0.01	-0.2	0.13	0.16	-0.16	-0.25	0.64
полиц	0.31	-0	0.05	0.07	0.57	-0.6	0.29	0.37	-0
прем	0	0.52	0.07	-0.06	-0.08	-0.15	-0.17	0.02	-0.25
прот	0.02	0.03	-0.61	0.13	-0.05	-0.22	0	-0.25	0
стран	0.01	0.22	-0.31	0.39	0.41	0.56	-0.22	0.4	-0
суд	0.12	0.01	-0.38	-0.62	-0.3	0.12	0.21	0.55	-0
сша	0.02	0.03	-0.61	0.13	-0.05	-0.22	0	-0.25	0
церемон	0	0.38	0.03	0.02	0.08	0.31	0.82	-0.29	0

·

3.41	0	0	0	0	0	0	0	0
0	3.3	0	0	0	0	0	0	0
0	0	2.27	0	0	0	0	0	0
0	0	0	1.49	0	0	0	0	0
0	0	0	0	1.18	0	0	0	0
0	0	0	0	0	0.98	0	0	0
0	0	0	0	0	0	0.71	0	0
0	0	0	0	0	0	0	0.43	0
0	0	0	0	0	0	0	0	0

·

T1	T2	T3	T4	T5	T6	T7	T8	T9
0.43	0.05	0.01	0.54	0	0.37	0.01	0.63	0
-0	0.02	0.65	-0.01	0.59	-0	0.09	-0.01	0.47
0.03	-0.7	-0.04	0.06	0.1	-0.16	-0.67	0.09	0.09
-0.22	-0.24	0.15	0.28	-0.11	-0.68	0.44	0.33	-0.13
0.69	-0.32	0.22	-0.49	-0.12	-0.03	0.27	-0.02	-0.19
-0.27	-0.34	0.44	0.29	-0.13	0.45	0.12	-0.31	-0.45
-0.03	0.3	0.14	-0.17	0.44	-0.15	-0.3	0.24	-0.71
-0.3	0.12	0.4	-0.39	-0.53	0.12	-0.23	0.46	0.13
0.35	0.35	0.35	0.35	-0.35	-0.35	-0.35	-0.35	0

Рисунок 8.2 – Разложение матрицы M

Удобство сингулярного разложения состоит в том, что оно выделяет ключевые составляющие матрицы, позволяя игнорировать шумы. Согласно простым правилам произведения матриц видно, что столбцы и строки, соответствующие меньшим сингулярным значениям, дают наименьший вклад в итоговое произведение. Например, можно отбросить последние столбцы матрицы *D* и последние строки матрицы *W*, оставив только первые 2 (рисунок 8.3).

wikileaks	0.57	-0.01
арестова	0.34	-0
великобритан	0.34	-0
вручен	0	0.52
нобелевск	0	0.52
основател	0.57	-0.01
полиц	0.31	-0
прем	0	0.52
прот	0.02	0.03
стран	0.01	0.22
суд	0.12	0.01
сша	0.02	0.03
церемон	0	0.38

·

3.41	0
0	3.3

·

T1	T2	T3	T4	T5	T6	T7	T8	T9
0.43	0.05	0.01	0.54	0	0.37	0.01	0.63	0
-0	0.02	0.65	-0.01	0.59	-0	0.09	-0.01	0.47

Рисунок 8.3 – Двумерное сингулярное разложение

При этом гарантируется оптимальность полученного произведения. Разложение такого вида называют двумерным сингулярным разложением. Отметим на графике точки, соответствующие отдельным текстам и словам, получится картинка распределения слов (рисунок 8.4).

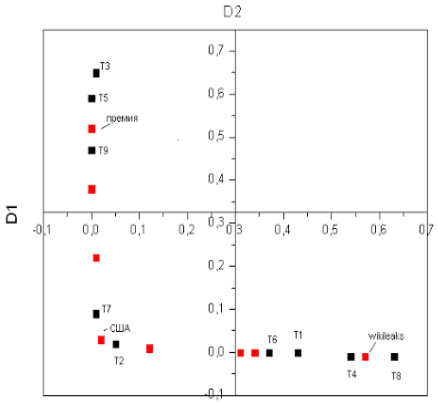

Рисунок 8.4 – Распределение слов документов

Из рисунка 8.4 видно, что статьи образуют три независимые группы, первая группа статей располагается рядом со словом «wikileaks», и если посмотрим названия этих статей, становится понятно, что они имеют отношение к wikileaks. Другая группа статей образуется вокруг слова «премия», в них идет обсуждение Нобелевской премии. Другая группа статей образуется вокруг слова США.

8.5 Методы обработки текстов в поисковых машинах

Понятие поиска. Одним из основных способов найти информацию в Интернете являются поисковые машины, ежедневно они сканируют сети Интернет, посещают веб-страницы и заносят их в свои базы данных. Это позволяет пользователю сформировать ключевые слова, нажать «submit» и увидеть, какие страницы удовлетворяют его запросу. Понимание того, как работают поисковые машины, просто необходимо веб-мастерам. Для них важна правильная, аналогичная представлению в поисковых машинах структура документов и всего сервера или сайта. Без этого документы будут не часто появляться в ответ на запросы пользователей к поисковой машине или даже вовсе могут быть не проиндексированы [9; 14].

Веб-мастера хотят повысить рейтинг своих страниц, что понятно: ведь на любой запрос к поисковой машине могут быть выданы сотни и тысячи отвечающих ему ссылок на документы. В большинстве случаев только 10 первых ссылок обладают достаточной релевантностью к запросу. Хочется, чтобы документ оказался в первой десятке, поскольку большинство пользователей не часто просматривает следующие за первой десяткой ссылки. Если ссылка на документ будет больше десяти, это также плохо, как если бы ее не было вовсе. В основе интеллектуального поиска в Интернете лежат методы поиска соответствующей заданным условиям информации. Рассмотрим основные характеристики такого поиска [9; 14].

Особенности поисковых машин. Каждая поисковая машина обладает рядом особенностей (реализует те или иные методы). Эти особенности следует учитывать при формировании своих страниц.

Тип поисковой машины. «Полнотекстовые» поисковые машины индексируют каждое слово на веб-странице, исключая лишь некоторые незначащиеся слова (союзы, предлоги). «Абстрактные» поисковые машины создают некий образ каждой страницы. Для веб-мастеров полнотекстовые машины полезней, поскольку любое слово, встречающееся на поисковой странице, подвергается анализу при определении его релевантности к запросам посетителей. Но для абстрактных поисковых машин может случиться, что страницы проиндексированы лучше, чем для полнотекстовых. Это может исходить от алгоритма экстрагирования, например, по частоте употребления в странице одних и тех же слов.

Размер поисковой машины определяется количеством проиндексированных страниц. Приведенные в таблице значения не слишком точны, но могут прояснить некоторые моменты. Например, в поисковой машине с большим размером могут быть проиндексированы почти все ваши страницы, при среднем объеме ваш сервер может быть частично проиндексирован, а при малом объеме ваши страницы могут вообще не попасть в каталоги поисковой машины.

Период обновления. Поскольку веб-пространство изменяется непрерывно, поисковые машины индексируют все без учета даты. Однако в каждый момент времени ссылки, выдаваемые в ответ на запросы пользователей, могут быть однодневной давности, а могут и месячной давности, а то и больше. Рассмотрим причины, по которым это происходит [9]:

− одни поисковые машины сразу индексируют страницу по запросу пользователя, а затем продолжают индексировать еще не проиндексированные страницы;

− вторые чаще могут сканировать более популярные страницы сети, чем менее посещаемые.

Дата индексирования документа. Некоторые поисковые машины показывают дату, когда был проиндексирован тот или иной документ. Это помогает пользователю понять, какой по сроку ссылку выдает поисковая система. Другие оставляют пользователям только думать об этом.

Указанные и не указанные страницы. В лучшем случае поисковые машины должны найти все страницы любого сервера в результате прохода по ссылкам. В реальности ситуация другая. Станицы серверов гораздо раньше появляются в индексах поисковых систем, если их указать дополнительно (Add URL). Не указанные отмечаются как non-submitted страницы. Если хотя бы одна страница сайта указана, то поисковые машины точно найдут другие страницы по ссылкам из данной. Но на это требуется больше времени. Отдельные машины сразу индексируют весь сайт, но большинство, однако, записав указанную страницу в индекс, оставляют индексирование сервера на потом.

Глубина индексирования. Этот параметр относится только к не указанным страницам. Он показывает, сколько страниц после указанной будет индексировать поисковая система. Большинство машин не имеют ограничений по глубине индексирования. На практике же это не совсем так. Вот несколько причин, по которым могут быть проиндексированы не все страницы: не слишком аккуратное использование фреймовых структур (без дублирования ссылок в управляющем (frameset) файле); использование параметра imagemap без дублирования их ссылками.

Поддержка фреймов. Если поисковый робот не умеет работать с фреймовыми структурами, то многие структуры с фреймами будут пропущены при индексировании. Поддержка ImageMap. Тут та же проблема, что и с фреймовыми структурами сайтов.

Защищенные паролями директории и сервера. Некоторые поисковые машины могут индексировать такие сервера, если им указать параметры «Username и Password». Это сделано для того, чтобы пользователи видели, что есть на данном сервере. Это позволяет сразу узнать, что такая информация есть, и, возможно, посетители подпишутся на эту информацию.

Частота появления ссылок. Основные поисковые машины могут определить популярность документа по тому, как часто на него ссылаются из других источников. Другие машины на основании этих данных «делают вывод», стоит или не стоит тратить время на индексирование подобного документа. Если сервер обновляется часто, то поисковая машина чаще будет его реиндексировать, если не часто – может индексировать редко.

Контроль индексации. Он показывает, какими средствами можно управлять той или иной поисковой машиной. Известные поисковые машины руководствуются предписаниями файла robots.txt. Другие поддерживают контроль с помощью META-тегов из самих индексируемых документов.

Перенаправление (redirect). Некоторые сайты перенаправляют посетителей с одного сервера на другой, и параметр *redirect* показывает, какой URL будет связан с этими документами. Это означает, что если поисковая машина не отрабатывает перенаправление, то могут возникнуть проблемы с несуществующими файлами.

Стоп-слова. Некоторые поисковые машины не включают определенные слова в свои индексы или могут не включать эти слова в запросы пользователей. Такими словами могут считаются предлоги или часто использующиеся слова. Например, Altavista игнорирует слово Web и для запросов типа Web developer будут выданы ссылки только по второму слову. Существуют способы избежать этого, заменяя эквивалентным словом [9; 14].

*Влияние на алгоритм определения релевантности***.** Поисковые машины часто используют расположение и частоту повторения ключевых слов в документе. Но дополнительные механизмы увеличения степени релевантности для каждой машины различны. Этот параметр показывает, какие именно механизмы существуют для той или иной машины.

Spam-штрафы. Все крупные поисковые системы «не приветствуют», когда какой-либо сайт пытается повысить свой рейтинг путем, например, многократного указания себя через Add URL или множественного упоминания одного и того же ключевого слова и т. д. В большинстве случаев подобные действия (spamming, stacking) наказываются, и рейтинг сайта понижается.

Поддержка META-тегов. По сути все поисковые машины должны учитывать метаданные при индексации страниц, однако на практике не все это выполняют.

Title. Данный параметр показывает, как поисковые машины генерируют заголовки ссылок для пользователя в ответ на его запрос.

Description. Этот параметр показывает, как поисковые машины генерируют описания ссылок для пользователя в ответ на его запрос.

Проверка статуса URL. Очень важный для веб-дизайнера элемент поисковой машины – можно ли проверить, насколько глубоко проиндексирован его сервер и попал ли он в индекс поисковой машины.

Удаление старых данных. Параметр, определяющий действия веб-дизайнера при закрытии сервера или перемещении его на другой адрес. Возможны два варианта: либо удалить старое содержание, либо переписать файл robots.txt.

Удаление содержимого: когда поисковая машина попытается реиндексировать документы и не найдет их, старые ссылки в индексе будут удалены. В этом случае все зависит от периода обновления данных для поисковой машины.

Robots.txt: когда поисковая машина запросит этот файл и «получит», что сервер закрыт от индексации, то все ссылки на файлы этого сервера будут удалены из индекса [9; 14].

8.6 Поисковые машины в Интернете

Основные поисковые машины. Какие из сотен поисковых машин действительно важны для веб-мастера? Конечно, широко известные и часто используемые. Но при этом следует учесть ту аудиторию, на которую рассчитан сервер. Например, если он содержит узкоспециальную информацию, то вряд ли стоит использовать поисковые системы общего назначения. В этом случае лучше обменяться ссылками с коллегами, которые занимаются сходными вопросами. Существует два вида информационных ресурсов о веб-страницах: пауки и каталоги [2; 9; 14].

Пауки: (spiders, crawlers) постоянно исследуют Интернет с целью пополнения своих баз данных документов. Обычно это не требует никаких усилий со стороны человека. Примером может быть поисковая система Altavista. Для поисковых систем довольно важна конструкция каждого документа. Значение имеют title, meta-теги и содержимое страницы [9; 14].

Каталоги. В отличие от поисковых машин в каталог информация заносится по инициативе человека. Добавляемая страница должна быть жестко привязана к принятым в каталоге категориям. Примером каталога может служить Yahoo. Конструкция страниц значения не имеет. Рассмотрим основные поисковые машины [9; 16–18].

Google (работает с 1998 года). Лидер поисковых машин Интернета занимает более 90 % мирового рынка и индексирует более десятки миллиардов вебстраниц. Google может находить информацию более чем на 185 языках. Google поддерживает поиск в документах форматов PDF, RTF, PostScript, Microsoft Word, Microsoft Excel, Microsoft PowerPoint и других. В 2024 году Google обрабатывает 90,91 % всех поисковых запросов в мире.

Bing – это вторая по популярности в мире поисковая система, разработанная и управляемая Microsoft. Она предлагает различные функции поиска, включая поиск в Интернете по изображениям и по картам. Наряду с расширенными функциями поиска на основе искусственного интеллекта. Помимо основных возможностей поиска, Bing известен своим удобным интерфейсом и интеграцией с другими продуктами и услугами Microsoft, включая Windows 11, Xbox и Microsoft 365. Интерфейс достаточно простой, с динамически обновляемым фоном и минимальным количеством ссылок. Есть поиск по картинкам, видео, словам или новостям. Поисковые системы Bing установлены по умолчанию в смартфонах на ОС Windows. Больше всего трафика у этого поисковика в США, Германии и Великобритании. В 2024 году доля этого поисковика составляет 3,64 %.

Yahoo – это самый первый в мире поисковик. Он появился в 1994 году, когда современные известные поисковые системы еще не были созданы. Поиск в Yahoo выполняется по словам, новостям, картинкам и видео на основе Bing. В 2024 г. он на третьем месте в мире, занимает 1,61 % международного рынка поисковых систем.

Baidu занимает четвертое место во всем мире за счет огромной популярности в Китае, аналогично Google в США или Yandex в России. Каждый месяц им пользуется более 610 млн пользователей. Среди стран СНГ китайские популярные поисковики не пользуются спросом из-за навязчивых расширений для перевода, которые воспринимаются браузерами как вредоносное ПО. В Baidu встроена энциклопедия, в Китае она опережает по популярности Википедию. В 2024 г. занимает 1,15 % мирового рынка.

Yahoo. Старейший каталог Yahoo был запущен в начале 1994 года. Достаточно известен. В марте 1996 запущен еще один каталог Yahoo – Yahooligans для детей. Появляются все новые и новые региональные и top-каталоги Yahoo. Поскольку Yahoo основан на подписке пользователей, в нем может не быть некоторых сайтов. Если поиск по Yahoo не дал подходящих результатов, пользователи могут воспользоваться поисковой машиной. Это делается очень просто. Когда делается запрос к Yahoo, каталог переправляет его к любой из основных поисковых машин. Первыми ссылками в списке, удовлетворяющих запросу адресов, идут адреса из каталога, а затем идут адреса, полученные от поисковых машин [9]. В 2024 г. занимает 1,13 % мирового рынка.

DuckDuckGo поисковик достаточно популярен благодаря тому, что никак не отслеживает действия пользователя в Интернете. Имеет дружественный интерфейс, переведен на разные языки, в том числе русский и украинский. Это гибридный поисковик, главные черты которого – классный поиск, отсутствие

поискового спама, минимальное количество рекламы (ее можно полностью отключить в настройках) и щепетильное отношение к модному на западе слову «privacy»: не следит за пользователями, сохраняя их анонимность.

Ask.com выдает разные варианты ответов на заданные вопросы в виде актуальных информационных статей. Чтобы начать пользоваться этим поисковиком, достаточно задать вопрос в поисковой строке.

Ecosia – поисковая система, созданная в Германии с целью финансирования социальных нужд, – организация направляет 80 % доходов от рекламы на высадку деревьев. Партнером поисковика является Bing. Для посадки одного дерева требуется приблизительно 45 поисков. Среди других поисковых систем доля Ecosia составляет около 0,10 %.

AOL входит в топ-10 поисковых систем Интернета, занимая 9 место. Название поисковика расшифровывается как «America Online». Но AOL уже не пользуется большим спросом из-за появления популярных поисковых гигантов.

Internet Archive – поисковая система, содержащая архив данных с 1996 года, замыкает топ-10 поисковых систем. В ней можно проверить, как сайт выглядел в любой заданный период времени с момента создания веб-архива. Это полезный инструмент, который позволяет проверить историю домена и узнать, как он менялся в течение времени.

Российские поисковые машины. Yandex. На yandex.ru первая страница работает в режиме обработки запроса на естественном языке. Такой возможности нет у других поисковых систем (это больше, чем просто поиск по всем словам, указанным в запросе, а поиск с «пониманием;). При этом желающие указывать в запросе логические операторы, могут пользоваться расширенным поиском со страницы. В 2024 году Яндекс обрабатывает 65,12 % всех поисковых запросов в России. На странице выдачи результатов добавлена возможность <Найти похожие документы>, чего опять же нет у других российских поисковых систем. С 1998 года в поисковой системе Яndex появилась возможность осуществлять повторный поиск только в найденных документах (нужно в случае, если найдено слишком много документов для уточнения запроса).

Поисковые машины в Беларуси [18]. По умолчанию поиск осуществляется по всем проиндексированным документам в базах.by, по содержимому html тегов title, description, keywords, author и по текстовой части документов с логикой «and» для всех слов запроса, а также в Яндексе по региону «Беларусь» по технологии Яндекс. При оптимизации данным тегам стоит уделить повышенное внимание. Белорусскими считаются все сайты, представленные в собственном каталоге, поэтому регистрация сайта в каталоге.by обязательна.

В цифрах: в 2023 году среди пользователей Интернета в Беларуси лидирует поисковая система Google с большим отрывом – 92,46 %. На втором месте – поисковик Яндекс – 3,3 %. Третий – mail.ru. Среди всех браузеров самыми популярными являются: Google Chrome – 59,37 %; Яндекс – 12,46 %; Safari – 8,89 %; Opera – 6,82 %.

Рассмотрим несколько общих советов, которые могут помочь увеличению появления ваших страниц в листингах поисковых машин [9; 14].

Достижение оптимального индексирования сервера поисковыми машинами. Каждая поисковая машина определяет релевантность по-своему. В довершение всего появляются новые страницы, старые обновляются – соответственно меняется и листинг. На странице должен быть текст, поскольку поисковые машины индексируют именно его. Страница с недостаточным количеством текста имеет мало шансов попасть в список ответа на запрос пользователя. Текст на картинке не может быть распознан поисковой машиной, поэтому рекомендуется записывать в тег ALT не только название рисунка, но и важнейшие слова из него [12].

Принцип перевернутой информационной пирамиды. Необходимо в начале документа выставлять главный смысл, что полезно как для людей, так и для поисковых машин. Однако главное, что видно человеку в документа, не всегда может находиться в его начале. Например, при табличной организации документа его релевантность некоторому запросу может оказаться ниже, чем при простой страничной организации того же документа, поэтому надо включать аннотацию документа в его начало и в META-теги.

Выбор ключевые слова. Несколько ключевых слов в заголовке может оказаться более сильным аргументом в определении релевантности запросу, чем слова в документе. Использование синонимов в ключевых словах не обязательно окажется решающим фактором при определении релевантности. Одно «экстра-слово» иногда лучше помогает, чем подборка синонимов.

Указание ключевые страницы сервера. Большинство поисковых машин индексируют страницы по гипертекстовым ссылкам из указанной (submitted) страницы. Иногда они ошибаются, поэтому полезно указывать первые три уровня дерева страниц сайта или те страницы, которые наилучшим образом отражают его суть. Необходимо наличие ссылок на внутренние страницы. Поисковые машины отыскивают страницы именно по гипертекстовым ссылкам. Чем больше ссылок внутри дерева документов, тем больше вероятность того, что ни одна страница не будет забыта при индексировании. Верно и обратное: если есть ссылки на другие, внешние сервера, то есть вероятность, что поисковый робот уйдет по этим ссылкам, не до конца проиндексировав страницы [12].

Фреймы могут ослабить поиск. Некоторые поисковые машины не могут проводить индексацию по ссылкам из FRAMESET. Чтобы избежать этого, необходимо обеспечить альтернативный вход и индексацию страниц, использовать META-теги или упростить дизайн. META-теги могут помочь контролировать действия поисковых роботов и машин, однако некоторые машины «умеют» работать со всеми тегами, некоторые – только с несколькими тегами, остальные и вовсе на воспринимают подобные вещи. Поэтому использование META-тегов не гарантирует, что страницы окажутся первыми в списке ответов на пользовательский запрос к машине.

Интерактивность. Некоторые поисковые машины не индексируют страницы через CGI и из баз данных. Поэтому рекомендуется создание статических страниц везде, где это возможно, использование баз данных для обновления уже существующих статических страниц. Плохо относятся поисковые роботы к

специальным символам в URL, например, к символу '?' Если страницы уже попали в индексы поисковых машин, нужно периодически проверять это. Если заметили подобные вещи – укажите страницы поисковой машине еще раз.

Повторное указание страниц. Многие поисковые машины индексируют сайты в соответствии с их изменением. Хорошо, если сайт редко изменяется, и реиндексация его происходит один-два раза в год. Если же сайт изменяется часто, рекомендуется указывать страницы поисковым машинам, раз в один-два месяца. Это может гарантировать, что содержимое индекса в поисковых машинах не будет отличаться от реального содержания страниц.

*Копирование фреймов***.** Большинство поисковых машин не умеют работать с фреймовыми структурами, не проверяют ссылки, определенные в структуре FRAMESET. Для того чтобы поисковые машины отработали такие ссылки, следует предпринять специальные меры по дублированию ссылок из FRAMESET в конструкции NOFRAMES. В содержимое NOFRAMES вставляется полный каталог документов сервера. Получается типичная карта сервера, пусть не такая как с использованием фреймов, зато пользователи видят, куда им дальше податься, и роботы поисковых машин могут идти по ссылкам и индексировать содержимое. Но у этого решения есть два узких места [12]:

1. Веб-дизайнеру при обновлении сервера приходится следить за корректностью фреймовой части и не забывать обновлять содержимое NOFRAMES.

2. Если пользователь попал не на заглавную страницу сервера, он увидит документы без фреймов. Поэтому при создании сервера с использованием фреймов надо не забывать включать средства навигации в каждый документ.

Измерение популярности. Лучший способ определить, как попадают на сервер, – посмотреть файл статистики (если он содержит поле HTTP_REFERER). Можно сделать это и с помощью поисковых машин. Общий для всех машин способ – набрать имя сервера, однако в этом случае в результаты запроса попадут и страницы самого сервера [12]. Списки наиболее популярных запросов к поисковым машинам можно посмотреть: через WebCrawler Search Ticker (20 случайно выбранных запросов в реальном времени); Yahoo Top 200 Search Words (200 самых популярных запросов к Yahoo) [12].

Резервирование веб-страниц. Для резервирования страниц был создан Архив Интернета и Машина времени (Wayback Machine), чтобы сохранить существующий контент для будущих поколений. Но в Архив Интернета попадают далеко не все страницы. В кэш Google попадает больше, но там определенный срок хранения. Для создания собственного архива веб-страниц используются сервисы: Archive-It: курируемая служба веб-архивирования; ArchiveWeb.page (десктопная программа и расширение для Chrome для создания веб-архивов); Brozzler: опенсорсная утилита, которая для скачивания контента использует настоящий браузер (Chrome или Chromium), а также youtube-dl и rethinkdb и т. д. [19].

8.7 ИС в распознавании образов

Традиционное направление ИИ, берущее начало у самых его истоков. Каждому объекту ставится в соответствие матрица признаков, по которой происходит ее распознавание. Это направление близко к машинному обучению, тесно связано с нейрокибернетикой. Также это научно-техническое направление, связанное с разработкой методов и построением систем на базе специализированных компьютеров для установления принадлежности некоторого объекта (предмета, процесса, явления, ситуации, сигнала) к одному из заранее выделенных классов объектов (образу) [2; 10].

Процесс распознавания основан на сопоставлении признаков, характеристик исследуемого объекта с признаками, характеристиками других известных объектов, в результате чего делается вывод о наиболее правдоподобном их соответствии. Методы распознавания образов используют в технической и медицинской диагностике. В последнее время широко используются в различных интеллектуальных системах безопасности, системах интеллектуальных зданий и т. д.

Как известно, решение задачи о разделении на два класса эквивалентно решению системы линейных неравенств. Если заданы два множества образов, принадлежащих соответственно классам ω_1 и ω_2, то решение ищется в виде вектора весов **w**, обладающего тем свойством, что для всех образов класса ω_1, выполняется условие $\mathbf{w}'\mathbf{x} > 0$ и для всех образов класса ω_2 − условие $\mathbf{w}'\mathbf{x} < 0$. Если образы класса ω_2 умножить на −1, то эквивалентное условие $\mathbf{w}'\mathbf{x} > 0$ становится общим для всех образов. Обозначив через *N* общее количество пополненных выборочных образов обоих классов, нашу задачу можно свести к отысканию вектора весов **w**, обеспечивающего справедливость системы неравенств (8.1).

$$\mathbf{X}\boldsymbol{w} > 0, \tag{8.1}$$

где X определяется по формуле (8.2).

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1' \\ \mathbf{x}_2' \\ \vdots \\ \mathbf{x}_N' \end{pmatrix}, \tag{8.2}$$

$\mathbf{w} = (\omega_1, \omega_2, \ldots, \omega_n, \omega_{n+1})'$ и **0** − нулевой вектор. Если образы обладают хорошим размещением, матрица **X** удовлетворяет условию Хаара, т. е. всякая подматрица $(n+1)\times(n+1)$ -матрицы **X** имеет ранг $n+1$.

Если вектор весов **w**, удовлетворяющий условию (8.1), существует, то неравенства называются *совместными*; в противном случае они *несовместны*. На языке распознавания образов говорят, что классы соответственно *разделимы* или *неразделимы*. Надо иметь в виду, что формулировка условия (8.1) предполагает, что все образы одного из классов умножены на −1.

В принципе для решения (8.1) можно воспользоваться и детерминистским и статистическим подходами. Детерминистский подход служит основой алго-

ритмов, которые конструируются независимо от каких-либо предположений о статистических свойствах классов образов. Статистические алгоритмы, отражают, с другой стороны, попытку найти аппроксимацию плотностей распределения $p(\omega_i | \mathbf{x})$ и использовать их затем в качестве байесовских решающих функций. Завершив изучение обоих подходов, заметим сходство между статистическими и детерминистскими алгоритмами.

В фундаментальной книге [14] даются понятия и детализации цифровых изображений, излагаются базовые методы их получения и обработки. Рассмотрены примеры решения прикладных задач, в основе которых лежит последовательное применение описанных алгоритмов распознавания изображений, представлены инструментарии их обработки – система Матлаб и библиотека OpenCV.

8.8 Интеллектуальные технологии в защите информации

Вместе с традиционными средствами защиты корпоративных систем, такими как антивирусы, детекторы уязвимостей, межсетевые экраны и детекторы вторжений, широко применяются средства автоматизации защиты, включающие коррелляторы событий, программы обновлений, средства аутентификации, авторизации и администрирования, системы управления рисками. Средства о применении интеллектуальных систем защиты информации (ИСЗИ) посвящены системам обнаружения атак, в качестве интеллектуального инструмента в которых используются *нейронные сети (НС), системы нечеткой логики* и основанные на правилах *экспертные системы (ЭС)* [3; 13].

Использование ЭС. Правила записываются в БЗ экспертной системы, которая обрабатывается подсистемой вывода. В ИС ЗИ экспертные системы в БЗ содержат описание классификационных правил, соответствующих профилям легальных пользователей и сценариям атак на ИС. Недостатки: система не является адаптивной, не всегда обнаруживаются неизвестные атаки.

Использование нейронных систем (НС). Если НС представлена в виде отдельной системы обнаружения атак, при обработке трафика происходит анализ информации на наличие злоупотреблений. Случаи с указанием на атаку перенаправляются к администратору безопасности. Подход быстродействующий, поскольку используется один уровень анализа. В сравнении с предыдущим подходом данный принцип обладает преимуществом в скорости, так как существует только один уровень анализа, а сама система обладает свойством адаптивности. НС применяют в системах криптографической защиты информации для хранения криптографических ключей в распределенных сетях. Одним из основных недостатков НС является «непрозрачность» формирования результатов анализа [3; 13].

Комбинированный метод. В системах обнаружения атак можно выделить применение НС, дополненных ЭС. Чувствительность системы возрастает, так как ЭС получает данные только о событиях, которые рассматриваются в каче-

стве подозрительных. Если НС за счет обучения стала идентифицировать новые атаки, то ЭС следует обновить, иначе новые атаки будут ею игнорироваться, прежние правила не способны распознавать данную угрозу. Использование гибридных нейро-экспертных или нейро-нечетких систем позволяет отразить в структуре системы нечетких предикатных правил, которые автоматически корректируются в процессе обучения нейронной сети. Свойство адаптивности нечетких НС позволяет решать отдельно взятые задачи идентификации угроз, сопоставления поведения пользователей с имеющимися в системе шаблонами, автоматически формировать новые правила при изменении поля угроз, а также реализовать систему ЗИ технической системы в целом.

В Беларуси и России для обнаружения и противодействия несанкционированным действиям используют различные математические методы и интеллектуальные инструменты. В ряде работ по защите информации рассматривается использование интеллектуальных мультиагентных систем: дается обзор инструментов реализации атак, онтология предметной области, определяются структура команды агентов средств защиты информации, механизмы их взаимодействия и координации. Используется аппарат НС и генетических алгоритмов для защиты сетей от программных атак, направленных на нарушение доступности ресурсов. НС используется для обнаружения признаков атак в сетевом трафике, идентификации форматов передаваемых данных, динамической идентификации участников обмена, а генетические алгоритмы – для получения близкого к оптимальному решению в задачах управления маршрутами и параметрами трафика при наличии нечеткости данных идентификации атаки в условиях дефицита информации или информационного «шума» [3; 13].

Для реализации активного аудита безопасности работы ИС применяется аппарат нечетких множеств. Для определения уровня защищенности сетей от угроз несанкционированного доступа, обнаружения злоупотреблений пользователей и программных атак применяются методы интеллектуального анализа данных, работающие по принципу адаптивной защиты от несанкционированного доступа – «анализ, прогнозирование, предупреждение». Ряд исследований посвящен идентификации и аутентификации пользователя по биометрическим, фонетическим параметрам, которые используют математический аппарат НС, комбинированные методы быстрой цифровой обработки сигналов и НС [3; 13].

Проведенный анализ приводит к выводу о необходимости решения не конкретных задач защиты информации с применением НС, систем нечеткой логики, ЭС, а формирования единого подхода к применению интеллектуальных средств для создания комплексной адаптивной защиты систем информационных технологий на основе биоаналогии. Проектирование ИСЗИ следует осуществлять как единый процесс построения адаптивной системы с внутренне присущими функциями защиты информации.

Для решения поставленной задачи лучшим сочетанием свойств обладают нечеткие НС, они сочетают в себе достоинства НС и нечеткой логики, опирающейся на опыт экспертов информационной безопасности. Механизм нечеткого логического вывода позволяет использовать опыт экспертов, представленный в

виде системы нечетких предикатных правил, для предварительного обучения нечеткой НС. Рассмотрим следующие свойства нечетких НС, необходимые для адаптивных средств защиты информации: распределенный параллелизм вычислений; адаптивность нейросредств ЗИ (нечеткие правил); возможность классификации угроз; прозрачность для анализа структуры; функциональная устойчивость и защищенность элементной базы [3; 13].

Модели защиты являются важной частью системы управления рисками. Учитываются параметры: актуальные угрозы, имеющиеся ошибки в ПО, важность, интервал и время простоя различных ресурсов, вероятность атаки, варианты защиты и возможная величина ущерба. Система управления рисками в системе ИС позволяет просчитывать существующие риски, моделировать комплекс контрмер, автоматически разрабатывать профиль защиты, оценивать остаточные риски [3; 13].

Разрабатываемые интеллектуальные средства выявления атак и несанкционированных информационных процессов в ИС уделяют внимание свойству адаптивности при построении перспективных систем, причем системы уровня почтовых шлюзов и межсетевых экранов ориентированы на выявление внешних атак, а системы серверного уровня – на нейтрализацию внутренних угроз в корпоративной системе. Известные ИСЗИ реализуют лишь механизмы оперативной реакции и нейтрализации угроз жизнедеятельности системы ИС, при этом не уделяя внимание координирующей роли, которую играет нервная система – верхний уровень иерархии защиты биологических систем в реализации эволюционного процесса накопления жизненного опыта системы. Новым направлением в ИСЗИ является использование интеллектуальных агентов (ИА), работающих в распределенной ИС и запрограммированных на поиск как вторжения, так и аномалий. Выделены следующие области использования ИА в защите информации: автоматизация проведения исследований по нарушению защиты (ИНЗ); автоматизация поиска по ЗИ (организаций, технологий, услуг и т. д.); интеллектуализация принятия решений по ЗИ [3; 13].

8.9 Открытые семантические технологии

В работе [7] подводится 5-летний итог развития проекта OSTIS (Open Sematic Technology for Intelligent Systems), целями которого являются:

− разработка технологии OSTIS, позволяющей быстро и качественно разрабатывать семантически совместимые компьютерные системы, управляемые знаниями, и способные создавать временные коллективы ИС для распределенного решения сложных задач;

− определение круга актуальных приложений таких систем;

− использование технологии OSTIS как основы практико-ориентированной подготовки студентов и магистрантов;

Под семантической технологией компонентного проектирования ИС понимается комплекс технологий, обеспечивающий целостное проектирование ИС, управляемых знаниями (КСУЗ), и включающий [8]:

− семантическую технологию компонентного проектирования баз знаний и программ, ориентированных на обработку баз знаний;

− семантическую технологию компонентного проектирования решателей задач в ИСах, управляемых знаниями;

− семантическую технологию компонентного проектирования пользовательских интерфейсов, обеспечивающих общение КСУЗ с пользователями;

− семантическую технологию компонентного проектирования подсистем мультисенсорного восприятия и анализа информации среде КСУЗ;

− семантическую технологию компонентного проектирования подсистем координируемого воздействия на внешнюю среду КСУЗ.

Компьютерная система, управляемая знаниями, – система, в основе которой лежит представленная унифицированным образом база знаний, содержащая в систематизированном виде всю информацию, используемую этой системой. В КСУЗ вся используемая и обрабатываемая информация представляется в виде семантически структурированной целостной базы знаний, отражающей на смысловом уровне картину Мира, в котором работает эта система. Решатель задач компьютерной системы, управляемой знаниями, представляет собой коллектив самостоятельных агентов, взаимодействующих между собой только через базу знаний (БЗ). Некоторые агенты могут взаимодействовать и с внешней средой с помощью рецепторных средств. Указанный характер взаимодействия рассматриваемых агентов позволяет трактовать КСУЗ как систему, управляемую своей БЗ. Память КСУЗ носит структурно перестраиваемый ассоциативный характер и в перспективе может быть реализована не только программно, но и аппаратно. Для реализации обработки знаний в такой памяти, для описания поведения агентов необходимы языки программирования, ориентированные на обработку знаний в структурно перестраиваемой ассоциативной памяти. При переходе на любую новую платформу реализации структурно перестраиваемой памяти и интерпретатора соответствующего базового языка программирования никаких изменений в БЗ и в решатель задач конкретной КСУЗ вносить не потребуется [8].

Не следует отождествлять понятие КСУЗ и понятие системы, основанной на знаниях. Понятие системы, основанной на знаниях, является более общим, т. к. БЗ не обязательно должна быть активной и семантически структурированной. Это означает, что управление в системе, основанной на знаниях, может осуществляться программой, а не ситуациями и событиями в обрабатываемой БЗ. Это означает, что обработка информации в системе, основанной на знаниях, не обязательно осуществляется коллективом агентов, работающих над БЗ и взаимодействующих через нее [8].

Переход от традиционного построения ИС к системам, построенным по архитектуре КСУЗ, представляет собой переход к иным методологическим принципам построения ИС. В основе традиционного подхода к построению ИС лежит доминирование программ, в результате чего структуризация обрабатываемых данных подстраивается под программы [8].

Основой построения КСУЗ является доминирование структуризации обрабатываемой информации над программами. При этом структуризация обрабатываемой информации осуществляется независимо от программ и определяется исключительно смыслом этой информации. Для того чтобы в ИС обеспечить управление знаниями, в состав БЗ должны входить [8]:

− формулировки задач, решаемых компьютерной системой;

− планы решения этих задач, представляющие собой описание различных вариантов сведения решаемых задач к иерархической системе подзадач;

− указание активных задач, решаемых в текущий момент (как атомарных задач, так и задач, сводимых к подзадачам).

В результате трансформации современных ИС в функционально эквивалентные КСУЗ получены различные, но семантически совместимые системы, отличающиеся структурной сложностью БЗ, функциональной сложностью агентов обработки знаний [8].

Актуальность КСУЗ обусловлена тем, что все большую важность приобретают задачи, для решения которых априори трудно локализовать информацию, достаточную для их решения. В ходе решения таких задач может потребоваться заранее непредсказуемая информация, хранимая в памяти системы, удовлетворяющая тем или иным требованиям. Примерами указанных задач являются: информационно-поисковые; интеграции информации; задачи, для решения которых априори не известны способы; анализа качества хранимой в памяти информации; логического вывода.

Для того чтобы специализированные решатели интегрировать в состав КСУЗ, достаточно организовать обмен конечного количества исходных и возвращаемых параметров с априори известной их семантической интерпретацией или обмен конечных информационных конструкций, представленных в оговоренном формате. К сожалению, для задач первого вида такой принцип взаимодействий компьютерной системы с решателями задач невозможен. Обмен информацией между решателями задач первого вида возможен только через общую для них память на основе унифицированного представления обрабатываемой ими информации. Чтобы быстро можно было найти информацию, востребованную в ходе решения задачи, вся хранимая в система информация должна быть структурирована, систематизирована, т. е. преобразована в целостный информационный контент [8].

Актуальность трансформации традиционных ИС в КСУЗ обусловлена также тем, что переход на унифицированное представление различного вида знаний создает реальные условия для устранения дублирования информации в различных системах. Это, в свою очередь, дает возможность унифицировать и средства обработки знаний. Методика компонентного проектирования, основанная на постоянно расширяемых библиотеках компонентов, для КСУЗ более широко применима, чем для ИС, по причинам [8]:

− унификация логико-семантической архитектуры КСУЗ и унификация представления знаний дает возможность расширить библиотеки многократно используемых (reusable) компонентов. Проектирование ИС становится более «крупноблочным» и более быстрым;

– агенты, работающие над БЗ ИС, управляемых знаниями, часто имеют предметно независимый характер, что делает их многократно используемыми компонентами в различных КСУЗ;

– КСУЗ могут интегрировать новые компоненты, вводимые в состав расширяемых библиотек многократно используемых компонентов. Каждая КСУЗ может эволюционировать в ходе своей эксплуатации как в результате деятельности своих разработчиков, так и на основе библиотек.

Качество каждой компьютерной системы можно оценивать в трех аспектах: как уровень приобретенных ею знаний и навыков; как скорость повышения уровня приобретаемых знаний и навыков; как набор и степень ограничений, накладываемых на уровень знаний и навыков [8].

Можно говорить о классификации КСУЗ по двум признакам: уровень интеллектуальности средств обработки знаний и уровень развития рецепторно-эффекторных средств взаимодействия с внешней средой. По первому признаку можно выделить: КСУЗ, не имеющие интеллектуальных решателей задач; интеллектуальные системы, управляемые знаниями. По второму признаку классификации выделим: КСУЗ, не имеющие развитых рецепторно-эффекторных средств взаимодействия с внешней средой; робототехнические компьютерные системы, управляемые знаниями [8].

Принципы внутреннего представления знаний в КСУЗ. В разработке КСУЗ ключевую роль играет способ внутреннего представления знаний. Качество представления знаний компьютерной системы определяется тем, насколько приблизилось это представление к смысловому (семантическому) представлению. Подход ОСТИС к формализации смысла представляемых знаний основан на следующих положениях [8].

Все известные языки выполняют две функции – коммуникативную (средство обмена сообщениями между субъектами) и когнитивную (как средство представления информационной модели описываемого Мира). Язык внутреннего представления знаний в памяти компьютерной системы не обязан выполнять коммуникативную функцию. От языка внутреннего представления знаний требуется только то, чтобы он обеспечил хранение знаний в удобном для их обработки виде. Язык внутреннего представления знаний должен выполнять только когнитивную функцию – быть средством представления внутренней информационной модели описываемого Мира [8].

В рамках внутреннего смыслового представления БЗ компьютерной системы исключается дублирование знаков. Внутренние знаки, обозначающие одну и ту же сущность, должны быть склеены. В рамках каждой БЗ исключается семантическая эквивалентность (дублирование) входящих в ее состав фрагментов, несущих одну и ту же информацию.

Тексты языка внутреннего смыслового представления знаний должны быть нелинейными в отличие от привычных текстов, т. к. нелинейные тексты способны более адекватно отразить описываемый ими Мир, который по своей природе нелинеен.

В рамках языка внутреннего смыслового представления знаний вводятся мощные средства перехода от информации к метаинформации (в частности, от слабо структурированных данных к связанным данным). Язык внутреннего смыслового представления знаний должен быть универсальным, т. е. должен обеспечить представление всевозможного вида знаний: спецификаций самых различных сущностей; документаций различных технических систем и предметных областей; различного вида онтологий предметных областей; текстов высказываний; текстов доказательства теорем; формулировок задач и классов задач; текстов и способов решений конкретных задач и классов [8].

Способ внутреннего представления знаний, используемый в технологии OSTIS, назван SC-кодом (Semantic Code). Знаки, входящие в состав текстов SC-кода, называются sc-элементами. Каждый sc-элемент можно считать инвариантом всего многообразия форм представления (во всевозможных языках и знаковых конструкциях) той сущности, которая обозначается этим sc-элементом. Таким инвариантом является только то, что указанный sc-элемент обозначает соответствующую ему сущность. Поэтому sc-элемент не имеет формы. В этом смысле он абстрагируется от формы своего представления в рамках той или иной знаковой конструкции. Каждому sc-элементу взаимно однозначно соответствует некоторый класс синонимичных знаков – множество всевозможных вхождений знаков обозначаемой сущности в самые различные внешние знаковые конструкции (тексты). Текст SC-кода (sc-текст) с синтаксической точки зрения – это абстрактная знаковая конструкция, представляющая собой множество sc-элементов, связанных между собой отношениями инцидентности. С семантической точки зрения sc-текст – это абстрактная знаковая конструкция, являющаяся инвариантом всего многообразия форм представления информации. Каждому sc-тексту взаимно однозначно соответствует класс всевозможных семантически эквивалентных внешних текстов, представляющих заданную информацию в самых различных языках [8].

Положения технологии, ориентированной на проектирование КСУЗ. Одним из главных направлений развития ИИ является создание общей, доступной и эффективной технологии комплексного проектирования КСУЗ. К принципам, лежащим в основе эффективной технологии проектирования, относится: отделение этапа проектирования от этапа производства; унификация результатов проектирования (для ИС – платформенная независимость логического проектирования); компонентное (модульное) проектирование, на основе пополняемых библиотек компонентов. Технология должна быть доступной, открытой, в соответствии с этим строится технологии OSTIS, ориентированной на проектирование КСУЗ [7; 8].

Базовая графодинамическая модель обработки информации. В качестве формальной основы проектируемых КСУЗ предлагается использовать графодинамические модели (ГДМ) обработки информации специального вида, ориентированные на параллельную асинхронную обработку. ГДМ обработки информации трактует процесс преобразования графовой структуры, в ходе которого меняется не только состояние элементов этой графовой структуры, но и

конфигурация этой структуры. Для создания ГДМ обработки информации недостаточно тех видов графовых структур, которые исследуются в теории графов. Общее определение графовой структуры, на основе которого можно строить ГДМ обработки информации, приведено в работе [8].

Без организации параллельной обработки информации невозможно рассчитывать на необходимую производительность интеллектуальных систем. ГДМ параллельной асинхронной обработки информации трактуется как абстрактная многоагентная система, состоящая из: абстрактной графодинамической памяти, в которой хранятся обрабатываемые графовые структуры; коллектива агентов, работающих над этой памятью и обменивающихся информацией только через нее. Графодинамическая память носит реконфигурируемый, структурно перестраиваемый характер, поскольку процесс обработки графовых структур сводится к генерации и удалению различных элементов графовых структур, а также к генерации и удалению пар инцидентности между этими элементами [8].

Агенты, работающие над общей графодинамической памятью, делятся на три вида: внутренние агенты, каждый из которых реагирует на определенного вида ситуации или события в графодинамической памяти и осуществляет изменение ее состояния, соответствующее своему функциональному назначению; рецепторные агенты, каждый из которых реагирует на определенные события во внешней среде и осуществляет отражение этих событий в графодинамической памяти; эффекторные агенты, каждый из которых реагирует на определенного вида команды, формируемые внутренними агентами в графодинамической памяти, и осуществляет соответствующее изменение материального состояния интеллектуальной системы, которое определенным образом влияет на изменение ее внешней среды. Агенты могут работать параллельно, если одновременно возникают условия инициирования агентов. Асинхронность деятельности внутренних агентов заключается в том, что наличие условия инициирования агента еще не означает начала его работы [8].

Унификация семантических моделей обработки знаний. В основе предлагаемого подхода к созданию технологии проектирования КСУЗ лежит принцип унификации семантических моделей интеграции знаний и семантических моделей интеграции целых систем, управляемых знаниями. На основе унифицированных семантических сетей необходимо обеспечить построение унифицированных семантических моделей интеграции знаний и использовать эти модели, во-первых, как основу процесса приобретения ИС новых знаний как со стороны конечных пользователей, так и со стороны разработчиков; во-вторых, как основу интеграции программ и различных семантических моделей расширения задач; в-третьих, как основу интеграции абстрактных логико-семантических моделей ИС [8].

Унифицированный способ кодирования различных видов знаний необходим для обеспечения совместимости различных видов знаний и различных языков представления знаний, что необходимо для интеграции знаний [9], а также для интеграции языков представления знаний. Без приведения интегрируемых знаний к общей, унифицированной форме интеграция невозможна. Для семан-

тических моделей обработки знаний предполагаются виды унификации: структуризации БЗ и типологии знаний; моделей информационного поиска; моделей интеграции знаний; моделей решения задач; моделей пользовательских интерфейсов; визуализации БЗ [8].

Компонентное проектирование ИС. Проектируемые КСУЗ обладают высоким уровнем гибкости, их разработка в технологии ОСТИС осуществляется поэтапно, переходя от одной целостной версии системы к другой. При этом стартовая версия системы может быть ядром соответствующего класса систем, входящим в библиотеку многократно используемых компонентов [8].

Предлагаемая технология концентрирует внимание не на создании стартовых версий разрабатываемых систем (это превращается в достаточно простую их сборку из имеющихся компонентов), а на расширении и совершенствовании текущих версий разрабатываемых систем в ходе их эксплуатации. Проектирование по предлагаемой технологии имеет компонентный характер. Организация такого сборочного модульного проектирования требует создания, пополнения и сопровождения библиотеки типовых ip-компонентов (компонентов интеллектуальной собственности – Intellectual Property Core). Библиотека ip-компонентов – это систематизированный каталог комплектации для сборочного проектирования ИС, формальное описание коллективного опыта всех разработчиков таких систем [8].

Доступность и открытость технологии. В предлагаемой технологии осуществляется постоянное накопление и систематизация накапливаемого проектного опыта по разработке различных классов и различных видов КСУЗ самими разработчиками. Конструктивным результатом этого процесса является расширение разделов библиотеки компонентов. Доступность предлагаемой технологии обеспечивается тем, что документация по технологии оформляется как раздел БЗ метасистемы, которая может выполнять функцию справочной ИС по предлагаемой технологии. От гибкости (модифицируемости) проектируемых КСУЗ переходим к гибкости самой технологии их проектирования, что является достоинством предлагаемой технологии [8].

Принципы организации проекта OSTIS. Проект OSTIS, направленный на создание и применение семантической технологии компонентного проектирования КСУЗ, имеет следующие особенности: технология является универсальной, т. к. любую ИС можно построить на ее основе; новая парадигма построения ИС, в которой доминирующую роль играют не программы, а обрабатываемая информация, имеющая в памяти системы онтологически унифицированное смысловое представление, основанное на согласованной системе понятий. Хранимая в памяти информация включает в себя как обрабатываемую часть, так и часть, обеспечивающую управление процессом обработки, т. е. управление агентами, работающими над памятью [8].

Можно выделить три уровня способностей при подготовки молодых специалистов в области ИС: (1) выполнять четко поставленные и не простые индивидуальные проектные задания; (2) эффективно участвовать в коллективных решениях нечетко поставленных проектных задач; (3) эффективно участвовать в совершенствовании используемой технологии [8].

Актуальные направления применения КСУЗ. Любую ИС можно построить по архитектуре КСУЗ, что даже без интеллектуализации решателей задач переведет эти системы на качественно новый уровень, позволяющий существенно увеличить темпы их совершенствования в ходе эксплуатации и обеспечить их полную семантическую совместимость. Есть целый ряд областей применения КСУЗ, в которых качество, структуризация и легкая модифицируемость контента играет решающую роль: мир вещей, умный дом, интеллектуальные робототехнические системы, системы ситуационного и интеллектуального управления, мир Интернета, включая интеллектуальных агентов: веб-сайты с семантически структурированным контентом, справочные ИС учебного назначения [8].

Актуальным является создание комплекса семантически совместимых справочных ИС по школьным дисциплинам и по общеобразовательным дисциплинам технических вузов (теория множеств, теория графов, теория отношений, абстрактная алгебра, математическая логика, теория алгоритмов и программ и др.), а также интеллектуальные help-системы; порталы научных знаний, обеспечивающие постоянно совершенствуемую систематизацию и структуризацию научных знаний в соответствующих областях; семантически структурированный аналог Википедии, которая может сохранить все существующие естественно-языковые статьи, сделав над ними семантически структурированную надстройку путем частичного или полного перевода этих статей на внутренний язык; семантически структурированные электронные научные журналы, актуализируемые соответствующими редакционными коллегиями, и автоматизирующие решение таких трудоемких задач, как рецензирование (в т. ч. и верификация доказательств) различных научных статей, которые должны быть представлены в формализованном виде; интеллектуальные корпоративные системы, управляемые знаниями; интеллектуальные системы автоматизации проектирования различных классов технических систем, интеллектуальные обучающие системы, включающие в себя соответствующие интеллектуальные справочные системы и имеющие возможности управлять деятельностью обучаемых, учитывая их индивидуальные особенности [8].

Выводы

На основании приведенного материала и с учетом источников [1–20].

1. Используется модель машинного перевода, включающая анализ и синтез естественно-языковых сообщений, которая предназначена для анализа текста: морфологический анализ – анализ слов в тексте; синтаксический анализ предложений, грамматики и связей между словами; семантический анализ – анализ смысла каждого предложения на основе предметно-ориентированной БЗ; прагматический анализ – анализ смысла предложений в окружающем контексте.

2. Методы синтеза программ используют два подхода: либо генерируют программу на машинном языке, либо составляют резидентный набор из заранее отлаженных модулей на время выполнения задачи. В основе синтеза объектов

могут быть использованы три подхода: дедуктивный, индуктивный и трансформаторный. Первый заключается в построении объекта на основе доказательства утверждения, что решение задачи существует. Во втором синтез основывается на использовании частных примеров, задающих ответ для отдельных исходных данных. В третьем искомый объект получается постепенно путем преобразования исходного описания по правилам, совокупность которых представляет знания о решаемой задаче.

3. В связи с появлением и развитием Интернета можно выделить такие направления, как представление знаний в Сети, извлечение знаний из информационных ресурсов Сети, интеллектуальный поиск релевантной информации, интеграция информационных ресурсов и приложений управление знаниями, построение глобальных систем знаний в Интернете и их обработка (семантический веб), построение и использование веб-сервисов.

4. Различные поисковые системы используют различные алгоритмы ранжирования, однако основные принципы определения релевантности следующие: количество слов запроса в текстовом содержимом документа (т. е. в html-коде); теги, в которых эти слова располагаются; местоположение искомых слов в документе; удельный вес слов, относительно которых определяется релевантность в общем количестве слов документа.

5. Метод LSA (Latent Semantic Analysis), опирающийся на векторное представление документов, может быть эффективно применен для распознавания и классификации веб-документов. В его основе лежит статистические методы анализа частоты слов. Его вычислительная емкость растет почти линейно при повышении числа обрабатываемых документов.

6. Каждая поисковая машина обладает рядом особенностей: тип машины, ее размер, период обновления страниц, указанные и не указанные страницы, глубина индексирования, поддержка фреймов, частота обновления ссылок, способность к обучению, контроль индексации, перенаправление, учет стоповых слов, алгоритмы релевантности, поддержка метатегов, удаление старых данных.

7. На март 2024 распределение по использованию поисковых машин следующее: Google – 93,37 %; Bing – 3,37 %; Yandex – 1, 64 %; Baidi –1,15 %; Yahoo – 1,13 %. Несмотря на мировой топ, в разных странах востребованы разные поисковые системы. Например, в Китае на первом месте Baidu, а в США лидируют Google и Bing, в России – Yandex.

8. Процесс распознавания основан на сопоставлении признаков, характеристик исследуемого объекта с признаками, характеристиками других известных объектов, в результате чего делается вывод о наиболее правдоподобном их соответствии. Методы распознавания образов используют в технической и медицинской диагностике. В последнее время широко используются в различных интеллектуальных системах безопасности, системах интеллектуальных зданий и т. д.

9. Вместе с традиционными средствами защиты ИС: антивирусы, детекторы уязвимостей, межсетевые экраны и детекторы вторжений применяются средства автоматизации защиты, включающие корреляторы событий, программы обновлений, средства аутентификации, авторизации и администрирования,

системы управления рисками. ИС ЗИ посвящены системам обнаружения атак, в качестве интеллектуального инструмента в которых используются НС, системы нечеткой логики и основанные на правилах ЭС.

10. В ИС ЗИ экспертные системы в базе содержат описание классификационных правил, соответствующих профилям легальных пользователей и сценариям атак на ИС. Если НС представлена в виде системы обнаружения атак, при обработке трафика происходит анализ информации на наличие злоупотреблений. В системах обнаружения атак применяются НС, дополненные ЭС.

11. Свойства нечетких НС, необходимые для адаптивных средств защиты информации: распределенный параллелизм вычислений; адаптивность нейронечетких средств защиты информации (системы нечетких правил); возможность классификации угроз; «прозрачность» для анализа структуры; функциональная устойчивость и защищенность элементной базы. Направлением в ИСЗИ является использование интеллектуальных агентов, работающих в распределенной ИС и запрограммированных на поиск как вторжения, так и аномалий.

12. Опыт создания комплексной технологии OSTIS и опыт разработки прикладных систем на ее основе создает необходимые предпосылки для начала перехода от традиционных компьютерных систем к КСУЗ, в которых не всегда требуется умение решать интеллектуальные задачи. Процесс развития технологий проектирования КСУЗ должен быть перманентным, а основой его должно быть активное сотрудничество науки, образования, инженерии и бизнеса [8].

Контрольные вопросы

1. Поясните суть машинного перевода.
2. Охарактеризуйте структуры ИС-переводчика.
3. Объясните особенность синхронного перевода.
4. Поясните суть методов синтеза программ.
5. Расскажите про три подхода к синтезу объектов.
6. Поясните направления представления знаний в Интернете.
7. Дайте суть принципа определения релевантности.
8. Охарактеризуйте метод LSA.
9. Охарактеризуйте особенности поисковых машин.
10. Поясните два вида баз данных о веб-страницах.
11. В чем суть метода распознавания образов.
12. Поясните математическую основу распознавания образов.
13. Поясните интеллектуальные инструменты для защиты информации.
14. Поясните новое направление в ИСЗИ.
15. Поясните основные положения ОСТИС.
16. Определите особенности КСУЗ.
17. Сформулируйте применение КСУЗ.

Литература, используемая в главе

1. Системы машинного перевода [Электронный ресурс]. – Режим доступа: https://knowledge.allbest.ru/programming/3c0a65625b2ad69b4d43b88521216c26_0.html. – Дата доступа: 15.12.2022.

2. Рассел, С. Искусственный интеллект, современный подход / С. Рассел, П. Норвинг. – М. : Вильямс, 2006. – 1408 с.

3. Вишняков, В. А. Информационное управление и безопасность: методы, модели, программно-аппаратные решения / В. А. Вишняков. – Минск : Изд-во МИУ, 2014. – 287с.

4. Indexing by latent semantic analysis [Электронный ресурс]. – Режим доступа: http://www.cs.bham.ac.uk/~pxt/IDA/lsa_ind.pdf#1. – Дата доступа: 15.12.2022.

5. Vishniakov, V. A. Models and Tools of Logical Inference Systems / V. A Vishniakov, O. V. German. – Moscow : Nauka, 1999. – 374 p.

6. Голенков, В. В. Семантическая технология компонентного проектирования систем, управляемых знаниями / В. В. Голенков, Н. А. Гулякина // ОСТИС-15 : материалы 5-й науч.-техн. конф. – Минск, БГУИР, 2015. – С. 57-78.

7. Представление и обработка знаний в графодинамических ассоциативных машинах / В. В. Голенков [и др.]. – Минск : Изд-во БГУИР, 2001. – 421 с.

8. Известные поисковые машины. Поисковые машины и каталоги [Электронный ресурс]. – Режим доступа: https://www.nexxdigital.ru/izvestnye-poiskovye-mashiny-poiskovye-mashiny-i-katalogi-poisk/. – Дата доступа: 15.12.2022.

9. Информатика : учебник для эконом. специальностей вузов / под ред. Н. В. Макаровой. – М. : Финансы и статистика, 1999. – 440 с.

10. Белов, В. В. Повышение пертинентности поиска в современных информационных средах / В. В. Белов, А. А. Терехов, В. И. Чистякова – М. : Горячая линия, 2012. – 158 с.

11. Метаданные [Электронный ресурс]. – Режим доступа: http://www.citforum.ru/ internet/search/tips.shtml. – Дата доступа: 15.12.2022.

12. Интеллектуальные средства и моделирование систем [Электронный ресурс]. – Режим доступа: http://elibrary.ru/item.asp?id=20299819. – Дата доступа: 15.12.2022.

13. Студопедия [Электронный ресурс]. – Режим доступа: https://studopedia.org/8-240447.html. – Дата доступа: 15.12.2022.

14. Старовойтов, В. В. Цифровые изображения: от получения до обработки / В. В. Старовойтов, Ю. И. Голуб. – Минск : Изд-во ОИПИ НАН Беларуси, 2014. – 202 с.

15. Какие поисковики самые популярные в интернете: сравнение регионов [Электронный ресурс]. – Режим доступа: https://serpstat.com/ru/blog/kakie-poiskoviki-samye-populjarnye-v-internete/. – Дата доступа: 15.03.2024.

16. Самые популярные поисковые системы в мире [Электронный ресурс]. – Режим доступа: https://makeagency.ru/blog/samye-populyarnye-poiskovye-sistemy-v-mire. – Дата доступа: 15.03.2024.

17. Популярные поисковики и браузеры в Белоруссии [Электронный ресурс]. – Режим доступа: https://etistudio.by/blog/populyarnye-poiskovye-sistemy-i-brauzery-vo-1-polugodii-2022-goda-v-belarusi#. – Дата доступа: 15.03.2024.

18. Создаем личный «Архив интернета» [Электронный ресурс]. – Режим доступа: https://habr.com/ru/companies/first/articles/584838/. – Дата доступа: 15.05.2024.

19. Латентно-семантический анализ [Электронный ресурс]. – Режим доступа: https://habr.com/ru/articles/110078/. – Дата доступа: 15.03.2024

9 ИНТЕЛЛЕКТУАЛИЗАЦИЯ ДЛЯ ПРИНЯТИЯ РЕШЕНИЙ В ИКС

9.1 Управление обслуживанием вызовов/сессий

Изменения в архитектуре и технологиях современных ИК сетей наряду с тремя известными направлениями (мобильность, мультисервисность, конвергенция сетей и услуг) включают [2; 3]:

– распределенное управление обслуживанием вызовов/сессий в сетевых элементах гетерогенных сетей нового поколения;

– усиление локального интеллекта этих сетевых элементов NGN в условиях сверхплотных сетей (с развитием IoT).

Область коммутации/маршрутизации и управления обслуживания вызовов/сессий в рамках трехуровневой концепции требует привлечения метода самоорганизации и мультиагентных моделей для управления этой областью.

На рисунке 9.1 представлены два подхода к управлению ИК [2; 3]:

– централизованное фиксированное планирование;

– децентрализованная динамическая самоорганизация.

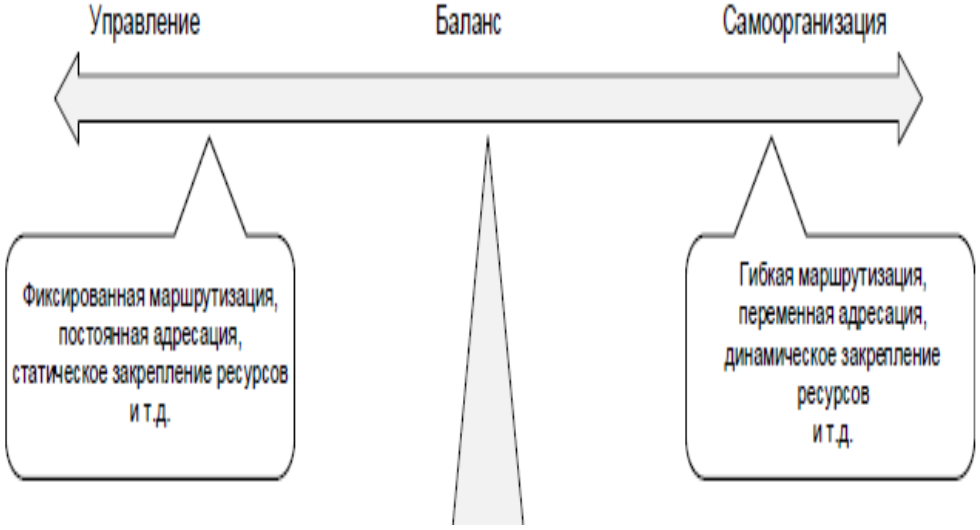

Рисунок 9.1 – Два подхода к управлению современными ИКС

Детерменированное управление заложено в модели OSI, где управляемые объекты идентифицированы в абстрактных выражениях как MO (managed objects). Управляющие и управляемые объекты, названные *менеджерами* и *агентами*, рассматриваются как одноранговые [2; 3].

Далее получил развитие протокол Simple Network Management Protocol (SMNP), который и определил архитектуры для объектов менеджера и агента, протокол связи, а также хорошо разработанную базу Management Information Base (MIB) с определениями для большинства типов элементов, находящихся в сетях IP.

ITU-T разработала тоже уже ставший традиционным подход к процедурам управления ИК, который соответствовал специфическим требованиям сетей связи и представлял собой комплект стандартов TMN [2; 3].

Главная идея и первый принцип управления TMN (рисунок 9.2) представлены пирамидой с пятью логическими уровнями [2; 3]:

– *Network Element Layer (NEL, уровень сетевого элемента)* – содержит часть сетевых элементов;

– *Element Management Layer (EML, уровень управления элементами)* – содержит агента управления для сетевых элементов;

– *Network Management Layer (NML, уровень управления сетью)* – связан с управлением всей сетью, состоит из ряда агентов сети;

– *Service Management Layer (SML, уровень управления службами)* – состоит из интерфейса заказчика, управления учетными записями, обеспечения обслуживания и обработки жалоб, никакое управление физическими объектами не имеет места;

– *Business Management Layer (BML, уровень управления экономической деятельностью)* – задачи планирования, соглашений между операторами, установки, реализации и отслеживания.

Вторым принципом *управления* TMN (рисунок 9.2) являются *функциональные области*, заключенные в абревиатуре FCAPS (*Fault, Configuration, Accounting , Performance, Security management*).

К новым принципам отнесем управлением SLA (Service Level Agreement), управлением мобильностью и точностью позиционирования, управление энергетикой (для IoT, сенсорных сетей) [2; 3; 4].

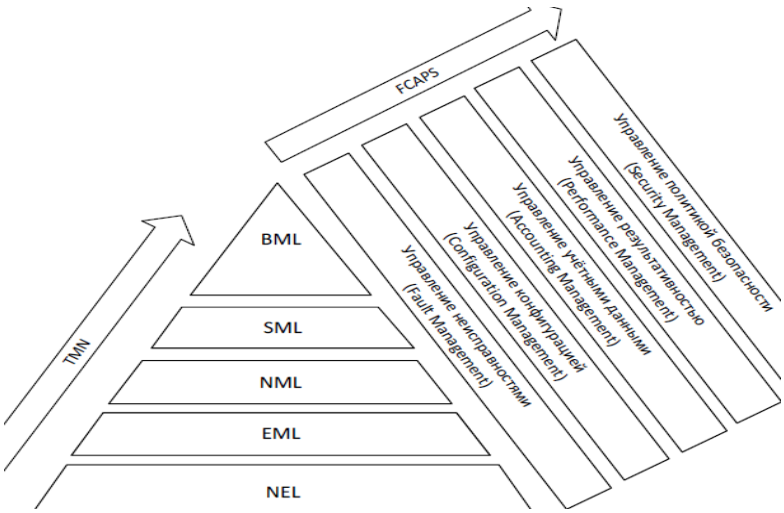

Рисунок 9.2 – Базовые принципы TMN

9.2 Мультиагентные модели управления в сетях NGN

МАС строится как совокупность рациональных агентов, объединенных в группы и взаимодействующих друг с другом для достижения общей цели эффективного управления ИК-сетью. Общение агентов друг с другом,

сотрудничество, координация и взаимодействие агентов в системе осуществляются помощью языка коммуникаций ACL (Agent Communication Language). Этим и ролевыми функциями агентов и нормами их взаимодействия определяется архитектура МАС [2; 3].

Дополним это определение свойствами ИА: *автономность* (агент функционирует без вмешательства со стороны); *способность общения* (взаимодействие и коммуникации с другими агентами); *реактивность* (восприятие среды и проявление соответствующей реакции на ее изменения); *базовые знания* (знания агента о себе, окружающей среде, включая других агентов). ИА представляется [2; 3]:

Интеллектуальный агент = данные + методы+ знания

Методы – это не только функции работы с данными, но и со знаниями и возможные методы воздействия на окружающую среду агентом.

В моделях МАС допускаются типы взаимодействия:

– горизонтальные связи – связи между равноправными агентами;

– вертикальные (субординационные) связи – связи подчинения; координация – согласованное действие агентов, подчиняющихся вышестоящему агенту-координатору.

Предусматриваются типы агентов: исполнители и менеджеры (первые подчиняются вторым); агенты-координаторы, ответственные за организацию взаимодействия агентов; интерфейсные агенты, служащие для связи с внешней средой; коммутационные агенты, обеспечивающие обмен информацией в системе. Типы агентов – реактивные и интеллектуальные.

Реактивные агенты имеют примитивную внутреннюю модель внешнего мира. Для них характерно использование концепции состояния и простейших механизмов поведения типа «стимул – реакция».

Интеллектуальные *агенты* отличаются от реактивных агентов наличием у них встроенной базы знаний и развитого механизма планирования действий, в связи с чем они часто называются **когнитивными.**

Когнитивные ИА обладают развитой и пополняемой символьной моделью ТК-сети, что достигается благодаря наличию у них базы знаний, механизмов решения и анализа управляющих воздействий.

Близкими к когнитивным являются так называемые ДА – делиберативные (deliberative) или рассуждающие агенты, которые на основе символьной модели внешней среды способны проводить собственные рассуждения, используя сравнения по образцам, и на их основе принимать самостоятельные решения или выполнять некоторые управляющие действия.

Расширим ситуационную модель поведения мультиагентных систем. Взаимодействие между агентами в МАС опишем на базе математической теории отношений. В ней классическое n-арное отношение определяется как подмножество декартова произведения произвольных n множеств [3]:

$$R \subseteq X_1 \times X_2 \times \ldots \times X_n \,. \qquad (9.1)$$

9.3 Математическая модель мультиагентной системы управления

Выражение (9.1) представим в виде нечеткого отношения [3; 4]:

$$R\colon X_1\times X_2\times\ldots\times X_n \to L, \tag{9.2}$$

где L – решетка.

Будем рассматривать бинарные отношения на множестве агентов A, в том числе обычные и нечеткие отношения:

$$R\subseteq A\times A, \tag{9.3}$$

$$\mathrm{MR}\ A\times A \ \to [0,1]. \tag{9.4}$$

Запись aRb означает, что агент a находится в отношении R с агентом b, а MR(a, b) понимается как степень интенсивности (сила) проявления этого отношения.

Пусть E – единичное (диагональное) отношение, т. е.

$$\forall a\ \text{b€a MR}\ (a, b) = \{1, \text{если a} = 1, 0 – \text{в противном случае}\}. \tag{9.5}$$

Рассмотрим более формализованную мультиагентную модель системы управления ИК, включающую средства BI (Business Intelligence), базирующиеся на системах учета сетевых ресурсов NRI (Network Resources Inventory), биллинговых системах следующего поколения и других системах OSS/BSS, общее число которых обозначим через N. Величина N определяет количество ИА {1, 2,...,N}, каждый из которых выполняет соответствующие задания [2].

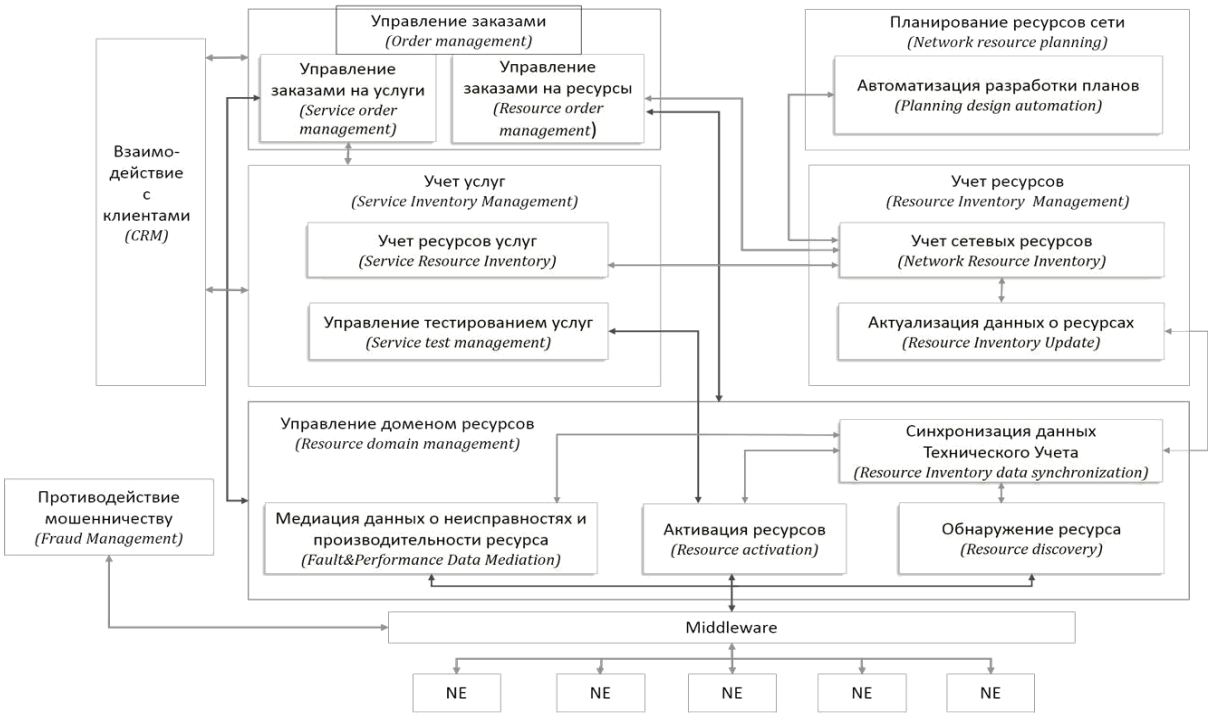

Рисунок 9.3 – Физическая модель управления ТК-сетью

Задания поступают в мультиагентную систему в различные моменты времени и на разные узлы. В каждый момент времени t состояние агента i {i = = 1, ..., n} описывается соответствующими вероятностно-временными характеристиками (ВВХ) [2].

Для решения задач управления ИК и, в частности, для организации поиска и обработки данных BI в распределенной ИТ-среде ТК оператора эффективным и целесообразным является использование мультиагентного подхода, в рамках которого система строится как совокупность агентов (агенты BI, агенты менеджера и агенты подсистем OSS/BSS).

Выбор мультиагентной технологии позволяет сочетать в единой системе как универсальные протоколы, так и другие средства работы с конкретными типами баз данных и подсистем управления ИК. Представим физическую модель управления ТК-сетью, которая показана на рисунке 9.3.

9.4 Мультиагентная модель сбора и обработки информации в системах управления ИК

Базовые операции BI (Business Intelligence) включают посылку запросов на данные в системы технического учета NRI (Network Resources Inventory), биллинга, технического обслуживания (Fulfillment), анализа производительности (Performance Management), безопасности, фрод-менеджемента FM (Fraud Management), контроля качества SQM (Service Quality Management) и др., а также сбор и интегральную обработку результатов этих запросов от N источников, приведенных на рисунке 9.4 [3].

По мере дальнейшего эволюционного развития сетей связи NGN и создания новых сервисов, сдвига трафика в сторону M2M, развития услуг, связанных с позиционированием абонентов, разнообразия сетевых элементов и внедрения принципов самоорганизации в построение сетей связи эти проблемы в обработке информации BI при управлении сверхсложными сетями пятого поколения становятся все более значимыми.

Информация для принятия решения D системой управления инфокоммуникациями состоит из N частей [2; 3]:

$$D = \{D_1, D_2, ..., D_N\}, \quad (9.6)$$

находящихся в N входящих в комплекс управления сетью системах ς (системах технического учета, биллинга, технического обслуживания, анализа производительности, фрод-менеджемента, контроля качества и т. д.). То есть имеем:

$$\varsigma = \{S_1, S_2, ..., S_N\}. \quad (9.7)$$

При этом имеется некоторое перекрытие данных, т. е.

$$D_i \cap D_j \neq 0. \quad (9.8)$$

На рисунке 9.4 представлена МАС, состоящая из N+1 агентов $M = \{M_1, M_2, ..., M_{N+1}\}$, соответствующая физической модели. В структуре каждый агент

реализует свою собственную стратегию Ω. Агенты имеют доступ к разным системам управления и возвращают информацию в ответ на те или иные запросы от BI. На запрос с индексом j агент M_i присылает данные $D_{ij}(+)$ из совокупности данных D_j.

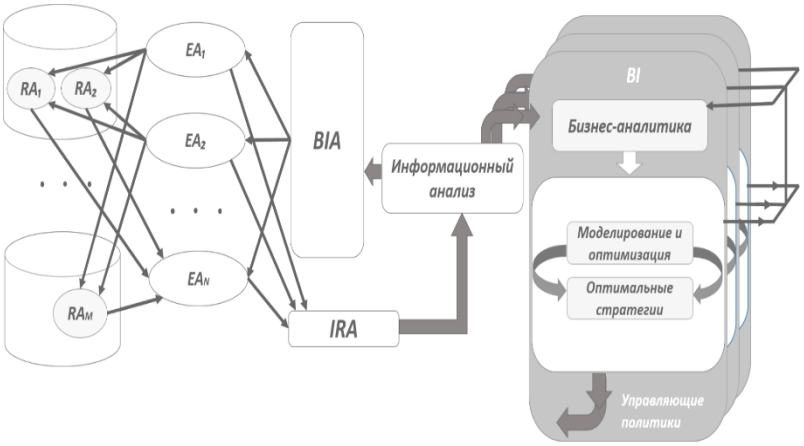

Рисунок 9.4 – Мультиагентная система

Затем присланные агентами данные объединяются по двум параметрам: данные, полученные из одной системы i, и данные из всех систем i =1, 2, …, N. Получаем выражение (9.9) [2]:

$$\Phi^{D_L}\left(\Omega_{N_i}(q_i, D_L), \ldots \Omega_{N_L}(q_L, D_L)\right) = \bigcup_{j=1}^{k}\left(\bigcap_{i=1}^{L} D_i^{I(+)}\right) = D^{+} \quad (9.9)$$

Для исследования МАС важны два параметра: *полнота* (ничего не забыли из нужного для BI) и *точность* (не прислали ничего лишнего). Эти параметры часто обозначают одним словом – *релевантность*. Релевантность – это соответствие ответа запросу, но с учетом введенных понятий: полнота и точность.

Полнотой называют отношение количества полученных результатов к общему количеству существующих в системе управления ИК данных, релевантных данному запросу (λзап – запросы с интенсивностью λ).

$$P = | D\,\lambda_3 \cap D+ | \,/\, D+. \quad (9.10)$$

Точность – это отношение количества релевантных результатов к общему количеству данных, ссылки на которые содержатся в ответе.

$$R = | D\,\lambda_3 \cap \mathrm{D}+ | \,/\, \mathrm{D}\lambda_3. \quad (9.11)$$

При оценке эффективности поиска информации используют и другие критерии – коэффициент потерь информации и коэффициент поискового шума. Считается, что коэффициент потерь информации = 0, а коэффициент поискового шума = 1. Точность может колебаться в диапазоне 0.1–1.0, полнота обычно лишь приближается к значениям 0.8 – 0.9.

Для обеспечения эффективности управляющих решений требуется не только точность и полнота данных от разных систем B/OSS, но и, в первую очередь, точность и полнота соответствующих значений интегральной функции $Ф^{DL}$. Для этого необходима *одновременная, кооперативная* обработка данных, добываемых агентами в разных системах.

9.5 Самоорганизация в управлении сетями 4/5G

Концепция самоорганизующихся сетей SON была создана некоммерческой организацией 3rd Generation Partnership Project (3GPP). Цель – повышение эффективности и гибкости управления ТК-сетями, сокращение затрат и увеличение прибыли Оператора, была ориентирована на применение в сетях мобильной связи поколений 3/4/5G. Современная и перспективная система управления сетью должна полностью поддерживать функциональность SON, как определено в стандартах 3GPP [2–4].

Параллельно эта концепция развивается в разработках некоммерческой организацией TM Forum – лидером «де-факто» в вопросах управления и автоматизации предприятий связи. Развиваемые TM Forum идеи связывают разработчиков ПО, вендоров ТК-оборудования, Операторов связи и справедливы применительно к управлению любыми сетями – как мобильной, так и фиксированной связи. Они и определили современный подход к автоматизации процессов эксплуатации сетей связи, предполагающий наличие в IT-структуре Оператора OSS-комплекса [2].

При эксплуатации сетей OSS-комплекс должен включать систему ТУ – Технического учета (Network Resource Inventory System, NRI), являющуюся фундаментом для работы с физическими и логическими ресурсами, и системы взаимодействия с оборудованием с целью управления сетевыми объектами.

Такой обмен данными организуется с применением промежуточного программного обеспечения (ППО), или Middleware, путем использования стандартных интерфейсов и протоколов MTNM, SNMP и др. и/или специально разработанных командных кодеков.

Системы управления сетью NMS, помогающие осуществлять [2; 3]:

– исследование сети (Network device discovery) с целью обнаружения новых элементов сети;

– мониторинг сетевых устройств (Network device monitoring) на предмет работоспособности устройств и соответствия SLA;

– анализ производительности сети, отслеживая ключевые показатели, такие как время безотказной работы оборудования, задержки и потери пакетов и т. д.

– настройку интеллектуальных уведомлений – настраиваемых оповещений, которые будут реагировать по конкретным сценариям сети по электронной почте или телефону.

Основными мотивами внедрения архитектуры SON является естественное желание Операторов сокращать операционные расходы за счет уменьшения

степени человеческого вмешательства на этапе планирования, внедрения и эксплуатации своих сетей, снижать капитальные затраты за счет оптимизации использования своих сетевых ресурсов, сохранять и наращивать прибыль, сокращая ошибки из-за человеческого фактора.

При быстром росте мобильной широкополосной передачи данных (ШПД) увеличивается спрос на услуги обмена данными, инициируя развитие технологий на радиоучастке сети. Независимые аналитики указывают на то, что объемы эксплуатационных задач, их качество и скорость их выполнения будут в ближайшем будущем отличаться от существующих сейчас.

В первую очередь это касается процессов планирования и развертывания сети Оператора при ее расширении, обеспечения требуемого качества обслуживания и бесперебойной работы всей инфраструктуры сети.

Существенную роль в достижении этих целей играет эксплуатационное управление OAM&P, состоящее из начальных букв: объединнение управления ресурсами (Operations), административное управление (Administration), техническое обслуживание (Maintenance) и ввод новых ресурсов (Provisioning) [3].

Эти задачи контролируются персоналом, даже со средствами автоматизации данный ручной труд требует большого количества времени, значительных финансовых затрат, высококлассных специалистов, но человеческие ошибки не исключены. Поэтому идеи и принципы, заложенные в концепцию самоорганизующихся сетей, начинают чаще находить отражение и применение в реалиях существующих сетей операторов.

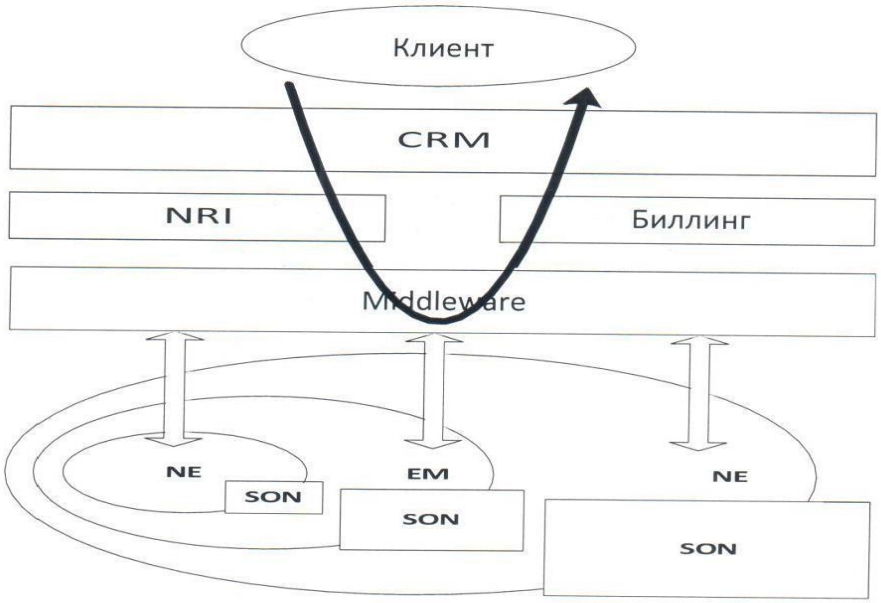

Рисунок 9.5 – Сквозное «прохождение» бизнес-процесса через системы

Для обеспечения связанных бизнес-процессов ИКС оператора необходимо взаимодействие OSS/BSS систем с сетевыми элементами и системой управления сетью. Системы Middleware являются промежуточным ПО между оборудованием и высокоуровневыми приложениями класса

OSS/BSS. Системы Middleware являются уровнем – посредником между функциями SON, то есть уровнем сети, и высокоуровневыми приложениями, такими как Inventory, CRM, Billing и др., как это представлено на рисунке 9.5 [2].

9.6 Сравнительный анализ альтернатив в условиях неопределенности условий реализации

Условия реализации представлены набором возможных исходов случайной величины [5]:

$$\boldsymbol{Y} = (Y_1, \dots, Y_\alpha \dots, Y_A).$$

Критерий Лапласа: предполагается, что вероятности исходов $Y_1, \dots, Y_\alpha \dots, Y_A$ равновероятны.

$$L = \max_I (1/A \sum E_i(Y_\alpha), \alpha = 1,\dots, A, i = 1, \dots, n.$$

Для примера используем данные в таблице 9.1.

Таблица 9.1 – Матрица выигрышей

Альтернативы	*Y*1	*Y2*	*Y*3
D	0,5	0,6	0,9
B	0,9	0,7	0,8
C	0,6	0,8	0,7

$$L = \max_{D,B,C} (1/3\ (0,5+0,6+0,9);\ 1/3\ (0,9+0,7+0,8);\ 1/3\ (0,6+0,8+0,7)) \rightarrow S_B.$$

Критерий Вальда (критерий осторожного наблюдателя): решение выбирается в расчете на наихудшие внешние условия, т. е. выбирается вариант с максимальным выигрышем в наихудших условиях среды:

$$W = \max_i \min_\alpha E_i (Y_\alpha),\ i = 1, \dots, n,\ \alpha = 1, \dots, A.$$

Для данных из таблицы 9.1 вычислим:

$$W = \max_{D,B,C} (0,5;\ 0,7;\ 0,6) \rightarrow S_B.$$

Критерий Сэвиджа: решение принимается в расчете на наихудшие внешние условия с использованием матрицы рисков:

$$W = \min_i \max_\alpha R_i (Y\alpha),\ i = 1, \dots, n,\ \alpha = 1, \dots, A.$$

Для примера возьмем данные из таблицы 9.2.

Таблица 9.2 – Матрица рисков

Альтернативы	*Y*1	*Y*2	*Y*3
D	0,4	0,2	0
B	0	0,1	0,1
C	0,3	0	0,2

$$S = \min_{DBC} (0,4;\ 0,1;\ 0,3) \rightarrow SB. \quad \alpha = 1,\dots, \ ,\mu \in [0,1].$$

Критерий Гурвица: решение принимается с учетом того, что возможны как благоприятные, так и неблагоприятные внешние условия. При использовании этого критерия требуется указать «коэффициент пессимизма» μ – число в диапазоне от 0 до 1, представляющее собой субъективную оценку возможности неблагоприятных внешних условий. Если есть основания предполагать, что внешние условия будут неблагоприятными, то коэффициент пессимизма назначается близким к единице. Если неблагоприятные внешние условия маловероятны, то используется коэффициент пессимизма, близкий к нулю. Для оценки решений используются как выигрыши, так и риски.

Критерий Гурвица для выигрышей:

$$\Gamma = \max_i [\min_{\alpha\ E} (Y)\ \mu + (1-\mu)\cdot \max_{E}(Y)] \ , \ i=1,\ldots, n, \ \alpha = 1,\ldots, A, \ \mu \in [0,1].$$

Критерий Гурвица для рисков:

$$\Gamma = \min [\mu \cdot \max_{R} Y) + (1-\mu)\cdot \min_{R} Y)] \ , \ i=1,\ldots,n.$$

Выводы

1. Изменения в архитектуре и технологиях современных ИК-сетей наряду с тремя направлениями (мобильность, мультисервисность, конвергенция сетей и услуг) включают: распределенное управление обслуживанием вызовов/сессий в сетевых элементах сетей нового поколения; усиление локального интеллекта этих сетевых элементов NGN в условиях сверхплотных сетей (с развитием IoT).

2. МАС строится как совокупность рациональных агентов, объединенных в группы и взаимодействующих друг с другом для достижения общей цели эффективного управления ИК-сетью. Общение агентов друг с другом, сотрудничество, координацию и взаимодействие агентов в системе осуществляют с помощью языка коммуникаций ACL (Agent Communication Language).

3. Предусматриваются типы агентов: исполнители и менеджеры; координаторы, ответственные за организацию взаимодействия агентов; интерфейсы, служащие для связи с внешней средой; агенты, обеспечивающие обмен информацией в системе. Выделяются два типа агентов – реактивные и интеллектуальные. Последние отличаются от реактивных агентов наличием у них встроенной базы знаний и развитого механизма планирования действий.

4. Для решения задач управления ИК и для организации поиска и обработки данных BI в распределенной ИТ-среде оператора эффективным и целесообразным является использование мультиагентного подхода, в рамках которого система строится как совокупность агентов (агенты BI, агенты менеджера и агенты подсистем OSS/BSS).

5. Концепция самоорганизующихся сетей SON была создана некоммерческой организацией 3rd Generation Partnership Project (3GPP). Цель – повышение эффективности и гибкости управления ТК-сетями, сокращение

затрат и увеличение прибыли оператора, была ориентирована на применение в сетях мобильной связи поколений 3/4/5G.

6. Для обеспечения сквозных бизнес-процессов оператора необходимо взаимодействие OSS/BSS систем с сетевыми элементами и системой управления сетью. Системы MW являются промежуточным ПО между оборудованием и приложениями класса OSS/BSS. Системы Middleware будут являться уровнем-посредником между функциями SON, уровнем сети и высокоуровневыми B/OSS приложениями, такими как Inventory, CRM, Billing и др.

7. В условиях неопределенности условий реализации в ИК-сетях можно использовать критерии: Лапласа – предполагается, что вероятности исходов $Y_1, \dots, Y_\alpha \dots, Y_A$ равновероятны; Вальда (критерий осторожного наблюдателя) – решение выбирается в расчете на наихудшие внешние условия; Сэвиджа – решение принимается в расчете на наихудшие внешние условия с использованием матрицы рисков; Гурвица – решение принимается с учетом того, что возможны как благоприятные, так и неблагоприятные внешние условия.

Контрольные вопросы

1. Три тренда в изменениях архитектуры и технологиях современных ИК.
2. Поясните два подхода к управлению ИК.
3. 2 типа агентов в управлении ИК?
4. Для исследования агентов важны 2 параметра?
5. Что системы управления сетью NMS помогают осуществлять?
6. Поясните эксплуатационное управление OAM&P.
7. Поясните критерий Лапласа
8. Поясните критерий Вальда.
9. Поясните критерий Гурвица.
10. Поясните критерий Сэвиджа.

Литература, используемая в главе

1. Бакланов, И. Г. NGN: принципы построения и организации / И. Г. Бакланов ; под ред. Ю. Н. Чернышева. – М. : Эко-Тренз, 2008. – 400 с.

2. Гольштейн, А. Б. Модели и методы управления инфокоммуникационными сетями : дис. д-ра техн. наук : 05.12.13. – СПб.,2019. – 281 л.

3. Гольдштейн, Б. С. Сети связи пост-NGN / Б. С. Гольдштейн, А. Е. Кучерявый. – СПб. : БХВ-Петербург, 2013. – 160 с.

4. Системный анализ и принятие решений в проектной и управленческой деятельности : учеб. пособие / Б. В. Никульшин и [и др.]. – Минск : БГУИР, 2021. – 72 с.

10 ИНТЕЛЛЕКТУАЛЬНЫЕ СРЕДСТВА И СИСТЕМЫ ИНТЕРНЕТА

10.1 Введение в технологию веб-семантик

Проблема автоматизированной обработки в Интернете. Интернет – это в основном сеть компьютеров, объединенных каналами и использующих семейство протоколов TCP/IP для связи между собой. Веб-пространство – сеть сайтов на веб-серверах, использующих гиперссылки для переходов между страницами. Обычное веб-пространство разрабатывается на языке разметки документов HTML, где HTML-страница описывает форму представления информации в веб-браузере. Автоматизировать даже такие простые задачи, как поиск людей, программ в таком Интернете затруднительно [1; 3; 7].

В веб-пространстве введено несколько стандартов, сделавших возможность работы на нескольких уровнях: семейство протоколов TCP/IP обеспечивает транспортировку бит в любое место сети, протокол HTTP и язык HTML обеспечивают средства представления и извлечения гиперсвязей текстовых документов. Эти стандарты и язык Java сделали возможным создание приложений, использующих данные инфраструктуры, что и позволило организовать динамическое веб-пространство.

Третье поколение веб-пространства называют содержащим знания, имеющим целью машинную обработку смысла информации без участия человека. Этот вид веба получил название семантического. Он позволит осуществлять интеллектуальную обработку за счет использования информационных брокеров, поисковых агентов, информационных фильтров и т. д. Но его реализация возможна при разработке уровней взаимодействия выше прикладного [3; 7].

Архитектура веб-пространства обеспечивает пользователей простым интерфейсом гипертекста к разнообразию отдаленных ресурсов, от статических документов для человеческого представления до различных диалоговых средств. Формат данных языка HTML облегчил широко распространенное использование сети, начиная с основного соединения на базе URI до разнообразного стиля текстового процессора, чтобы поддержать основные глобальные функциональные возможности гипертекста. Дополнение форм к HTML обеспечило минимальный, но функциональный пользовательский интерфейс и привело к диалоговому сервису данных.

В то время как инфраструктура HTML облегчила шаг в глобальной информационной технологии, она страдает от некоторых ограничений типа «один размер», соответствующий всем решениям: возможности документа потеряны, поскольку содержание сжато в примитивные структуры HTML.

Когда сеть достигла критической массы как среда для человеческой связи, на следующей стадии строится семантическая сеть, которая включает документы или части документов, описывающие явные отношения между объектами и содержащие семантическую информацию, предназначенную для автоматизированной обработки компьютерами или агентами.

Идея веб-пространства была предложена в 1998 году Тимом Бернерсом-Ли, она представляет собой сеть информационных узлов, которые связаны друг с другом таким образом, чтобы имеющаяся информация могла легко обрабатываться компьютером [1]. Современная тенденция развития Интернета заключается в переходе от документов, «читаемых компьютером» (machine readable) к документам, которые «понимаемы компьютером» (machine understandable). Semantic Web представляет собой сеть информационных узлов, которые связаны друг с другом таким образом, чтобы имеющаяся информация могла легко обрабатываться компьютером. Его можно рассматривать как эффективный способ представления данных в Интернете или как глобально связанную сеть баз данных. Данный проект предлагает реализацию полной системы по автоматизированному созданию и хранению семантического ядра контента, предоставленного в Интернете [7; 8].

Как указал профессор Джон Сова, – Semantic Web – многодисциплинарная тема, которая объединяет теории и методы трех областей: логика (формальные структуры, правила логического вывода); онтология (описание типов сущностей предметной области); теория моделей.

Этап Semantic Web представляет собой переход на уровень знаний и автоматизированной обработки. Технология Semantic Web разрешит компьютеру интерпретировать информацию, представленную в вебе, наравне с людьми, для чего разработана графовая модель описания ресурсов RDF (Resource Description Framework). В общем Semantic Web (по Тиму Бернерсу-Ли) – это: интероперабельность данных между программными приложениями и организациями; набор интероперабельных стандартов для обмена знаниями; архитектура для взаимосвязанных сообществ и словарей [1; 7].

Языки для представления знаний. XML (eXtensible Markup Language) представляет собой простой, мощный и гибкий текстовый формат для описания документов произвольной структуры. XML был разработан и утвержден в качестве стандарта в 1998 году консорциумом W3C для упрощения реализации, а также для обеспечения связи между SGML и HTML. Он является подклассом языка SGML, однако более прост для понимания и обработки.

Функции XML следующие: представление синтаксиса для других языков разметки; семантическая разметка веб-страниц. XML-представление может использоваться на веб-странице вместе с таблицей стилей XSL, что определяет корректный вывод на экран разных элементов; единый формат обмена данных. XML-представление может передаваться между двумя применениями как объект данных. Язык XML разрешает каждому создавать свой собственный формат документов и потом писать документы в этом формате. Эти форматы документов могут включать разметку, которая уточняет содержание контента документа. Документ с разметкой может «читаться» компьютером.

RDF. Для описания предметной области ресурсов предложен стандарт RDF (Resource Description Framework), принятый в 1999 году консорциумом W3C и поддержанный многими ведущими производителями ПО. RDF представляет собой модель описания метаданных. Этот язык использует XML-

синтаксис [2; 7]. В то время как модель данных XML является графом с обозначенными вершинами и необозначенными дугами (без связей), модель данных RDF является графом с обозначенными как вершинами, так и дугами, который разрешает определять связи между сущностями. Целью модели RDF является стандартизировать определение и использование метаданных, которые описывают ресурсы веб [2; 7].

Стандарт RDF включает две основные части – первая, способ описания ресурсов, а также способ задачи схем, по которым ресурс описывается. Эта часть RDF определяет простую модель для описания объекта, который рассматривается в качестве ресурса, как связей между ресурсами в терминах поименованных свойств и значений. Вторая часть (RDF Schema – RDFS) служит для задачи структуры предметной области и аналогична диаграмме классов в UML. На RDF можно описывать как структуру ресурса, так и связанную с ним предметную область. RDF описывает ресурсы в виде ориентированного размеченного графа – каждый ресурс может иметь свойства, которые в свою очередь также могут быть ресурсами или их коллекциями [2; 7].

Базовый строительный блок в RDF – это тройка «объект – атрибут – значение», который часто записывают в виде A(O, V), которая читается как «объект O имеет атрибут A со значением V». Такую связь можно также представить как ребро с меткой A, которое объединяет два узла, O и V: [O] – A –> [V]. Такая нотация довольно полезна, поскольку RDF разрешает менять местами объекты и значения. Таким образом, каждый объект может играть роль значения, которое в графическом представлении отвечает цепочке из двух ребер с метками. RDF допускает форму представления, в которой любое выражение RDF в тройке может быть объектом или значением, т. е. графы могут быть как вложенными, так и линейными [2]. В вебе это разрешается, например, выражать сомнение или согласие с выражениями, созданными другими людьми [7].

Пути к семантической сети. Язык XML зародился как проект для снятия ограничений языка HTML на структурированные документы, выбирая простой для реализации, но все же расширяемый набор из стандарта SGML для использования в сети. Он появился как инфраструктура для структурированного представления и также обмена данными. Тем временем в усилиях направить дальнейшее воздействие сети на общество члены консорциума W3C пришли к выводу, что необходимо создать платформу для интернетного контентного выбора (Platform Internet Content Search − PICS), которая обеспечит пользователей способностью выбрать содержание на основании меток, при этом поддерживая провайдеров и других потребителей. Критический компонент PICS − описание системы оценки, схема; каждая метка PICS указывает на описание в сети, на поля в метке.

PICS был разработан как первый шаг к обобщенным меткам, которые позволят любую составляющую сети сделать понятной по качеству ресурсов: терминов и условий для использования. Metadata выполняет необходимую работу, чтобы закончить представление: структурированных меток, правил, интеграцию с цифровыми подписями. Проект меток PICS был обобщен для инфор-

мационной модели, использующей направленные помеченные графы (DLGs). Это модель RDF, преобразование в последовательную форму было определено в синтаксисе XML. Оценки системы PICS были включены как специальные случаи в проекте RDF-схем.

XML-документы имеют механизм для самоописания в виде описание текста документа (Document Type Definition − DTD). Поскольку использование XML стало более разнообразным и интенсивным, ограничения DTD стали актуальными, особенно в области печатания данных, модульности, и повторного использования. Поэтому консорциум W3C начал работу над XML в области создания поколения XML-схем. Намерение состояло в том, что язык RDF будет просто встроен в язык XML. Но затем проект RDF начал бы включать списки имен для соединения с XML именами элементов к адресам сети, что привело к усложнению.

Через какое-то время взаимодействия усложнялись. Модель объекта документа (Document Object Model − DOM), которая была создана как усилие согласовывать скрипты HTML для обслуживания в браузерах, была расширена включением XML и стала основополагающим прикладным программирующим интерфейсом (API) для многих платформ программного обеспечения сети и структурированных репозитариях данных. ПО, основанное на этих платформах, рассматривает RDF не как XML потоки, но как объекты проекта DOM. Появление компонента преобразования (Extensible Style Language − XSL) в качестве полезного компонента позволило по-новому посмотреть на многих из синтаксических проблем проекта в RDF. Преимущество синтаксиса, которое позволяло легко управлять XSL, не было очевидным в ранней стадии проекта RDF.

Представление знаний при помощи RDF. Существует две основные технологии для представления знаний в Semantic веб: языки XML и RDF. XML позволяет каждому разрабатывать собственные теги − неотображаемые метки вида <zip code> или <alma mater>, которые размечают саму веб-страницу или фрагменты ее текста. Скрипты, или программы, могут читать самые сложные тексты, но их создателям необходимо знать все теги, использованные при написании страницы. Другими словами, XML позволяет пользователям добавлять произвольную структуру в документы, не объясняя, что эта структура означает.

Смысловое значение выражения описывается с помощью языка RDF, который кодирует его в набор троек, причем каждая тройка примерно соответствует подлежащему, сказуемому и дополнению (object, verb, subject) в обычном предложении. Эти тройки могут быть описаны в том числе и с помощью тегов XML. Суть RDF в том, что конкретному объекту (человеку, веб-странице или еще чему-либо) присваивается атрибут (является сестрой, является автором) с определенным значением (другой человек, другая веб-страница).

Данной структуры оказалось вполне достаточно для того, чтобы описать абсолютное большинство данных, обрабатываемых автоматически. Субъекты и объекты единственным образом идентифицированы с помощью универсального идентификатора ресурсов (URI Universal Resource Identificator) и используются на веб-странице в качестве ссылок (URL, Uniform Resource Locator), явля-

ется распространенным типом URI). Связки также идентифицируются с помощью URI, и любой желающий может создать новую связку, новое понятие, определив для него URI на какой-либо веб-странице.

Тройки, окутывающие паутиной информацию RDF-документа, кодируется с помощью URI. Благодаря этому возникает уверенность в том, что любое понятие на веб-странице не просто написано, а привязано к уникальному определению, по которому оно отыскивается в Интернете. Для примера: имеется доступ к различным базам данных служащих, содержащим среди прочей информации адреса. Когда требуется найти тех, кто имеет определенный индекс, то необходимо знать, какое поле в каждой базе данных содержит фамилии, а какое индексы. Для этого RDF использует URI-коды, а не фразы, для того чтобы описывать отношения: «(поле 5 в базе данных А) (является полем типа) (индекс)». Одно из приложений RDF − RSS (RDF Site Summary) − применяется для автоматического обмена ссылками на заглавные материалы между родственными сайтами и поддерживается свободным браузером Mozilla.

Онтологии. Две базы данных могут использовать разные идентификаторы для обозначения одного и того же, например, индекс. Программа, которая сравнивает, дополняет или сопоставляет данные двух баз должна знать, что эти два понятия используются для обозначения одного и того же. В идеале программа должна иметь алгоритм для распознавания одного и того же, сказанного по-разному, однако это пока еще остается теорией [3; 6].

Решение этой проблемы выражается в третьем по счету основном понятии Semantic веб (кроме) − онтологии, т. е. общему для нескольких источников списку данных. Вообще, онтология как раздел философии содержит учение о бытие как таковом, онтология как дисциплина изучает теории о структуре бытия, природе вещей. Специалисты же по ИИ и Semantic веб включили данное понятие в свой жаргон, чтобы обозначать документ или файл, формально определяющий отношения терминов. Наиболее часто встречающаяся онтология в веб-пространстве включает таксономию и набор правил вывода.

Таксономия определяет классы объектов и отношения между ними. Например, адрес может быть определен как тип месторасположения, а код города − это не что иное, как тип городского адреса, и так далее. Классы, подклассы и отношения внутри составляющих таксономии − это мощный инструмент при использовании веба. Большое разнообразие отношений достигается прежде всего приданием различных свойств классам и возможностью наследовать эти свойства подклассам. Если код города относится к типу город, и большинство городов имеют веб-сайты, то можно считать код города привязанным к соответствующему веб-сайту, если нет БД, напрямую связывающей код города и веб-сайт.

Правила вывода в онтологии несут не менее важное значение. Предположим, имеется правило: «Если код города связан с кодом области этого города, то полный городской адрес содержит также и код области». В таком случае программа может использовать дедуктивный метод, чтобы по адресу университета МИУ, например, определить, что тот находится в Минске, а значит, явля-

ется университетом Республики Беларусь. Нельзя сказать, чтобы компьютер понимал смысловое значение, но детально обрабатывать данные, разбираться в терминах и отношениях, делать выводы – за счет этого он «понимает» прочитанное, являясь тем самым эффективным помощником человека.

Агенты Semantic Web. О реальной базе Semantic Web можно говорить тогда, когда будет создана не одна, а множество программ, как можно обширнее покрывающих содержимое всемирной сети, обрабатывающих информацию и обменивающихся между собой результатами. Эффект от использования таких программ, так называемых автоматических служб и агентов, экспоненциально возрастает с появлением все новых и новых автоматизированных страниц и агентов. Технология Semantic Web рассчитывает, что все агенты, будут «добывать» информацию, «помогая» друг другу.

Важным аспектом функционирования агентов является обмен «подтверждениями», написанными на унифицированном языке Semantic Web (языке, выражающим правила логического вывода и информацию, подобную той, что отражена в онтологии). При необходимости проверить достоверность поиска компьютер обращается к специальной службе за подтверждением. Далее текст, осмысленный программой, переводится на унифицированный язык Semantic Web, встроенный в компьютер механизм логического вывода считывает данные, сопоставляет их и выясняет, верная ли информация найдена, представляя ключевые веб-страницы. Состояния готовности таких специальных служб сегодня, как впрочем, и Интернета вообще, не позволяет продемонстрировать технологию подтверждения по описанной выше схеме, однако показать текущие предварительные версии унифицированного языка Semantic Web вполне могут.

Semantic Web предлагает более гибкую систему, в которой агент-продавец и агент-покупатель могут достигнуть взаимного понимания путем обмена онтологиями, содержащими словарь необходимого для дискуссии. Агенты могут даже выдвигать новые идеи, касающиеся осмысления текста, при открытии ими новых онтологий. Семантика также позволяет воспользоваться сервисом, лишь частично удовлетворяющим запросу [1; 3].

Обычный процесс работы веб-агента подразумевает создание так называемой «цепочки значений»: составляющие информации передаются от одного агента к другому, причем каждый добавляет свое значение так, что формируется конечный готовый информационный продукт, запрашиваемый пользователем. Для сложного запроса некоторые агенты помимо семантики используют технологии ИИ. Но ведущая роль все же остается за Semantic Web, закладывающей основы и стержень для осуществления интеллектуальной обработки.

10.2 Подходы к архитектуре семантического веб-пространства

Требование для модели сети. Традиционно документы и базы данных были определены так, что производитель и потребитель имели предварительное соглашение по структуре информационных единиц. Но это стало недостаточно

для долгосрочного развития семантической сети. Семантическая сеть должна разрешить распределенным пользователям работать независимо, но чтобы взаимопонимание увеличивалось с добавлением новой информации без изменения существующей. Этот подход позволяет пользователям решать двусмысленности и разъяснять несуразности [3; 4; 7].

Поэтому семантическая сеть должна быть основана на средстве, которое может расширяться, поскольку понимание человека расширяется. Это средство должно иметь возможность захватывать информацию, которая связывает независимые представления накладывающихся областей знаний. Спецификация XML 1.0 позволяет обмениваться частичной информацией, что дает возможность распознавать часть документа. XML специфицирует синтаксические конструкции, называемые «хорошо сформулированными», которые являются средством, позволяющим документам включать внешнюю информацию, но обрабатываемую старыми средствами этого уровня.

Способности вести переговоры. Увеличение децентрализации приложений в процессе развития семантического веба требует от документов включать смешанные элементы из различных источников. Однако комбинаторное количество делает практически невозможным определение множества типов документов, понимающих все возможные словари. Тем не менее возможности XML разрешают смешивать словари. Модель RDF усиливает эту возможность. В процессе работы над XML-схемой предложены пути объединения типов документов, используя пространство имен.

XML-документы как и таблицы реляционных БД ограничены, но веб-пространство практически безгранично. Фундаментальность его отлична от традиционных гипертекстовых систем в интеграции резервирования связей для расширяемости. Поскольку каждая часть поддерживает связи в ограниченном веб-пространстве, нет средств, которые могли бы рассматривать это пространство в целом.

Допуская ограниченное пространство, процессор запросов может предположить, что он имеет полную информацию и может решить, что нет элементов или записей, удовлетворяющих запросу. Поскольку существуют средства, которые обрабатывают запросы в форме «найти все связи X в веб-пространстве», они не могут решить, что нет связей с X, если только нет знаний в вебе, что действительно так. Обсуждение показало, что проблема запросов, ограниченная XML-набором, решена в некоторых параметрах и возможно в сообществе.

Ожидается, что данные, ограниченные в приложениях, будут комбинироваться позже с другими данными в вебе. Приложения, которые выполняются в веб-пространстве, должны иметь возможность использовать общие рамки для объединения информации из других приложений.

Связанность HTML требует, чтобы все связи из ресурса были выражены в его контексте. Но это ограничение HTML, а не веб-архитектуры. Развитие связей XML такое, что разрешаются связи из ресурса, выраженные вне его. Однако, если связи HTML и XML ведут от одной к другой, в семантическом вебе

данные должны быть связаны любые с любыми. Требования к модели данных семантического веб-пространства таковы, что нет фундаментальных ограничений связей типа, что сказано, о чем и где.

Глобальная универсальность и локальные ограничения. Семантическое веб-пространство должно обеспечить путь объясняющей информации из различных систем. Эти системы могут использовать различные модели данных, что означает требование общей концепции данных на нижнем уровне. К такой модели относится прямой помеченный граф.

Другая особенность семантического веб-пространства в том, что поддерживается план как для существующих и будущих систем. Оптимизация – способность нумеровать и индексировать все объекты данного типа – важна для локальных операций, но не для веб-пространства. Примером является область действия идентификаторов. В объектной модели тип переменной объявляется после объявления типа объекта. Модели категории связанности аналогичны моделям с ненумерованными наборами свойств.

Механизм, адаптированный в RDF для управления выражения ограничений, делает все объекты, связи, типы, утверждения первым классом объектов веба. Это позволяет семантическому вебу делать утверждения о самом себе.

Все предложения в вебе появляются в контексте. Контекст необходим приложениям для определения правдоподобия предложений, при машинной обработке, не все предложения рассматриваются как истинные. Истинность или более прагматично – правдоподобность оценивается для информации, найденной в вебе. Коммерческие приложения развили технологии обмена информации без требования истинности.

Хотя проектирование веба резервирует интеграцию связей для расширяемости, понятие «все знания об этой вещи здесь» не может фиксироваться, когда экспортируются в веб-пространство БД и объекты. Большой вклад, связанный с этим допущением, заключается в том, что хотя гипертекстовые связи соединяют различные информационные системы, семантический веб будет связывать данные из различных систем, разрешая их сложную и удаленную обработку.

Необходимо рассмотреть оптимизацию, которую не включает язык RDF, и находящуюся в системах БД. Она связана с операциями над составными объектами и часто представляется как локализация. Операции, такие как удаление, сравнение над структурными объектами, часто используют локализацию. С этими ограничениями локализации трудно иметь дело в Интернете, поэтому эти отношения должны быть выражены в семантическом вебе.

Понятие архитектуры веба. С точки зрения архитектуры Semantic Web можно рассматривать как три яруса [7; 8]:

− *базис*, который состоит из уникальной глобальной идентификации ресурса, метаданных для декларирования фактов о ресурсах и общего языка для выражения метаданных и знаний. Последний реализован с помощью онтологий для общедоступного понимания и общего словаря метаданных и правил для добавления новых метаданных и знаний;

− *базовый сервис,* например логический вывод и запросы к метаданным и онтологиям, разъяснение таких выводов, управление доверием (trust), агенты, поисковые системы, серверы онтологий;

− *сервисы приложений,* например сервис агентства путешествий, сервис оптовой покупки и т. д.

Фундаментальными основами Semantic Web являются: архитектура WWW (URI/IRI, Unicode, XML, HTTP); графовая модель представления полуструктурированных данных (OEM, Lore); формальная логика (первого порядка, базы знаний, фреймы); криптография с открытым ключом.

URI. В вебе для идентификации элементов используются унифицированные идентификаторы ресурсов URI (компактная строка символов, которая используется для идентификации абстрактного или физического ресурса). Одна из форм URI есть URL (Uniform Resource Locator), унифицированный указатель ресурса. URL это адрес, по которому загружается веб-страница. В 2005 году на смену URI был предложен интернационализированный идентификатор ресурса – Internationalized Resource Identifiers (IRI), идентифицирующий абстрактный или физический ресурс на любом языке мира. В будущем идентификаторы IRI призваны заменить URI [7; 8].

10.3 Представление знаний в веб-пространстве

*Деревья, графы и таблицы***.** Распределенная модель данных, являющаяся ядром языка запросов, важный вопрос проектирования. Наиболее хорошим примером является реляционное исчисление, реализованное в SQL. XML-спецификация определяет элементы документа в форме дерева, а RDF-спецификация определяет модель данных в форме прямого помеченного графа (Direct Label Graph − DLG). Проектирование языка запросов для XML и RDF должно начинаться соответственно с этих моделей. В отдельных работах отмечается, что модель DLG может быть использована в XML [2; 7].

Эта модель поддерживает отношения родители – дети, выражая элементы локализации. Поскольку она работает с отдельным документом, эти разработки зависят от связанности XML, т. е. хорошо определенных конструкций для соединений внутри документа. Многие приложения метаданных требуют, чтобы базовые структуры RDF утверждений были видимы всеми системами, хотя последние не все понимают семантику данных утверждений.

Содержание выбранного приложения добавлено как операционное требование, которое утверждение делает видимым при обращении без ссылки к схеме. Ожидания от HTML предлагают, что авторы имеют возможность выражать простые конструкции связанности XML прямо в документе без обращения к схеме или описания его типа (DTD). Будущие работы будут связаны с идентификацией семантических дуг и кросс-связей XML-документа.

Реляционные данные. Модели семантического веба тесно связаны с моделью реляционных баз данных. Набор RDF предложений по вершинам графа соответствует строкам таблицы. Соединение в БД соответствует расщеплению в

графе. Реляционные БД оптимизируются путем обработки большого количества предложений с одинаковыми свойствами и это может соответствовать оптимизации в XML-последовательности для большого объема одинаковых данных. Следует ожидать, что базовые структуры, которые поддерживают последовательность реляционных БД, могут быть совместимы с графовой моделью данных RDF [2; 7].

Многие записи таблицы могут иметь одинаковые свойства. Отдельная ячейка, соответствующая свойству, строго типизирована. Комбинированные правила РБД слабосвязанные, и запрос может объединять колонку таблиц, соответствующих типов данных без проверки семантики. Объединение отдельных полей – другой случай, когда вынужденные ограничения в семантическом вебе не могут быть абсолютными. Семантический веб не разрабатывается как новая модель данных, а в качестве поддержки связей между многими различными моделями.

Типы объектов. Многие из объектно-ориентированных представлений моделируют функции объектов. Для семантического веба представляет интерес модель данных последовательности объектов. Эта последовательность может быть представлена как последовательность полей данных, выражающих различные свойства объектов. В большинстве объектных систем тип объектов обозначает ограничения методов, поддерживаемых объектами. Последовательность данных объекта часто рассматривается во внутреннем представлении и она скрыта от вызова методов. Используя этот принцип, XML-технология связана с последовательностью методов удаленных вызовов и не может быть использована для обеспечения взаимодействия между реализациями [2; 7].

Тем не менее, когда взаимодействие является целью для последовательности объектов, оно становится допустимым для них в вебе. В этом случае проектирование самоописываемых форматов, используя XML, делает объекты более живучими во времени. Необходимы словари XML для представления скрытия данных, когда объектные системы представляются в семантическом вебе. Этот механизм поддерживается RDF и может быть совместно использован другими XML- приложениями.

Описание связей. Первым элементом Semantic Web является простая модель типизации данных. Схема – это средство для описания содержания и связи между термами. Так же как XML, Schema используется для определения словаря, RDF Schema разрешает разработчикам определять конкретный словарь для данных RDF и указывать виды объектов, к которым могут применяться эти атрибуты. Механизм RDF Schema предоставляет базовую систему типов для моделей RDF [2].

На основе RDF в 2003 был предложен рабочий проект RDF Vocabulary Description Language 1.0: RDF Schema. Схема RDF была разработана как простая модель типизации данных для RDF. RDF является языком общего применения для представления информации в Интернет. Данная спецификация описывает, как использовать RDF для описания RDF-словарей. Она определяет базовый словарь, предназначенный для этих целей, и принятые соглашения,

которые могут быть использованы при создании приложений Semantic веб для поддержки более сложных словарей RDF-описаний. Язык описания словаря RDF определяет классы и свойства, которые могут быть использованы для описания других классов и свойств, а также производить некоторые более сложные вещи, такие как создание диапазонов и областей для свойств.

Три наиболее важных понятия, которые дает нам RDF и схема RDF, – это «Ресурс» (rdfs:Resource), «Класс» (rdfs:Class) и «Свойство» (rdfs:Property). Эти понятия являются «классами» в том понимании, что этим классам могут принадлежать термины. RDF Schema определяется в терминах базовой информационной модели RDF – структуры графа, который описывает ресурсы и свойства.

Все словари RDF используют некоторую базовую структуру: они описывают классы ресурсов и типы связей между ресурсами. Эта общность разрешает использовать разнородные словари, созданные для машинной обработки, и отвечает требованиям по созданию метаданных, в которых утверждения могут быть получены из множества разнородных децентрализованных словарей, созданных различными сообществами по разным принципам и разными методами [7; 8].

Описание с помощью RDF не ограничивается только описанием документов Интернета. Этот стандарт довольно универсальный и гибкий для того, чтобы описывать большинство типов структурированных данных. Например, в RDF естественно выражаются диаграммы сущность-связь, которые широко применяемы для проектирования баз данных. Описание семантики ресурса на RDF может быть как «внешним», когда описывается ресурс в целом, так и «внутренним», когда описывается внутренняя структура ресурса – будь-то база данных, XML-документ или целый сайт.

Важной особенностью стандарта RDF, как и лежащего в его основе XML, является расширяемость. На RDF можно задать структуру описания источника, используя и расширяя встроенные понятия RDF-схем, такие как классы, свойства, типы, коллекции. Модель схемы RDF включает наследование; наследоваться могут как классы, так и свойства [7].

Кроме описания структуры, RDF разрешает оперировать утверждениями. Выражение «ресурс R1 как свойство P имеет ресурс R2» можно проинтерпретировать и как предикат P(R1, R2), а потом использовать это утверждение как объект других утверждений. Такая интерпретация разрешает описывать с помощью RDF концептуальную информацию. Таким образом, RDF целиком подходит на роль универсального языка описания семантики ресурсов и взаимосвязей между ними [8].

*RDF- и RDF-схема***.** Рассмотрим основные черты RDF- и RDF-схемы (для краткости RDFS) с критикой отдельных решений. Пререгативой семантического веба являются машинно-обрабатываемые семантики информации. Фреймы для описания ресурсов − RDF являются основой для обработки метаданных, они обеспечивают взаимодействие между приложениями для обмена машинно-распознаваемой информацией в веб-пространстве. В целом RDF определяет модель для описания машинно-обрабатываемой семантики данных, которая состоит из объектов трех типов.

Ресурсы, которыми могут быть отдельные веб-страницы, их части или их наборы, а также непрямые веб-объекты (книги), имеющие URI. Свойства – специфические характеристики, атрибуты или связи, используемые для описания ресурсов. Ресурсы вместе с поименованными свойствами и величинами этих свойств для ресурса рассматриваются как RDF-предложение [2].

Три части предложения называются соответственно субъект, предикат и объект. RDF определяет тройку − объект-свойство-величина в качестве базовых примитивов и для них вводит стандартный синтаксис. Например

```
<rdf:RDF>
<rdf:Description about=«http://www.w3.org»>
<Publisher>World Wide Web Consortium</Publisher>
</rdf:Description>
</rdf:RDF>
```

указывает, что http://www.w3.org (субъект) имеет (предикат) издателя W3C (объект). Предложения могут быть связаны в цепочку

```
<rdf:RDF>
<rdf:Description about=«http://www.w3.org/Home/Lassila»>
<Creator rdf:resource=«http://www.w3.org/staffId/85740»/>
</rdf:Description>
<rdf:Description about=«http://www.w3.org/staffId/85740»>
<Email>lassila@w3.org</v:Email>
</rdf:Description>
</rdf:RDF>
```

Пример показывает, что http://www.w3.org/Home/Lassila (субъект) создан сотрудником с номером 85740 (объект). В следующем предложении этот ресурс (сотрудник с номером 85740) выполняет роль субъекта для утверждения, что его электронный адрес − lassila@w3.org. Наконец, предложения являются также ресурсами, определяемыми рекурсивно, разрешая вложенность. Все это приводит к модели данных в виде помеченного гиперграфа с каждым предложением, являющимся предикатно помеченной связью между объектом и субъектом. Каждая вершина гиперграфа может быть другим графом.

Язык OIL. Он базируется на развитии XML с расширениями, использующими логику описаний (DL), произвольными булевыми выражениями, которые появляются при появлении имени класса. Определение слота может рассматриваться как класс и использоваться в виде выражения:

− определение класса имеет дополнительное поле, которое определяет его как примитив или не примитив;

− глобальные определения слота расширены для спецификации суперслотов и связи таких свойств, как транзитивность и симметричность;

− дополнительные правила, накладываемые XML-документами, не требуются;

OIL также ограничивает XOL в следующих моментах:

− будет поддерживаться концептуальное моделирование, но не индивидуализм;

− не поддерживаются некоторые ограничения слотов;

− слот inverse может быть определен только в глобальном слоте.

OIL-онтология состоит из списка $d_1, \ldots d_n$, где каждый элемент это определение класса или слота. Класс определений является парой (CN, D) или тройкой (CN, P, D), где CN − имя класса, D − описание класса, P – примитивы. Класс описаний состоит из подкласса компонент, выражений и ограничений. Слот ограничений состоит из имени и списка $SN(a_1, a_2, \ldots a_n)$. Ограничение может быть в виде:

− величина ограничения списком класса выражений, записываемое для любого $C_1, \ldots C_n$.

Для того чтобы поддержать решаемость языка, кардинальные ограничения должны быть применяемы к простым слотам, который не транзитивный и не имеет транзитивных подслотов. Определение слота − пара (C, D), где CN − имя слота, D – описание слота. Описание слота состоит из компонент = списка имен, за которыми следуют ограничения. Определение слота может быть записано

$$SN[R_{N1}; : : : ; R_{NR}; S_1; : : : ; S_m]$$

*Использование OIL для RDF/RDF-схемы***.** Установки стандартов не только для структуры синтаксиса документов, но и для семантики. Обработка семантики поддерживается средствами XML, RDF и их схем. Рассмотрим следующие доработки: представление знаний и вывод на верхнем уровне веба, который будет базироваться на верхнем уровне RDF. Для этого разработан язык OIL, который встраивается в структуру семантического веба. Он объединяет черты фреймовых языков и формализм логики описаний.

Одной из ключевых идей нового языка является то, что он имеет выразительные примитивы, которые могут быть использованы в RDF/RDF-схеме. Предпочтение RDF отдано более чем XML потому, что второй выражает правила грамматики и структуру документа, а первый разработан для обработки семантики представлением модели области в терминах объект-связь-объект и техника представления знаний может быть использована для нахождения связи между отдельными описаниями. Конечно, это не решает проблему межобработки, т. е. нахождения семантического соответствия между объектами, но использование RDF для обмена данными повышает уровень повторного использования. Поскольку уровень XML независим от уровня RDF, модель области последнего может использоваться при изменении синтаксиса первого [2; 3; 7].

10.4 Онтологии

Понятие онтологии. Онтологии играют определяющую роль в обработке и разделении знаний между программами в вебе. Онтологии определяются как представление разделяемых концептуализаций. Они обеспечивают разделяемое и общее понимание доменов для взаимодействия между людьми и прикладными системами и разработаны для облегчения и использования знаний компьютерами и агентами [1; 3; 7; 8].

Первое использование онтологий в вебе, основанном на знании, применяется в сайтах электронной коммерции, где они необходимы для:

− установления машинной коммуникации между покупателями и продавцами;

− для становления вертикальной интеграции рынков;

− для повторного описания различных торговых площадок.

Второе направление использования онтологий – это поисковые машины, в которых при переходе от одного понятия к другому находятся страницы с разными по написанию, но одинаковыми по смыслу словами. Типично онтологии содержат иерархическое описание важных концепций данной области и описывают свойства каждого концепта, используя механизм величин атрибутов. Дальнейшие связи между концепциями могут описываться логикой семантики.

Онтологии в общем виде определяются как совместно используемые формальные концепции конкретных предметных областей, они дают общее представление о понятиях, информацией из которых могут обмениваться люди и приложения. Они разрешают наполнять понятием домен фиксированием сущностей (entities) и связей в домене. Указание, в каких связях принимает участие сущность, частично разрешает понять и ее значение (содержание), поскольку это предоставляет возможность видеть, где данная сущность входит в отношения с другим доменом [7; 8].

Онтологии основываются на математическом аппарате формальной логики (descriptive logic, DL), малое подмножество которого охвачено RDF-схемой. DL является подмножеством логики первого порядка, которое вычислимо. Дополнительные возможности в дополнение к имеющимся в RDF являются целью онтологических языков, таких, как DAML+OIL и OWL. Данные два языка основаны на RDF и RDF Schema. Цель данных языков – обеспечение ресурсов дополнительной машинно-обрабатываемой семантикой, т. е. они направлены на обеспечение машинного представления ресурсов в форме, которая более соответствует их оригиналу из реального мира [7; 8].

Разметка документов Semantic Web с помощью онтологических терминов позволит производить автоматическую обработку их содержания. Таким образом, онтологии определяются как ключевая технология для развития Semantic Web. Онтологии в состоянии сыграть критически важную роль в организации обработки знаний на базе веб, их общего использования и обмена ими между приложениями.

*Язык OWL***.** Наиболее развитым языком представления онтологий в настоящее время является OWL (Web Ontology Language), который расширяет возможности XML, RDF и RDF Schema. Этот язык основан на DAML+OIL. Проблемы, которые возникли в DAML+OIL, были вызваны постоянным изменением ядра спецификаций RDF, на котором основан DAML+OIL [3; 8].

Онтология OWL является последовательностью аксиом и фактов с добавлением ссылок на другие онтологии, которые считаются включенными в онтологию. Онтологии OWL являются веб-документами и на них можно ссылаться. Онтологии также имеют не связанную с логикой компоненту (пока еще не

определенную), что может быть использовано для записи авторства, и другая не связанная с логикой информация, ассоциированная с онтологией. Фактически это словарь, который расширяет набор терминов, определенных в RDFS.

Онтологии включают информацию о классах, свойствах и частных случаях, каждый из которых может иметь идентификатор ID, который является ссылкой URI. OWL имеет три модификации: OWL Lite (простой); OWL DL (с полной разрешимостью); OWL Full (с полной выразительной мощностью). Каждая из этих модификаций (кроме Lite) является расширением предыдущей. Как следствие: любая OWL Lite онтология является OWL DL онтологией, а любая OWL DL онтология является OWL Full онтологией [7; 8].

Главные характеристики языка веб-онтологий OWL: использует синтаксис XML; имеет инструкции для представления дерева классов и инструкции для указания принадлежности индивидов классам; включает систему описания свойств: область определения, область значений; может задавать характеристики свойств: симметричность, транзитивность, функциональность; имеет инструкции для указания эквивалентности (склеивание) классов [7; 8].

Онтология OWL является последовательностью аксиом и фактов с добавлением ссылок на другие онтологии, которые считаются включенными в онтологию. Онтологии OWL являются веб-документами и на них можно ссылаться. Онтологии также имеют не связанную с логикой компоненту (пока еще не определенную), что может быть использовано для записи авторства, и другая не связанная с логикой информация, ассоциированная с онтологией. Фактически это словарь, который расширяет набор терминов, определенных в RDFS [8].

Онтологии включают информацию о классах, свойствах и частных случаях, каждый из которых может иметь идентификатор ID, который является ссылкой URI. OWL имеет три модификации: OWL Lite (простой); OWL DL (с полной разрешимостью); OWL Full (с полной выразительной мощностью).

Каждая из этих модификаций (кроме Lite) является расширением предыдущей. Как следствие: любая OWL Lite онтология является OWL DL онтологией, а любая OWL DL онтология является OWL Full онтологией. Главные характеристики языка веб-онтологий OWL: использует синтаксис XML; имеет инструкции для представления дерева классов; имеет инструкции для указания принадлежности индивидов классам; OWL имеет систему описания свойств: область определения, область значений;

Язык DAML (DARPA Agent Markup Language) был разработан агентством передовых оборонных исследовательских проектов (Defense Advanced Research Projects Agency) в 2000 году как расширение XML и RDF. Последняя версия языка DAML+OIL обеспечивает большой набор конструкций для создания онтологий и разметки информации таким образом, чтобы компьютеры были способны их прочитать и понять. В этой связи необходимо также упомянуть еще одну разработку DARPA – язык DAML-S – Semantic Markup for Web Services.

Как указывается в основном проекте, OWL почти полностью похож на DAML+OIL. Основные и существенные отличия от DAML+OIL состоят в следующем: устранение некоторых ограничений; способность прямо указывать,

что свойство может быть симметричным; устранение некоторых неиспользуемых конструкций DAML+OIL, особенно ограничение с дополнительными компонентами [3; 7].

Существует также несколько маловажных расхождений, которые включают в себя некоторые изменения имен некоторых конструкций, однако основная цель, преследуемая при создании OWL, заключалась в том, чтобы максимально корректно сохранить имена DAML+OIL [7; 8].

DAML+OIL является языком семантической разметки для веб-ресурсов. Он основывается на ранних стандартах W3C, таких как RDF и RDF Schema, и расширяет эти языки более полными примитивами моделирования. DAML+OIL обеспечивает примитивы моделирования, которые по обыкновению используются в языках, основанных на фреймах. Онтология DAML+OIL (или база знаний, knowledge base) есть коллекция RDF – троек. Онтология, как правило, содержит иерархию понятий предметной области и описывает важные свойства каждого понятия с помощью механизма «атрибут – значение». Связи между понятиями могут быть описаны с помощью дополнительных логических утверждений.

Для семантического веба необходимо иметь универсальный язык представления знаний. По ряду технологических и прагматических причин этот идеал недостижим и необходимо работать с множественным представлением метаданных. Но тем не менее язык RDF-схемы достаточен для выражения необходимого верхнего уровня на базе примитивов RDF. Определение отнологии в RDF означает определение RDF-схемы, которая в свою очередь определяет термины и связи нужного языка. В этой онтологии два аргумента подкласса объектов – класс и класс выражение. Рассматривая класс-выражение как объект, устанавливается факт, что выражение определяет новый класс [7; 8].

Поскольку каждая онтология использует собственное множество имен, термы из разных онтологий могут быть смешаны в одном RDF документе без конфузий. Поскольку RDF определяет четкую структуру имен, возможно сделать утверждение, что для одного языка термины определены в другом языке. Заметим, что это невозможно для XML-языка, поскольку в нем нет средств определения для тегов (объект, атрибут, величина) и нельзя сделать предположение о структуре объекта. В целом предложения могут быть сформулированы в виде [7; 8]:

– использование RDF-схемы для описания моделирующих примитивов языка;

– использование результирующего документа RDF-схемы для описания специфической онтологии;

– использование документов RDF-схемы из 1 и 2 для описания примеров онтологии языка.

Такая техника может быть применена для любого языка представления знаний. Тот факт, что OIL поддерживает базовые примитивы RDF-схемы (rdfs:subClassOf и rdfs:subPropertyOf), делает это возможным, поскольку его онтологии будут доступны для любых приложений RDF-схемы. Более того, пря-

мое соответствие с SHIQ DL может быть использовано для обеспечения поддержки механизма рассуждения в OIL. Основное использование OIL может быть в разработке онтологии, которая может быть использована в приложениях, доступных только RDF-схеме. Однако в будущем развитие информированности приложений языка будет иметь более сильный семантический смысл.

Системы Ontolingua [6] разработаны в Лаборатории систем знаний (Knowledge System Laboratory) Стэнфордского университета. Система предназначена для хранения и ведения библиотеки формальных непроцедурных описаний фрагментов моделей и понятий некоторых предметных областей. Она обеспечивает удаленный гипертекстовый доступ пользователей к библиотеке, поиск и некоторую поддержку процесса формализации пользователем его понятий и задач на основе формальных описаний, хранящихся в системе. Язык Ontolingua использует принципы объектно-ориентированного подхода и является расширением компьютерно-ориентированного языка Knowledge Interchange Format (KIF) для обмена знаниями и взаимодействия между программами.

10.5 Логический вывод в веб-пространстве

Недостатки представления знаний. Как утверждают авторы стандарта RDF, он имеет ряд отсутствующих свойств, которые они указывают как следующие [3; 4; 7]:

− невозможность указания мощности множества значений свойства, например, что «Человек имеет только одного биологического отца»;

− невозможность указания того, что представленное свойство (например, hasAncestor – имеет предка, прототип) является транзитивным, например, что «если A hasAncestor B, и B hasAncestor C, тогда A hasAncestor C»;

− невозможность указания того, что два разных класса, определенных в разных схемах, фактически представляют одно и то же понятие;

− невозможность указания того, что два разных экземпляра (instances), определенные раздельно, фактически представляют один и тот самый субъект;

− невозможность определения новых классов в терминах операций (например, объединение и пересечение) над другими классами.

Эти недостатки частично компенсируются средствами логической обработки.

Логическая обработка. Принцип «логического вывода» простой: это возможность выводить новые данные из знаний, которые уже есть. В математическом смысле выполнение запроса является одной из форм логического вывода (например, возможность вывести из массы данных некоторый результат поиска). Логический вывод является одним из ведущих принципов Semantic Web, так как он разрешает легко создавать SW-приложения [7; 8].

Для того чтобы Semantic Web стал довольно выразительным и смог помогать людям в разных ситуациях, возникает необходимость построения мощного логического языка, который поддерживает логический вывод. Дискуссии относительно методов, и даже возможности выполнения этой задачи до сих пор ве-

дутся очень активно; обращается внимание на то, что в RDF недостаточны возможности квантификации, и что эта область определена недостаточно хорошо. Большое число приложений обрабатывает информацию, имеющую вид логических выражений. Например, конфигурация файлов определяет доступ для контроля, спецификации профилей показывают необходимость не только структурных, но и логических комбинаций. Необходимо обеспечить словарь такого логического словаря [8].

Системы представления знаний на базе формата для обмена знаниями (Knowledge Interchange Format − KIF) включают не только логический уровень информации, но и кванторы и механизмы вывода. Базовая DLG модель RDF обеспечивает базу для выражения и обмена такими данными, но в будущем необходимо определить общие термы для расширения логического языка. Это может быть использование XML. Поскольку логические системы также делают допущение о полном доступе к информации, это приводит к отказам при работе в вебе, и необходимо сравнение моделей, когда типовые модели представления знаний (ПЗ) представляются для семантической сети.

Многие системы ПЗ работают с первичным узлом представления любой данной концепции, однако семантический веб может иметь много независимых узлов, которые представляют одно и то же. Также системы ПЗ хранят виды структур, чтобы помочь алгоритмам выполнить определенные типы запросов к данным. Предположение, что полная спецификация объектов в документе должна существовать в одном месте, появляется в SGML, Ada, SQL и других объектно-ориентированных системах. В то же время импорт многих интерфейсов в программный модуль также демонстрирует концепцию создания нового модуля, используя независимо созданные языки [7; 8].

Машины для семантического вывода. Для обработки знаний в семантическом вебе необходимы машины вывода, которые получают (выводят) новые знания из существующих. Здесь возможны два подхода – на основе механизмов вывода общей логики и специальных алгоритмов для описания методов решения проблем. На основании первого подхода необходимо различать различные виды языков представления и машин вывода, которые рассмотрим ниже.

Использование полной логики первого (ЛПП) порядка для спецификации требует полного автоматического доказательства теорем. ЛПП является самоопределяемой и поддерживающей логический вывод, но в вычислительном отношении не применимой для большого количества аксиом. Многие проблемы находятся вне библиотеки решений и доказатели теорем пытались их решить, но часто имели неудачу. Это означает, что в веб-пространстве такие программы не могут обрабатывать огромное количество знаний. Кроме того, для этой логики возможна поддержка непротиворечивости, что в вебе невозможно.

Хорновская логика – другой пример ЛПП. Хорновская логика и логика данных (с нулевым функциональным символом) изучается в области дедуктивных БД и логического программирования и имеет эффективные стратегии вычислений. Эти стратегии включают виды поиска снизу вверх и сверху вниз. По-

следняя стратегия лежит в основе языка Пролог, но имеет ограничения для применения в вебе. Поэтому системы со стратегией вывода сверху вниз имеют ограничения относительно декларативной обработки в вебе.

Другой подход от CYC, включающий 1 MB аксиом и использующий ЛПП, организует аксиомы в контексте, поддерживает непротиворечивость для одного контекста и ограничивает выводимость несколькими шагами. Для веб CYC мал. Интерактивный решатель теорем не всегда подходит для агентов, поскольку они не всегда могут ждать пользователя. Альтернативой может быть несколько подмножеств ЛПП, рассматриваемых далее.

Логики высших порядков имеют мощную выразительность по сравнению с обычной логикой. Однако согласно теореме Геделя они не имеют четких механизмов вывода. Также существует две проблемы – синтаксис и семантика высших порядков, которые ограничивают их применение. Логика второго порядка может быть преобразована в логику первого порядка. Под синтаксисом высшего порядка понимается язык, в котором переменным разрешается появляться на местах предикатов и/или функциональных символов. По контрасту семантика высших порядков объявляет семантические структуры, в которых переменные могут быть в диапазоне выше отношений доменов и функций, построенных из индивидуальных доменов. В семантике первого порядка переменные могут быть только в диапазоне индивидуальных доменов или предикатов, но не их множеств.

Исчисление предикатов – пример логики, где синтаксис и семантика первого порядка. Существуют логики, которые имеют синтаксис высшего порядка, но семантику первого, например, логика фреймов. В семантике высшего порядка равенство предикатных символов истинно, если и только если эти символы интерпретируются через одни отношения или функции. Другими словами, логики с семантикой высшего уровня имеют встроенную расширяемую теорию равенства предикатов и функций. Приложения с синтаксисом высшего порядка включают предложения, выражающие истинность о других предложениях.

Описательная логика. Эти логики позволяют специфицировать иерархию терминологии, используя ограниченный набор ЛПП. Они обычно имеют хорошие свойства вычислимости, но сервисы выводимости ограничены классификацией и предположениями. Это означает, что заданные формулы, описывающие классы, классификатор связывает с определенной логикой описания, которая будет определять классы, к которым принадлежит пример. С точки зрения моделей логика описаний принадлежит к мональным логикам.

Выражение предикатной логики с тремя переменными утверждает, что модель ограничена. Примерами систем являются Loom, FaCT, CLASSIC. DLML, определенные DTD. Отметим, что семантическая межобработка в семантическом вебе должна быть получена на базе существующих разработок в RDF. Модель данных RDF разработана на базе инженерии знаний, и язык разработан как ее расширение. Сообщество веб рассматривает XML как важный шаг к семантической интеграции, но лучшая основа для этого – RDF. Сообщество ИИ заинтересовано в использовании техники RDF для веб-пространства.

Rule Interchange Format (RIF) – формат обмена правилами. Цель этого разрабатываемого консорциумом W3C-стандарта – определение формата, который бы разрешил транслировать правила между разными языками правил и благодаря этому обеспечить обмен правилами между системами, основанными на правилах [8]. Системы, основанные на правилах, получили широкое распространение в информационных технологиях. К их числу относятся, например, экспертные системы и системы дедуктивных баз данных. Разработки технологий Semantic Web обеспечивают новую среду использования таких систем. Поэтому консорциум W3C уделяет отдельное внимание этой области. Спецификация RIF может рассматриваться как составная часть комплекса стандартов Semantic Web.

Рабочей группой, организованной при консорциуме для разработки этого стандарта, подготовлен и обсуждается рабочий проект документа, который систематизирует случаи использования RIF и требования к этому языку. Важнейшее требование к создаваемому стандарту – обеспечение возможности его использования не только при текущем состоянии технологий, основанных на правилах, но и его гибкости, достаточной для обеспечения его использования в процессе их эволюции [8]. Рабочий проект документа, который описывает случаи использования, даст возможность определить функциональные требования к RIF и на этой основе разработать адекватные спецификации языка.

Правила вывода новых фактов SWRL. Благодаря дополнению OWL языком RuleML (подмножество Datalog) в виде словаря SWRL (A Semantic Web Rule Language) появилась возможность использовать дизъюнкты Хорна (Horn-like rules) для явного указания способа вывода новых фактов из RDF-утверждений. Пока словарь SWRL находится в стадии стандартизации. Хотя работы над этим уровнем Semantic Web продолжаются, однако в нашем распоряжении есть уже достаточный набор средств для построения Semantic Web: утверждение, цитирование (материализация) в RDF, классы, свойства, области, документирование в схеме RDF, непересекающиеся классы, свойства однозначности и уникальности, типы данных, инверсии, эквивалентности, списки и прочее.

Модель онтологии «Графический пользовательский интерфейс» используется для формирования одного из компонентов структуры интерфейса – модели выразительных средств в рамках онтолого-ориентированного подхода к разработке интерфейса. Онтология графического пользовательского интерфейса, описанная на языке OIL, доступна в Интернете [7].

10.6 Интеллектуальные агенты для техники

В качестве перспективы развитии ИИ в Интернете необходимо отметить основные направления разработок для нужд электронного бизнеса. В первую очередь это технология «интеллектуального агента», создающая эффект постоянного присутствия в сети информационного робота, запрограммированного своим хозяином на сбор и фильтрацию необходимой информации, на поиск

людей и организаций (отвечающих заданным критериям), на проведение определенной стадии переговоров с интеллектуальными агентами других участников экономики и т. п. Данная технология позволит снизить информационную перегрузку участников сетевой экономики, повысить скорость и эффективность процедур установления контактов, проведения переговоров, поддержки соглашений и т. п. К числу атрибутов агента относятся [3; 4]:

− реакция − способность получать из своего окружения информацию;

− деятельность − способность производить действия;

− самостоятельность − способность принимать решения (действовать) на основе собственных знаний; данное свойство представляет особый интерес для разработчиков агентов, потому как функция человека, осуществляющего контроль за работой агента, сведена к минимуму;

− коммуникабельность − способность для достижения цели перемещаться по узлам сети и контактировать с людьми или другими агентами; для осуществления данного свойства необходима разработка специфических языков и онтологий;

− самообучение − является желательным атрибутом для агента.

Наиболее часто используемое определение агента состоит в том, что программный агент это программная сущность, которая функционирует продолжительно и автономно в конкретном окружении, часто – вместе с другими агентами. Агенты могут быть специализированные, они должны уметь общаться с другими агентами с целью обнаружения сервисов, продуктов, информации или других агентов. Сервисы, представленные в сети, могут быть реализованы как агенты.

Возникает проблема создания архитектуры для взаимодействия агентов, где бы агенты могли описывать свои цели с использованием заранее определенных словарей, где возможно было бы производить поиск и подбор необходимых сервисов и информационных ресурсов, а также использовать многие другие возможности. Для того чтобы упростить и стандартизировать процедуру поиска бизнес-партнера и установления с ним контакта, разрабатывается инфраструктура ebXML, которая, с одной стороны, объединяет богатый опыт компаний по сотрудничеству в среде Интернет, а с другой − основана на XML – основополагающем стандарте обмена данными в веб. Подход ebXML имеет все шансы стать универсальным инструментом электронного бизнеса, однако он не способствует широкому использованию программ-роботов, потому как недостаточно выделяет семантику, т. е. передает смысловое содержание документов.

Идея RDF заключается в том, что все предметы могут быть описаны с помощью триплетов: объект-свойство-значение или, в общем виде, субъект-предикат-объект. Субъекты и объекты, принадлежащие веб, единственным образом идентифицированы с помощью URLator). Для идентификации ресурсов, не принадлежащих веб, (это могут быть файлы, email-адреса, физические ресурсы), используется URI, который позволяет создать практически любые ресурсы и определить между ними отношения. Таким образом, RDF позволит сделать информационное пространство веб структурированным для возможности присутствия и работы в нем роботов.

Структура интеллектуального агента. Для реализации предложенных моделей бизнес-процессов в семантическом веб-пространстве предлагается концепция и архитектура интеллектуального агента (ИА). Центральным для подхода является использование онтологий для описания основы представления знаний и придания ясности семантики веб-документов. Онтологии необходимы для формального описания словарей, используемых для описания предметных областей, а также для описания семантической структуры сложных объектов и однородных, распределенных полуструктурированных информационных источников.

ИА имеет архитектуру, включающую четыре элемента: интерфейс для формулирования запросов, информационного агента для сбора необходимых знаний из веба, машины вывода для получения ответа и менеджера БД для кэширования семантики аннотаций. ИА использует семантическую информацию для ответа на запросы. Он обеспечивает ответы в хорошо определенных синтаксисе и семантике, которые могут быть понятыми и обработанными другими агентами и ПО осуществляет доступ к информации физически распределенной и неоднородной в веб-пространстве, а также работает с информацией, которая не имеет форму фактов, но может быть получена из других фактов и базовых знаний. ИА может быть применен для автоматической генерации документов, для извлечения информации из слабоструктурированных текстов, для создания новых источников текста.

ИА включает четыре базовые машины. Машина запросов (МЗ) получает запросы и выдает ответы на основе БД, которая наполняется инфоагентом и машиной вывода. Информационная машина (ИМ) отвечает за сбор фактических знаний из веба, используя различные стили метааннотаций, прямые аннотации. Машина вывода (МВ) использует факты и онтологии для получения дополнительных фактических знаний, неявно заданных. Менеджер БД − основа системы, получает факты от ИА, обменивается ими с МВ и выдает факты в МЗ. Запросы выполняются не только на основе источников, к которым имеется доступ, но динамической БД. К прямым фактам из источников добавляются правила для получения дополнительной информации.

Онтологии играют обобщенный структурный принцип. ИА использует их для извлечения знаний, МВ – для получения фактов, МБД – для структурирования БД. Основной язык – язык представлений, основанный на языке фреймов. Язык запросов определяется как подмножество языка представлений. Более сложное выражение может быть построено на основании элементарных выражений, соединенных знаками логических операций. Поскольку для пользователя не удобно использовать для запросов логический язык, предложена структура языка запросов с табулярным интерфейсом. Используя этот интерфейс, автоматически обнаруживается запрос в текстовой форме и выдает ответ. Информационный агент извлекает фактические знания из источников.

Структурирование информации для ИА. Для работы ИА по обработке семантически размеченной информации предлагаются четыре возможности. Первая − в HTML вводятся небольшие расширения путем интегрирования семантической аннотации в документ, получая HTML-A. Из веба собираются

нужные страницы, выбираются аннотации, которые затем разбираются. Вторая – структурирование информационных источников, которые не имеют аннотаций. Показано, что возможно использовать структуру и регулярность веб-источников для извлечения семантических знаний.

Третья – ИА может использовать RDF-аннотации. RDF обеспечивает средства для добавления семантики в документ без представления внутренней структуры документа. МВ может обрабатывать RDF-описание. Поэтому в состав ИА будет введен анализатор для транслирования RDF-описания в триплеты, которые вставляются в БД.

Четвертой возможностью является использование языка XML. Во многих случаях теги, описанные DTD, могут иметь семантику, используемую для информационного извлечения. Эта информация может быть обработана ИА для получения ответа. XML разрешает определение новых тегов и обеспечивает семантическую информацию определением структуры документа. DTD определяет структуру дерева при описании документа, различные уровни которого имеют теги, обеспечивающие семантику элементарных частей DTD, является средством для описания онтологий. Ограничение ИА заключается в том, что он работает со статическими веб-объектами, т. е. не может обрабатывать скриптовые средства, которые динамически генерируют информацию.

МВ получает факты, собранные веб-сборщиком вместе с терминологией и аксиомами онтологии, на основании которых генерируется ответ на запрос. Для этого выполняется следующее. Сначала логика фрейма транслируется в хорновскую логику. На второй стадии используется техника дедуктивных БД. Адаптирована известная модель семантики, которая вычисляется на основе развития динамической фильтрации. Хорошо основанные семантики, в общем трактуемые семантики для закрытого мира используют рассуждения с отрицанием.

Выбран подход закрытого мира, который означает, что все ответы на запросы корректны в аспекте полноты известных фактов. Другие подходы используют отрицание веба. Они доказывают, что он имеет открытую природу и его просмотр бесконечен. В этом случае модель рассуждения закрытого мира неприемлема. БД динамически расширяется по требованию во время запроса подкачкой новых страниц к известным.

В каждый момент времени при запросе происходит загрузка страниц из веба, что при невысокой скорости снижает производительность. Из-за открытости вычислений необходимо ввести ограничения, когда машина вывода должна остановить поток информации из веба. Примеры для открытой семантики используют связи, определенные для страниц при загрузки новых на время запроса.

Невозможны декларативные запросы для этой семантики – истинные на одном шаге вывода, они могу быть ложными на другом. Эта семантика работоспособна, когда имеются онлайновые запросы и область действия страниц для запроса неизвестна на время запроса и необходимо динамически добавлять страницы в БД. В данном случае используется подмножество веб-пространства, захваченное информационным агентом и применяется концепция закрытого мира для этого фрагмента.

При проектировании ИА принято важное решение − разделение машины просмотра и машины вывода. Первая периодически собирает информацию из веба и кэширует ее. Вторая использует кэш при ответе на запросы. Разделение вывода и сбора фактов сделано для эффективного рассуждения. Запрос реализуется с помощью индексирования кэша, но не извлечения страниц из веба. ИА уточняет архитектуру введением второго разделения – запросов и вывода. Машина вывода работает на заднем плане, получая факты из БД, выводит новые и записывает их обратно. Машина запросов не работает напрямую с выводом, а получает ответ из БД. Когда вывод ограничен во времени, он может выполняться на заднем плане независимо от времени. Использование техники БД для интерфейса запроса и ее базовые факты обеспечивает средства живучести.

10.7 Веб-сервисы для интеллектуализации бизнес-процессов

Понятие веб-сервисов. Веб-сервисы преобразуют XML-документы в ИТ-системах. Веб-сервисы − это XML-приложения, осуществляющие связывание данных с программами, объектами, базами данных либо с деловыми операциями целиком. Между веб-сервисом и программой осуществляется обмен XML-документами, оформленными в виде сообщений. Стандарты веб-сервисов определяют формат таких сообщений, интерфейс, которому передается сообщение, правила привязки содержания сообщения к реализующему сервис приложению и обратно, а также механизмы публикации и поиска интерфейсов [5].

Они могут использоваться для обращения к таким интернет-приложениям, как система предварительных заказов или контроля выполнения заказов. Веб-сервисы пригодны для B2B-интеграции (business-to-business), замыкая приложения, выполняемые различными организациями, в один производственный процесс. Веб-сервисы также могут решать более широкую проблему интеграции приложений предприятия (Enterprise Application Integration, EAI), осуществляя связь нескольких приложений одного предприятия с другими приложениями, размещенными как «до», так и «после» брандмауэра.

Как видно из рисунка 10.1, веб-сервисы представляют собой оболочку, обеспечивающую стандартный способ взаимодействия с прикладными программными средами, такими как системы управления базами данных, как .NET, J2EE (Java2 Platform, Enterprise Edition), CORBA (Common Object Request Broker Architecture), посредники пакетов планирования ресурсов предприятия (Enterprise Resource Planning, ERP), брокеров интеграции [5; 9].

Интерфейсы веб-сервисов получают из сетевой среды стандартные XML-сообщения, преобразуют XML-данные в формат, «понимаемый» конкретной прикладной программной системой, и отправляют ответное сообщение. Программная реализация веб-сервисов (базовое программное обеспечение, нижний уровень) может быть создана на любом языке программирования с использованием любой операционной системы и любого middleware [9].

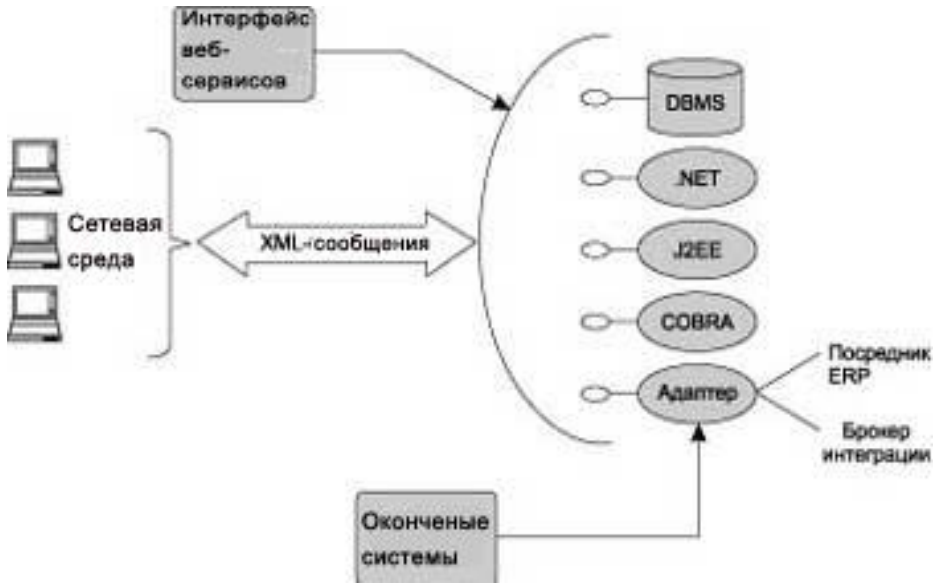

Рисунок 10.1. Веб-сервисы взаимодействуют с прикладными системами

Поиск информации. Большинство сервисов вызываются по Сети посредством ввода данных в HTML-формы и отправки этих данных сервису путем добавления их в строку унифицированного указателя информационного ресурса (Uniform Resource Locator, URL).

Этот пример иллюстрирует простоту веб-взаимодействия (например, поиска, покупки акций или запроса маршрута движения), где параметры и ключевые слова внедряются непосредственно в URL. В данном случае представлен простой запрос поиска skate boots в строке обращения к поисковой машине Google. Ключевое слово *search* представляет сервис, к которому будет осуществлено обращение, а параметр Skate+boots является строкой поиска, которая была введена в HTML-форме на странице веб-сайта Google. Сервис поиска Google передаст этот запрос к различным поисковым машинам, которые вернут список URL для страниц, на которых имеется соответствие параметру поиска Skate+boots. Данный малоэффективный способ поиска в Сети основан на установлении соответствия указанной текстовой строки и индексированных HTML-страниц. XML предоставляет преимущества при передаче данных через Интернет. Теперь предыдущий запрос можно представить в виде XML-документа [9]:

```
<SOAP-ENV:Body>
 <s:SearchRequest
   xmlns:s=«www.xmlbus.com/SearchService»>
   <pl>Skate</pl>
   <p2>boots</p2>
   <p3>size 7.5</p3>
 </s:SearchRequest>
</SOAP-ENV:Body>
```

Отправка запроса в виде XML-документа имеет следующие преимущества: возможность определения типов данных и структур, большую гибкость и расширяемость. XML может представлять структурированные данные или дан-

ные определенного типа (например, допустимо указывать значение поля size (размер) как в виде строки цифр, так и в форме числа с плавающей точкой) и содержать больший объем информации, чем это допускает URL.

Данный пример представлен в форме SOAP-сообщения (Simple Object Access Protocol) − стандартной формы обмена XML-сообщениями − одной из технологий, лежащих в основе веб-сервисов. В SOAP-сообщении имя запрашиваемого сервиса и входные параметры представлены в виде отдельных XML-элементов. Рассматриваемый пример также иллюстрирует использование пространства имен XML (xmlns:), еще одного важного элемента веб-сервисов.

Автоматизация работы в Интернете. Следующее поколение глобальной сети будет основано на программно-ориентированных взаимодействиях. веб-сервисы предполагают использовать созданные для взаимодействия людей глобальные сети совершенно в иных целях. Программно-ориентированные взаимодействия будут автоматически выполнять операции, которые ранее обязательно требовали «ручного» вмешательства [9]:

− поиск и покупка товаров и услуг по самой выгодной цене;

− согласование заказов авиабилетов и мест в ресторане на определенную дату (планирование путешествий);

− оптимизация коммерческих операций закупки товаров, выписки счетов и доставки.

Веб-сервисы являются не только интерфейсом объектов, программ, связующего программного обеспечения и баз данных для доступа по Сети. Объединение ряда веб-сервисов позволяет осуществлять новые типы взаимодействия [9].

На рисунке 10.2 представлен порядок осуществления взаимодействия веб-сервисов с карманным компьютером, подключенным по беспроводному каналу к процессору веб-сервисов. Процессор веб-сервисов получает запросы от функции «календарь» карманного компьютера и находит веб-сервисы, связанные с расширенной функцией календаря, такой как заказ столика. Для завершения планирования путешествия после успешного резервирования места в ресторане процессор веб-сервисов контактирует с веб-сервисами бронирования авиабилетов и номеров в гостиницах. На рисунке 10.3 показано, как веб-сервисы могут изменить способ построения и использования коммерческого приложения [9].

Заинтересованный в закупке коньков оптовый покупатель вводит запрос с помощью собственной локальной системы управления складскими запасами, доступной компьютерам магазинов как веб-сервис. Локальная система управления складскими запасами контактирует через Интернет с веб-сервисом производителя и посылает ему заказ на определенное число коньков, основываясь на объеме имеющихся складских помещений и учитывая наиболее ходовые размеры ботинок.

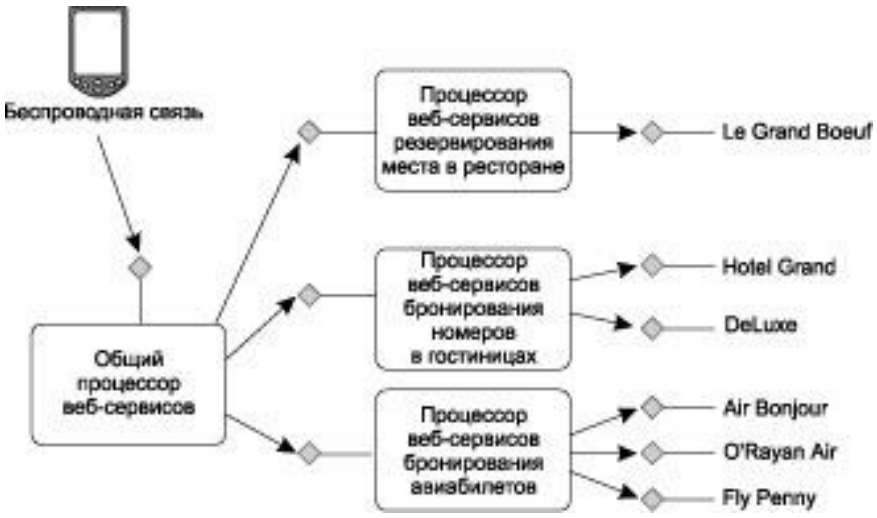

Рисунок 10.2 – Приложения веб-сервисов для путешествий

Система ввода заказов компании Skateboots Company состоит из нескольких сервисов, включая специальную часть, которая отвечает уникальным аспектам ее продукции, и несколько торговых компонентов, выполняющих стандартные функции, например, проверку полномочий пользователя, проверку кредитоспособности и отслеживание счета. Все эти сервисы предоставляются другими компаниями, специализирующимися на оказании таких услуг через Интернет [9].

Создание использующих веб-сервисы коммерческих приложений влечет за собой необходимость установления между этими сервисами соответствующих взаимоотношений, что реализуется с помощью любой комбинации языков программирования, операционных систем, пакетов программ, размещенных «до» или «после» брандмауэра. (Это также является способом решения сложных EAI-проблем.) Установление необходимых отношений (или потока управления) связанных веб-сервисов также приводит к автоматизации соответствующих бизнес-процессов и процедур [9].

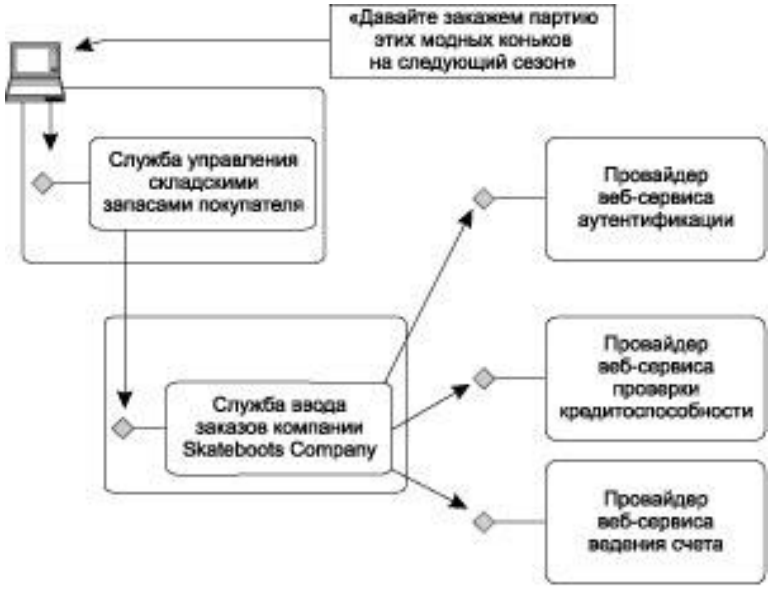

Рисунок 10.3 – Веб-сервисы для службы ввода заказов Skateboots Company

Взаимодействие. На простейшем уровне веб-сервисы могут восприниматься как интернет-ориентированные текстовые брокеры интеграции. Любые данные могут преобразовываться в ASCII-текст и обратно, и этот подход в течение долгого времени был общим знаменателем для систем графического вывода и систем управления базами данных. Ориентированные на использование текста системы также лежат в основе успешного развития Интернета, на котором базируется дополнительная абстракция веб-сервисов. Любой компьютер или операционная система может поддерживать HTML, браузеры и веб-сервисы; и при получении по сети файлов им совершенно безразлично и даже неизвестно, с каким типом прикладной системы они взаимодействуют [5; 9].

То же самое можно сказать и о веб-сервисах, которые зачастую вызывают замешательство, когда разработчики традиционного, устоявшегося компьютерного окружения пытаются воспринимать веб-сервисы как отдельный тип распределенной программной среды, такой как CORBA, J2EE или .NET. Поскольку веб-сервисы гораздо более абстрактны − как посредники (преобразователи), а не как интерфейсы − так и будет продолжаться, пока они не будут приведены в соответствие с общими описаниями и соглашениями.

Веб-сервисы поддерживают несколько парадигм обмена сообщениями. Уровень абстракции, на котором оперируют веб-сервисы, подразумевает такие стили взаимодействия, как эмуляцию удаленного вызова процедуры (Remote Procedure Call, RPC), асинхронный обмен сообщениями, однонаправленную передачу сообщений, широковещание и публикацию/подписку. Основные СУБД, такие как Oracle, SQL Server и DB2, поддерживают анализ XML и службы преобразования, обеспечивая непосредственное взаимодействие между веб-сервисами и СУБД [9].

Производители связующего программного обеспечения предоставляют возможность привязки веб-сервисов к своим программным системам (серверам приложений и брокерам интеграции). Для пользователя взаимодействие с веб-сервисами может проявляться в интерактивной или пакетной форме, поддерживающей синхронную и асинхронную модели связи; а также как пользовательский интерфейс, написанный с использованием Java, Visual Basic, офисных приложений, браузеров или «толстых» клиентов СУБД.

Веб-сервисы выполняют RPC- и документно-ориентированное взаимодействия. Стандарты и технологии веб-сервисов обычно подразумевают два основных типа моделей взаимодействия приложений [5; 9]:

− удаленный вызов процедуры (онлайновая);

− документно-ориентированный (пакетная).

RPC-ориентированные взаимодействия удобны для краткого обмена данными. В RPC-ориентированном взаимодействии запросы веб-сервисов приобретают форму вызова метода или процедуры с соответствующими входными или выходными параметрами. В отличие от документно-ориентированного взаимодействия, RPC-ориентированное взаимодействие производит отправку документа, специально отформатированного для передачи в отдельную логическую программу или базу данных (рисунок 10.4) [9].

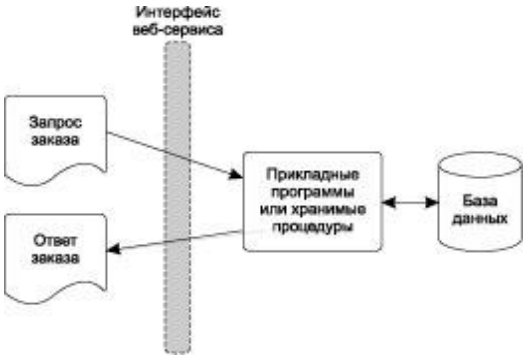

Рисунок 10.4 – Веб-сервисы поддерживают интерактивный заказ

Если поставка коньков невозможна, будет получено сообщение об отсрочке выполнения заказа или о полном отказе от его выполнения. В отличие от документно-ориентированного стиля взаимодействия, запрос и ответ моделируются как синхронные сообщения, то есть приложение, посылающее сообщение, ждет реакции на него.

Документно-ориентированные взаимодействия удобны для обмена большими объемами данных. Запросы веб-сервиса имеют форму завершенного XML-документа, предназначенного для обработки целиком. Например, веб-сервис, который представляет заказ на поставку (оптовый заказ на поставку коньков), должен предъявлять производителю полную форму заказа (рисунок 10.5). Производитель высылает заказчику подтверждение, свидетельствующее о том, что заказ принят и будет выполнен в соответствии с последовательностью выполнения бизнес-процесса [9].

Последовательность выполнения может содержать такие этапы, как проверка в базе данных предыдущих заказов данного оптового покупателя на предмет исчерпания кредитного лимита, согласованных объемов или графика поставок по данному заказу. Необходимые взаимодействия определяются в договоре торговых партнеров. Документно-ориентированные взаимодействия предполагают, что использующие веб-сервисы стороны согласовали порядок оформления общих документов, таких как заказ на приобретение, счет за доставку или общий счет. Эти стороны обычно идентифицируются как «торговые партнеры».

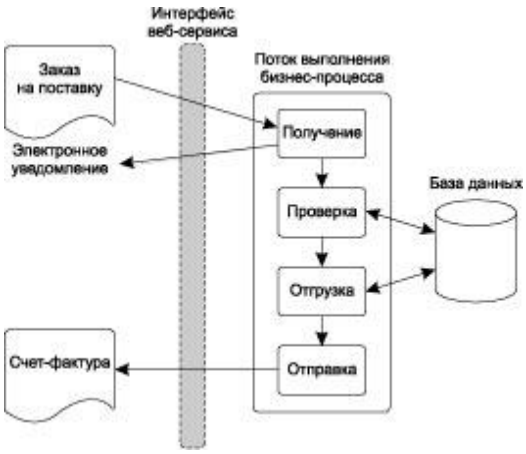

Рисунок 10.5 – Веб-сервис обрабатывает полный заказ на поставку

Торговые партнеры согласовывают общий поток выполнения процесса или модель взаимодействия при обмене документами, например, оговаривают необходимость подтверждения квитанции заказа на приобретение, передачу специальной информации о состоянии в ответ на запрос заказа или отправку сигнала оповещения по электронной почте после отгрузки заказа. В ходе реализации бизнес-процесса необходим обмен полными документами.

Технология веб-сервисов [9]. Порядок описания, поиска и взаимодействия веб-сервисов друг с другом определяют стандарты. Взаимодействующие через Интернет программы должны уметь обнаруживать друг друга, находить информацию, позволяющую им осуществить связь, понимать, какая модель контактирования должна быть применена (простая, типа «запрос/ответ» или более сложная последовательность), и договариваться об использовании таких услуг, как защита информации, подтверждение передачи сообщений и составление сделок. Для транспортировки и преобразования данных в программы и обратно веб-сервисы требуют использования нескольких смежных XML-технологий.

Язык XML (Extensible Markup Language) − фундамент, на котором строятся веб-сервисы. Он предоставляет язык определения данных и порядок их обработки. XML представляет семейство связанных спецификаций, публикуемых и поддерживаемых интернет-консорциумом (World Wide Web Consortium, W3C) и другими организациями.

WSDL (Web Services Description Language) − технология, основанная на XML, определяющая интерфейсы веб-сервисов, типы данных и сообщений, а также модели взаимодействия и протоколы связывания.

10.8 Применение веб-сервисов

Интеллектуальные реализации. Веб-сервисы исполняют роль, которая ранее отводилась реляционным базам данных, языкам программирования 4-го поколения и начальному искусственному интеллекту. Они представляют новый уровень, а не фундаментальное изменение, которое приходит на смену существующей компьютерной инфраструктуре. Этот новый технологический уровень выполняет новую функцию, обеспечивает механизм интеграции, определенный на более высоком уровне абстракции [4; 5; 9].

Веб-сервисы являются дополнением и не конфликтуют с существующими приложениями, программами и базами данных. Разработка приложений по-прежнему требует знаний Java, VB и C#. Все, что является новым, − это способ преобразования данных для передачи в приложение и обратно на основе стандартных форматов XML-данных и протоколов, что позволяет обеспечить новый уровень интеграции и взаимодействия.

При проектировании и разработке новых программ и баз данных разработчики могут учитывать веб-сервисы, но эти программы и базы данных все равно потребуют использования упаковщиков веб-сервисов. Веб-сервисы сами по себе не являются исполняемыми; они полагаются на исполняемые программы, написанные с помощью языков программирования или сценариев [5; 9]:

− протоколе SOAP (Simple Object Access Protocol) − совокупность XML-технологий, определяющих «конверт» для связи веб-сервисов (привязанный к HTTP и другим транспортам), − специализированный формат для передачи XML-документов по ГС, а также соглашения RPC-взаимодействий;

− технологии UDDI (Universal Description, Discovery and Integration) − реестр веб-сервисов и механизм поиска. Он используется для хранения и упорядочения деловой информации, а также для нахождения указателей на интерфейсы веб-сервисов.

Использование для ЭБ. Основные стандарты веб-сервисов используются согласованно. После обнаружения WSDL в UDDI или другом месте генерируется SOAP-сообщение для отправки на удаленный сайт. Как видно из рисунка 10.6, при предоставлении документа по адресу веб-сервиса программа использует XML-схему определенного типа (такую как WSDL), позволяющую преобразовать данные из ее входного источника и на основе того же WSDL-файла создать экземпляр XML-документа в формате, согласованном с целевым веб-сервисом. WSDL-файл используется для определения как входного, так и выходного преобразования данных [9].

Сначала программист создает WSDL-файл, который описывает веб-сервисы, поддерживаемые SOAP-процессором (1), и с помощью API UDDI регистрирует информацию в хранилище (2). После отправки предприятием этих сведений (с другой контактной информацией) элемент реестра будет содержать URL, указывающий на WSDL-файл сайта SOAP-сервера или на другой файл XML-схемы, описывающий данный веб-сервис [5; 9].

После этого SOAP-процессор другого предприятия запрашивает реестр (3) для получения WSDL или другой схемы (4). Для отправки конкретной операции по определенному протоколу (6) клиент может сгенерировать соответствующее сообщение (5).

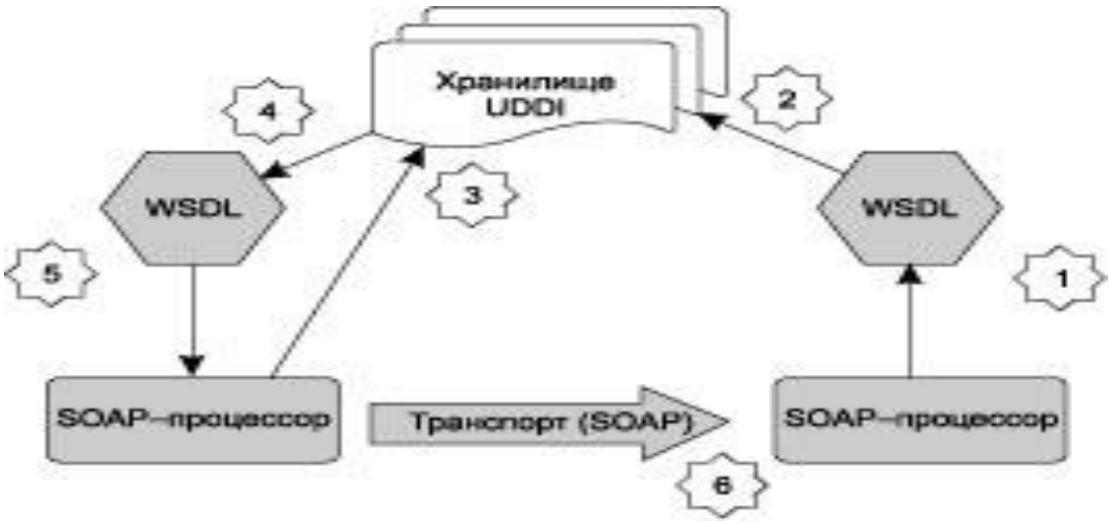

Рисунок 10.6 – Создание и использование веб-сервисов

SOAP-процессор отправляющего компьютера преобразует данные из собственного формата в тип данных, предопределенный XML-схемой, содержащейся в WSDL-файле, на основе таблиц преобразования для текстов и других данных. Таблицы преобразования «связывают» собственные типы данных с соответствующими конкретной XML-схеме. (Стандартное преобразование типов широко используется в Java, Visual Basic, CORBA и других известных системах). SOAP-процессор получающего компьютера выполняет обратное преобразование данных из типов XML-схемы в собственные типы данных [9].

Файлы описаний веб-сервисов регистрируются с помощью URL, указывающий на IP-адрес веб-ресурса. Схемы веб-сервисов являются одной из форм веб-ресурса, они содержатся в доступных через Интернет файлах и к ним применим тот же механизм, что используется при загрузке HTML-файлов. Главное отличие между загрузкой HTML-файла и обращением к ресурсу веб-сервиса заключается в том, что веб-сервис оперирует XML-документами, а не HTML-документами и опирается на соответствующие технологии, такие как использование схем, преобразование, проверка подлинности, что и обеспечивает поддержку удаленного соединения приложений.

Для проверки достоверности сообщений веб-сервисы используют XML-схемы [9]. После получения документа реализация Веб-сервиса сначала должна проанализировать XML-сообщение и удостовериться в корректности данных, выполнить проверку качества услуг (Quality-of-Service), такую как проверка политики безопасности или соглашений торговых партнеров, а затем произвести последовательность связанных с данным документом коммерческих операций.

Веб-сервис, доступный по данному интернет-адресу, идентифицируется с помощью публичного WSDL-файла, который может быть загружен на отправляющий компьютер и использоваться при генерации сообщения. Компания Skateboots Company также осуществляет отправку в общедоступный каталог UDDI-листинга, позволяющего клиентам находить компанию с помощью технологии UDDI [9].

Размещенные по адресу skateboots.com программы представляют собой прослушивающий HTTP-процесс, связанный с соответствующими веб-сервисами для распознавания XML-сообщений, определенных в данном формате. Эти программы включают в себя XML-анализаторы и преобразователи. Кроме того, они осуществляют конвертацию данных SOAP-сообщения в форматы, необходимые для системы ввода заказов компании Skateboots Company [9].

10.9 Дополнительные технологии в веб-сервисах

Ядро технологий веб-сервисов (SOAP, WSDL и UDDI) полезно для наведения мостов с несопоставимыми технологиями и предоставления документов в общий поток делового процесса. Однако для того чтобы быть нужными для большего количества типов приложений и соответствовать представлению о веб-сервисах как строительных блоках для работы через Интернет, технологии веб-сервисов должны обладать рядом дополнительных характеристик, функций и сервисов качества обслуживания [9].

На современном этапе деятельность по развитию веб-сервисов в качестве более полезного базиса очень похожа на эволюцию технологии Common Object Request Broker Architecture **(**CORBA). Группа Object Management Group (OMG) в 1990-х годах взяла на себя обязательства всесторонне определить архитектуру такого программного обеспечения. В результате открытых совместных усилий мы получили мощный набор спецификаций, формализующих осуществление сделок, асинхронный обмен сообщениями, требования к безопасности, отказоустойчивости и т. д. Подобные усилия в отношении веб-сервисов были инициированы W3C и также привели к разработке аналогичной архитектуры [5; 9].

Дополнительные технологии могут стать частью стандарта. В мире веб-сервисов основные поставщики промышленного программного обеспечения уже достигли примирения по стержневым стандартам. Microsoft, IBM, Sun Microsystems, BEA Systems, Oracle, IONA и другие компании договорились о реализации SOAP, WSDL и UDDI, хотя сохранилась некоторая разница во мнениях о роли ebXML-регистрации. Однако нефундаментальные стандарты зачастую оспариваются. К таким спорам относятся разногласия Microsoft и IBM по вопросу определения последовательности действий, то есть продолжается противоборство XLANG и WSFL (Web Services Flow Language) [9]. Кроме того, остаются конкурирующие предложения по реализации контекста безопасности.

Дополнительные технологии в основном сконцентрированы в следующих ключевых направлениях: безопасность; поток управления процессом; транзакции; передача сообщений. Наиболее важными из дополнительных технологий веб-сервисов являются технологии безопасности.

Безопасность подразумевает конфиденциальность и целостность данных веб-сервисов. Никто, кроме конечного получателя данных, не должен иметь возможность изучать или изменять содержание сообщения. Вопрос безопасности также важен в плане контроля доступа к веб-сервисам, особенно при совместном использовании нескольких веб-сервисов. Для авторизации и аутентификации предложено несколько стандартов (например, Security Authorization Markup Language, SAML). Также имеются стандарты для шифрования открытым ключом (XML Key Management Specification, XKMS). Конечно же, основой всей безопасности Интернета является протокол защищенных сокетов (Secure Socket Layer, SSL), а для HTTP-протоколов − HTTPS (secure HTTP), обеспечивающие базовый уровень безопасности [5; 9].

Помимо HTTPS, брандмауэров, SAML, XKMS, использования цифровых подписей и XML-кодирования компания Microsoft предложила системы WS-License (управления удостоверениями безопасности) и WS-Security (распространение удостоверений безопасности, относящихся к взаимодействию с веб-сервисами).

Информационные потоки автоматизируют выполнение бизнес-процесса. Поток процесса является важнейшим понятием при автоматизации взаимодействий бизнес-процесса через Сеть и в пределах предприятия. Поток процесса также зачастую называется «оркестровкой», поскольку он определяет взаимоотношения в цепочке взаимодействий, необходимых для достижения постав-

ленной цели, такой как заполнение заказа на поставку, выполнение бронирования при подготовке путешествия или составление бизнес-плана. Поток смоделирован как серия шагов, определенных для конкретного бизнес-процесса. Последовательность шагов создает совокупность функций, для которых определяется интерфейс веб-сервиса.

Транзакции. Происходит переопределение механизма транзакций для Сети. В области автоматизации коммерческих операций давно используется система контроля, гарантирующая, что, несмотря на программные или аппаратные сбои, платформы, на которых осуществляются транзакции, обеспечивают согласованные результаты при выполнении последовательности связанных с данными операций [9].

Однако традиционные протоколы и технологии не могут быть непосредственно применены к Сети, поскольку они предназначены для работы в средах с тесным взаимодействием, где возможно наличие блокировок записей баз данных при ожидании уведомления о результатах транзакции и в которых для автоматического обнаружения нарушения связи доступны протоколы, ориентированные на соединение. Для разрешения этой проблемы веб-сервисов предназначен предложенный OASIS протокол BTP (Business Transaction Protocol), который посредством определения слабо связанного протокола гарантирует корректное распространение и доступность для совместного использования результатов множества взаимодействий веб-сервисов.

Механизм надежной *передачи сообщений.* Протоколы передачи сообщений реализуют модель связи, определенную для таких взаимодействий веб-сервиса, как асинхронное однонаправленное взаимодействие, запрос/ответ, широковещание и обычное равноправное взаимодействие. Дополнительные технологии веб-сервисов также могут зависеть от уровня доставки сообщений конкретного сервиса качества обслуживания, такого, например, как надежная (гарантированная) доставка, распространение контекстов безопасности и контекстов транзакций, а также верная маршрутизация сообщений по определенному пути, на котором имеется один и более промежуточных узлов. Компания IBM в связи с этим представила надежный протокол HTTP (HTTPR).

Смежные разработки. IBM сотрудничает с Microsoft в области разработки WS-Inspection, предназначенной для поиска информации о веб-сервисах, имеющихся у конкретного адресата сообщения. Кроме того, Microsoft предложила системы WS-Referral и WS-Routing для определения пути конкретного сообщения веб-сервиса, включая любое число промежуточных узлов, позволяющие тем самым определить, как перенаправлялось сообщение при прохождении по указанному маршруту [9].

BEEP − ориентированный на соединение протокол. Разработанный протокол Blocks Extensible Exchange Protocol (BEEP) представляет собой ориентированный на соединение протокол. Уже определена привязка SOAP для BEEP, и в данном случае SOAP-сообщения наследуют от BEEP дополнительные сервисы качества обслуживания, поддерживая контекст сеанса на узлах отправителя и получателя. Контекст может быть использован для соотнесения множества

сообщений с большим модулем пересылки, а также для соотнесения множества сообщений, поступающих от одного источника или предназначенных одному получателю. Контексты безопасности и транзакций также могут быть связаны с соединением.

С веб-сервисами связано множество технологий и стандартов. Другие значимые стандарты и технологии представлены следующими организациями [8; 9]:

– OASIS − обработка входящих ebXML и другие предложения XML, такие как BTP, SAML, UDDI и WS-Security;

– RosettaNet − влияла на концепции веб-сервисов, разрабатываемых группой производителей электроники для осуществления взаимодействия типа B2B через Интернет;

– UserLand − разработчик XML-RPC − предшественника SOAP;

– OAGI (Open Applications Group, Inc.) − определение канонических форматов XML-документа для коммерции и промышленности.

Работа этих и других групп зачастую концентрировалась на содействии принятию XML для специальных коммерческих целей, таких как использование базовых стандартов для определения форматов документов и протоколов, которые должны использоваться в электронной промышленности, финансовой сфере, сфере здравоохранения и других областях. Поскольку веб-сервисы основаны на языке XML, весьма значимой становится деятельность почти всех органов и консорциумов, занимающихся стандартами и совершенствованием XML-технологий. Результатом этой работы являются BTP и SAML, появившиеся в ходе развития веб-сервисов и ставшие кандидатами на утверждение консорциумом W3C.

В рамках инициативы WSI разрабатываются примеры прикладных решений, предложения и дополнительные требования, призванные гарантировать совместимость решений разных поставщиков. Это сулит широкие возможности по интеграции различных информационных систем в рамках единого согласованного набора спецификаций.

Во многих случаях интеграция информационных ресурсов требует комбинирования обращений более чем к одному веб-сервису для реализации пользовательского запроса. Веб-сервисы должны иметь возможность поддерживать взаимодействие с другими приложениями в дополнение к стандартным процедурам обработки данных. Более того, процесс предоставления агрегированной распределенной информации может включать в себя разбиение на набор взаимосвязанных этапов обработки данных, взаимодействие ряда веб-сервисов, вмешательство людей в процесс обработки пользовательских запросов и другие элементы прикладной логики. Поэтому процесс сбора и интеграции гетерогенных данных может представлять собой логически сложную композицию обращений к хранилищам информационных сущностей посредством интерфейсов веб-сервисов – определять автоматизированный поток обработки данных.

Для описания композиций веб-сервисов на данный момент различными ассоциациями предлагается ряд стандартов. Среди них можно отметить следующие языки описания автоматизированных потоков работ, участниками которых являются веб-сервисы [8; 9]:

– WSFL (Web Services Flow Language) – позволяет определять композиции веб-сервисов в виде графовой модели рабочего процесса;

– BPML (Business Process Modeling Language) – определяет блочную модель композиции веб-сервисов;

– BPEL4WS (Business Process Execution Language For Web-Services) – представляет собой гибрид блочной и графовой моделей описания взаимодействий веб-сервисов.

Эти языки позволяют описывать композиции веб-сервисов, что позволяет определять сложные, распределенные процессы по извлечению, обработке и интеграции информации. Для решения таких сложных распределенных задач особенно хорошо подходит мультиагентная технология.

Для выполнения конкретных задач веб-сервисы должны обмениваться сообщениями, сообщать информацию о себе и предоставляемых услугах в виде, удобном как для машинной обработки, так и доступном для понимания человеком. Для решения этой задачи консорциумом были предложены языки метаописаний сервисов WSDL, а также онтологический язык веб-сервисов OWL-S. В настоящее время консорциумом предложен проект языка моделирования сервисов – Service Modeling Language (SML) [8].

10.10 Перспективы интеллектуализации Интернета

Web-intelligence (WI) предлагает новое направление научных исследований и разработок, подталкивая технологию к манипулированию смыслом данных и создавая распределенный интеллект, который действительно сделает это возможным. WI исследует фундаментальное и практическое влияние, которое искусственный интеллект и развитые информационные технологии будут оказывать на следующее поколение систем, служб и сред, оснащенных возможностями веба. Кратко обсудим ряд аспектов WI [3; 4].

Автономная поддержка веб. Интеллектуальная Сеть функционирует как автономная сущность, автоматически регулируя функции и взаимодействия со связанными веб-сайтами и доступными прикладными службами.

Представление. Агенты интеллектуальной Сети могут использовать язык Problem Solver Markup Language для спецификации своих ролей, установок и связей по отношению к любым другим службам. Также должна обеспечиваться возможность обработки и понимания естественного языка [6; 7].

Самонаведение и самообучение. Помимо семантических знаний, извлекаемых и обрабатываемых путем интеллектуального поиска, агенты интеллектуальной Сети должны также включать динамически создаваемый источник метазнаний, имеющих дело со связями между понятиями, и знания о пространственных и временных ограничениях для планирования использования служб.

Интеллектуальная Сеть персонализирует взаимодействия путем запоминания предпочтений пользователей и учета тем и сайтов, к которым обращался пользователь в течение различных сеансов.

Q-язык. Существует несколько языков описания поведения агентов и протоколов их взаимодействия. Однако, по мнению автора, в контексте веб должны учитываться потребности непрофессиональных разработчиков приложений. Q предназначен для описания сценариев взаимодействия агентов и пользователей на основе внешних ролей агентов. Он обеспечивает интерфейс между компьютерными профессионалами и создателями сценариев. Язык является расширением известного языка Scheme, представляющего собой диалект языка Lisp.

BOD (Behavior-Oriented Design). Предлагается расширить концептуальную основу Semantic Web путем включения поведения как неотъемлемой части этой основы. Такое расширение позволит добиться использования полного потенциала Semantic Web не только в качестве хранителя знаний, но и как нечто, могущее производить действия с этими знаниями. При использовании предлагаемого подхода к построению Semantic Web система выглядит как набор агентов, человекоподобных акторов со своими мнениями, намерениями и возможностями. Для достижения возможности создания нового облика Semantic веб требуется разработать методологию построения агентов, могущих безопасно расширять свое поведение, и развить методы разметки Semantic веб для поддержки этого типа расширения. Предлагается использовать агенто-ориентированный подход к инженерии программного обеспечения, названный авторами BOD (Behavior-Oriented Design). Соответствующий язык разметки базируется на языке DAML-S

Поисковая технология Semantic Web интегрируется с операциями преобразования, автоматическим извлечением информации и семантической поддержкой сопровождения информации и обеспечения пользовательских представлений. Все средства основаны на трехуровневой архитектуре. На нижнем уровне путем задания запросов на естественном языке извлекаются метаданные из структурированных и полуструктурированных документов. На среднем уровне репозиторий аннотированных данных обеспечивает доступ к метаданным, которые используются для поддержки оперативно доступных источников. На верхнем уровне клиенты и провайдеры могут использовать развитые методы для изучения и модификации областей знаний. На трех уровнях для обеспечения структурных и семантических определений документов используются онтологии.

Web-mining. По мере развития методов добычи данных они играют возрастающую роль при решении задач создания WI. Анализируют основные факторы, ограничивающие возможности доступа к веб: отсутствие качественного поиска по ключевым словам, отсутствие эффективного доступа к подключенным к веб-базам данных, отсутствие автоматически конструируемых каталогов, отсутствие примитивов семантически-ориентированных запросов и т. д. Задачи, стоящие перед разработчиками средств Web-mining: добыча данных в поиско-

вых машинах; анализ структур связей в вебе; автоматическая классификация веб-документов; добыча структур и семантического содержания веб-страниц.

Выводы

На основании вышеизложенного и с учетом источников [1–9] заключаем.

1. Первое поколение веба связывается со статическими HTML-страницами, второе поколение − с динамическими, обрабатываемыми автоматически. Третье поколение веб называют содержащим знания, имеющим целью машинную обработку смысла информации, получило название семантического. Направления разработки семантического веба включают специальную разметку веб-информации, представление знаний и ее обработку, использование онтологий и разработку интеллектуальных агентов.

2. В настоящее время есть три основные технологии для представления знаний в Semantic Web: XML, Resource Description Framework (RDF) и OWL (Ontology Web Language). XML позволяет каждому разрабатывать собственные теги − неотображаемые метки вида <zip code> или <alma mater>, которые размечают саму веб-страницу или фрагменты ее текста. Смысловое значение выражается с помощью RDF, который кодирует его в набор троек, причем каждая тройка примерно соответствует подлежащему, сказуемому и дополнению (object, verb, subject) в обычном предложении.

3. Две базы данных могут использовать разные идентификаторы для обозначения одного и того же, например, индекс. Программа, которая сравнивает, дополняет или сопоставляет данные двух баз, должна знать, что эти два понятия используются для обозначения одного и того же. Решение этой проблемы выражается в третьем по счету основном понятии Semantic Web − онтологии, т. е. общему для нескольких источников списку данных.

4. Технология Semantic Web рассчитывает, что все веб-агенты, даже те, которые задумывались для независимой работы, будут «добывать» информацию, «помогая» друг другу. Обычный процесс работы веб-агента подразумевает создание так называемой «цепочки значений»: составляющие информации передаются от одного агента к другому, причем каждый добавляет свое значение так, что формируется конечный готовый информационный продукт, запрашиваемый пользователем.

5. Модель объекта документа Document Object Model (DOM), которая была создана как усилие согласовывать скрипты HTML для обслуживания в браузерах, была расширена включением XML и стала основополагающим прикладным программирующим интерфейсом (API) для многих платформ программного обеспечения сети и структурированных репозиториях данных. Программное обеспечение, основанное на этих платформах, рассматривает RDF не как XML потоки, но как объекты проекта DOM.

6. Для представления знаний (обмена онтологиями) разработан язык OIL, который базируется на развитии языка XML с расширениями, использующими логику описаний (DL), произвольными булевыми выражениями, которые появляются при появлении имени класса. Определение слота может рассматривает-

ся как класс и использоваться в виде выражения: определение класса имеет дополнительное поле, которое определяет его как примитив или не примитив; глобальные определения слота расширены для спецификации суперслотов и связи таких свойств, как транзитивность и симметричность; дополнительные правила, накладываемыми XOL-документами, нетребуются;

7. Для обработки знаний в семантическом вебе необходимы машины вывода, которые получают новые знания из существующих. Здесь возможны два подхода – на основе механизмов вывода общей логики и специальных алгоритмов для описания методов решения проблем. По первому направлению работы ведутся на базе различных логик, но все они имеют те или иные ограничения.

8. Веб-сервисы − это XML-приложения, осуществляющие связывание данных с программами, объектами, базами данных либо с деловыми операциями целиком. Между веб-сервисом и программой осуществляется обмен XML-документами, оформленными в виде сообщений. Стандарты веб-сервисов определяют формат таких сообщений, интерфейс, которому передается сообщение, правила привязки содержания сообщения к реализующему сервис приложению и обратно, а также механизмы публикации и поиска интерфейсов.

9. Веб-сервисы сами по себе не являются исполняемыми; они полагаются на исполняемые программы, написанные с помощью языков программирования или сценариев: протоколе SOAP − совокупность XML-технологий, определяющих «конверт» для связи веб-сервисов (привязанный к HTTP и другим транспортам) технологии UDDI (Universal Description, Discovery and Integration) − реестр веб-сервисов и механизм поиска. Он используется для хранения и упорядочения деловой информации, а также для нахождения указателей на интерфейсы веб-сервисов; WSDL (Web Services Description Language) − технология, основанная на XML, определяющая интерфейсы веб-сервисов, типы данных и сообщений, а также модели взаимодействия и протоколы связывания.

10. Ядро технологий веб-сервисов (SOAP, WSDL и UDDI) полезно для наведения мостов с несопоставимыми технологиями и предоставления документов в общий поток делового процесса. Для того чтобы быть нужными для большего количества типов приложений, технологии веб-сервисов должны обладать рядом характеристик, функций и сервисов качества обслуживания. Дополнительные технологии сконцентрированы в ключевых направлениях: безопасность; поток управления процессом; транзакции; передача сообщений.

11. Технология интеллектуального агента создает эффект постоянного присутствия в сети информационного робота, запрограммированного на сбор и фильтрацию необходимой информации, на поиск людей и организаций (отвечающих заданным критериям), на проведение определенной стадии переговоров с интеллектуальными агентами других участников экономики и т. п. Данная технология позволит снизить информационную перегрузку участников сетевой экономики, повысить скорость и эффективность процедур установления контактов, проведения переговоров, поддержки соглашений и т. п.

12. К числу атрибутов интеллектуального агента относятся: реакция − способность получать из своего окружения информацию; деятельность − способность производить соответствующие действия; самостоятельность − способность принимать решения и действовать на основе собственных знаний; коммуникабельность − способность для достижения цели перемещаться по узлам сети и контактировать с людьми или другими агентами; самообучение.

13. Интеллектуальной агент имеет архитектуру, включающую четыре элемента: интерфейс для формулирования запросов, информационную систему поиска и сбора необходимых знаний из веба, машину вывода для получения возможных ответов и менеджера БД для кэширования семантики аннотаций. ИА использует семантическую информацию для ответа на запросы. Для работы ИА предлагаются четыре возможности. Первая − в HTML вводятся небольшие расширения путем интегрирования семантической аннотации в документ, получая HTML-A. Из веб собираются нужные страницы, выбираются аннотации, которые затем разбираются. Вторая – структурирование информационных источников, которые не имеют аннотаций. Показано, что возможно использовать структуру и регулярность веб-источников для извлечения семантических знаний. Третья – ИАс может использовать RDF аннотации, RDF обеспечивает средства для добавления семантики в документ без представления внутренней структуры документа. МВ может обрабатывать RDF описание. Четвертой возможностью является использование XML.

14. Задачи, стоящие перед исследователями и разработчиками средств категории Web-mining: добыча данных в поисковых машинах; анализ структур связей в вебе; автоматическая классификация веб-документов; добыча структур и семантического содержания веб-страниц и т. д.

Контрольные вопросы

1. Приведите основные технологии для разработки Semantic Web.
2. Поясните смысловое значение знаний (выражается с помощью RDF).
3. В чем заключается понятие онтологии.
4. Какие 4 компонента включает ИА?
5. Поясните атрибуты ИА.
6. Охарактеризуйте использование онтологий в веб-пространстве.
7. Поясните виды логического вывода в веб-пространстве.
8. На чем базируется представление знаний в веб-пространстве?
9. Поясните способы семантической разметки веб-пространства для ИА.
10. Что интегрируется в веб-сервисе для планирования ресторана за границей?
11. На каких компонентах базируются веб-сервисы?
12. Охарактеризуйте 2 вида взаимодействия в веб-сервисах.
13. Поясните 2 шага создания веб-сервиса.
14. Стандарты для авторизации, аутентификации и шифрования в веб-сервисах.
15. Что разработала компания Microsoft для веб-сервисов в области безопасности?
16. Охарактеризуйте перспективы интеллектуализации Интернета.

Литература, используемая в главе

1. Berners-Lee, T. The Semantic Web / T. Berners-Lee, J. Hendler, O. Lassila // Scientific American. – 2001. – P. 28–37.

2. Klyne, G. Resource Description Framework (RDF): Concepts and Abstr. Syntax [Electronic resource] / G. Klyne, J. J. Carroll. – Mode of access: http://www.w3.org/TR/rdf-concepts/. – Date of access: 04.07.2022.

3. Semantic Web organization. [Электронный ресурс]. – Режим доступа: http://www.semanticWeb.org/. – Дата доступа: 15.12.29015.

4. Вишняков, В. А. Интеллектуальные системы в управлении / В. А. Вишняков. – Минск : Изд-во МИУ, 2010. – 364 с.

5. Ньюкомэр, Э. Вэб-сервисы: для профессионалов / Э. Ньюкомэр. – СПб. : Питер, 2003. – 324 с.

6. KSL Interactive Ontology Server [Электронный ресурс]. – Режим доступа: http:// www-ksl-svc.stanford.edu. – Дата доступа: 15.12.29015.

7. Использование семантического Веба в качестве модели [Электронный ресурс]. – Режим доступа: http://elibrary.ru/item.asp?id=23128676. – Дата доступа: 15.12.2022.

8. Semantic Web как новая модель информационного пространства интернет [Электронный ресурс]. – Режим доступа: https://docplayer.ru/58949420-Semantic-web-kak-novaya-model-informacionnogo-prostranstva-internet.html. – Дата доступа: 15.12.2022.

11 ТЕХНОЛОГИИ ОБЛАЧНЫХ ВЫЧИСЛЕНИЙ ДЛЯ ИНТЕЛЛЕКТУАЛЬНОГО УПРАВЛЕНИЯ

11.1 Понятия облачных вычислений

Под облачными вычислениями (ОВ) понимают возможность получения требуемых вычислительных ресурсов по запросу из сети, причем пользователю не важны детали реализации этого механизма и он получает из так называемого «облака» все необходимое [1]. Типовым примером могут служить поисковые системы, интерфейс которых очень прост, но в то же время они предоставляют пользователям большие вычислительные ресурсы для поиска нужной информации. Мощные вычислительные центры (дата-центры) не только позволяют хранить и обрабатывать внутри себя разнообразные данные, но и дают возможности для создания собственных ИТ-ресурсов (инфраструктур, платформ), позволяя начинающим компаниям не тратить ресурсы на создание собственной ИТ-инфраструктуры [1; 2].

«Облако» является новой бизнес-моделью для получения информационных услуг. Эта модель снижает капитальные затраты почти до нуля, понижает оперативные затраты за счет меньшего штата обслуживания. Эта модель позволяет ИТ-отделам компаний сосредоточиться на стратегических проектах, а не на рутинных задачах управления собственным центром обработки данных. ОВ – это не только сетевая технологическая инновация, но и способ создания новых бизнес-моделей, когда у небольших производителей ИТ-продуктов появляется возможность быстрого предложения рынку своих услуг и мало затратного способа воплощения своих бизнес-идей. Поддержка облачных вычислений в сочетании с инвестициями в открывающихся компаниях создают развивающуюся систему инновационных производств [1; 2].

ОВ являются альтернативным ответом на системную специализацию и усиление роли аутсорсинга в ИТ. Таким образом, переход к облачным вычислениям означает аутсорсинг процессов управления ИТ-инфраструктурой профессиональными внешними поставщиками. Многие из современных поставщиков ИТ-решений на базе облачных вычислений предоставляют возможность не только использовать существующие облачные платформы, но и создавать собственные, отвечающие технологическим и юридическим требованиям заказчиков, т. е. выполняют модифицированный ИТ-аутсорсинг [1; 2].

Технология ОВ работает следующим образом: вместо приобретения, установки и управления собственными серверами для запуска приложений происходит аренда серверов у крупных компаний, таких как Microsoft, Amazon, Google или другой компании. Затем пользователь управляет своими арендованными серверами через Интернет, оплачивая при этом только фактическое их использование для обработки и хранения данных.

Вычислительные облака состоят из тысяч серверов, размещенных в дата-центрах, обеспечивающих работу десятков и сотен тысяч приложений, которые одновременно используют миллионы пользователей. Обязательным условием

эффективного управления такой крупномасштабной инфраструктурой является максимально полная автоматизация. Кроме того, для обеспечения различным видам пользователей − облачным операторам, сервис-провайдерам, посредникам, ИТ-администраторам, пользователям приложений − защищенного доступа к вычислительным ресурсам облачная инфраструктура должна предусматривать возможность самоуправления и делегирования полномочий [1; 2].

Согласно исследованиям компании Gartner технологии ОВ развивались в три этапа, немного совпадающих друг с другом по времени. Первый этап (2007–2012) – период формирования рынка. ОВ в этот период развивались за счет компаний, которых ОВ привлекают возможностью быстрого выхода на рынок и радикального повышения эффективности разработки. На этом этапе ОВ наиболее эффективны в рамках ИТ-проектов, предусматривающих возврат инвестиций в перспективе 18–24 месяца [1; 3].

Основной результат второго этапа (2011–2014) – становление рынка ОВ. К 2014 году количество облачных предложений превзошло потребности рынка, борьба за пользователей среди различных облачных вендоров достигла своего пика, что привело к серии слияний и поглощений. В то же время зрелость облачных предложений повысилась и консервативные пользователи начали всерьез рассматривать возможность использования облачных вычислений. Продолжительность облачных проектов увеличилась, и компании инициируют проекты, предусматривающие возврат инвестиций в перспективе от 3 до 5 лет. К 2015–2016 году ОВ стали предпочтительны при разработке простых в архитектурном отношении приложений среди тысяч ведущих глобальных компаний.

В 2015–2018 годах наступило накопление критической массы и массовое распространение облачных вычислений. Доминировать на рынке стали ключевые поставщики, которые получили возможность предлагать рынку свои технологии в качестве стандартов де-факто.

Технологии. С понятием ОВ часто связывают такие сервис-предоставляющие (Everything as a service) технологии, как [1; 2]:

− «Инфраструктура как сервис» (Infrastructure as a Service или IaaS);

− «Платформа как сервис» (Plaatform as a Service, PaaS);

− «ПО как сервис» (Software as a Service или SaaS).

Инфраструктура как сервис (IaaS) − это предоставление вычислительной инфраструктуры (небольшой ВЦ предприятия) как услуги на основе концепции облачных вычислений. IaaS состоит из трех основных составляющих: аппаратные средства (серверы, системы хранения данных, клиентские системы, сетевое оборудование); операционные системы и системное ПО (средства виртуализации, автоматизации, основные средства управления ресурсами); прикладное ПО (для управления производственными системами).

IaaS основана на технологии виртуализации, позволяющей пользователю оборудования делить его на части, которые соответствуют текущим потребностям производства, тем самым увеличивая эффективность использования имеющихся вычислительных мощностей. Компания или предприниматель должны

оплачивать всего лишь реально необходимые для работы серверное время, дисковое пространство, сетевую пропускную способность и другие ресурсы. Кроме того, IaaS предоставляет в распоряжение компании весь набор функций управления на одной интегрированной платформе.

IaaS избавляет предприятия от необходимости поддержки сложных инфраструктур центров обработки данных, клиентских и сетевых инфраструктур, а также позволяет до минимума снизить капитальные затраты и сократить текущие расходы.

Платформа как сервис (PaaS) − это предоставление компьютерной платформы для разработки, тестирования, развертывания и поддержки веб-приложений как услуги. Для запуска и использования веб-приложений разработчику не нужно приобретать оборудование и программное обеспечение, нет необходимости организовывать их поддержку. Доступ для клиента может быть организован на условиях аренды. Данная модель имеет следующие достоинства: масштабируемость, отказоустойчивость, виртуализация, безопасность [1; 2].

Масштабируемость PaaS предполагает автоматическое выделение и освобождение необходимых ресурсов в зависимости от количества обслуживаемых приложением пользователей. PaaS как интегрированная платформа для разработки, тестирования, разворачивания и поддержки веб-приложений позволит весь перечень операций по обслуживаниию выполнять в одной интегрированной среде. Способность создавать программный код и предоставлять его в общий доступ внутри компании повышает производительность по созданию приложений на основе модели PaaS.

ПО как сервис (SaaS) – модель развертывания программного приложения, которая предусматривает предоставление этого приложения конечному пользователю как сервис по требованию. Доступ к такому приложению осуществляется посредством браузера. Основное преимущество модели SaaS для клиента состоит в отсутствии затрат, связанных с установкой, обновлением и поддержкой работоспособности данного программного обеспечения, его информационной безопасности. В модели SaaS ряд достоинств: приложение возможно использовать для удаленного доступа; одним приложением могут пользоваться несколько клиентов; оплата за услугу взимается либо как ежемесячная плата, либо на основе суммарного объема транзакций; поддержка приложения входит уже в состав оплаты; модернизация приложения может производиться обслуживающим персоналом незаметно для потребителя [1; 2].

С точки зрения разработчиков программного обеспечения, модель SaaS позволит эффективно бороться с нелицензионным использованием программного обеспечения, благодаря тому что пользователь не может хранить, копировать и устанавливать программное обеспечение. В целом программное обеспечение в рамках SaaS можно рассматривать в качестве более удобной и выгодной альтернативы внутренним информационным системам организации.

С 2017 появилась технология XaaS «Что угодно как услуга» [10]. Помимо трех вышеописанных моделей она может организовывать семь типов бизнеса: AnaaS «Аналитика как услуга» (компании превращают данные в аналитические

данные и используют их для принятия бизнес-решений); DaaS (рабочий стол, как услуга) позволяют пользователям управлять всей своей рабочей силой через безопасный веб-браузер; FaaS (функция как услуга) позволяют предприятиям использовать определенные функции или результаты, не заставляя их разрабатывать или запускать приложения; STaaS (хранилище как услуга) может высвободить внутренние ресурсы и снизить затраты организации; CaaS (контейнер как услуга), если клиент не хочет создавать контейнер для хранения библиотеки кода, он может приобрести его у компании CaaS для решения этой проблемы; DBaaS (БД как услуга) позволяют компаниям организовывать, фильтровать и хранить данные о клиентах в программном обеспечении, к которому нужный сотрудник может легко получить доступ; AaaS (аутентификация как услуга) предоставляют пользователям возможность внедрять решения для контроля доступа на своей платформе.

Рассмотрим несколько вариантов структур ОВ для организаций.

Частное облако (private cloud) используется для предоставления сервисов внутри одной компании, которая является одновременно и заказчиком и поставщиком услуг. Это вариант реализации «облачной концепции», когда компания создает ее для себя самой, в рамках организации. Реализация структуры ОВ private cloud снимает один из основных вопросов, который возникает у заказчиков при ознакомлении с этой концепцией – вопрос о защите данных с точки зрения информационной безопасности. Поскольку ОВ ограничены рамками самой компании, этот вопрос решается стандартными существующими методами. Для private cloud характерно снижение стоимости оборудования за счет использования простаивающих или неэффективно используемых ресурсов, а также, понижение затрат на закупки оборудования за счет сокращения логистики [1; 2].

Гибридные «облака» представляют собой такое внедрение структуры ОВ, при котором часть системы размещается в публичном «облаке», т. е. на базе дата-центров облачного провайдера, а часть – в приватном «облаке», т. е. на серверах, принадлежащих лично компании. Структура гибридное «облако» не является самостоятельным типом облачных внедрений, а лишь показывает на тесную интеграцию публичных и приватных облачных систем.

11.2 Основы виртуализации

В основе виртуализации для технологии ОВ лежит возможность одного компьютера выполнять работу нескольких вычислителей благодаря распределению его ресурсов по разным уровням (средам). С помощью виртуальных серверов и виртуальных компьютеров можно разместить несколько ОС и несколько приложений в едином физическом пространстве.

Возможность запуска нескольких виртуальных машин (ВМ) на одной физической вызывает определенный интерес среди ИТ-специалистов не только потому, что это повышает гибкость ИТ-инфраструктуры, но и виртуализация на самом деле позволяет экономить средства [1; 2].

ОС не может различить виртуальную и физическую машины. Это же можно заметить касаемо приложений и других ресурсов в сети. Но несмотря на это, ВМ состоят исключительно из программных составляющих и не включают оборудование.

Рассмотрим основные разновидности виртуализации: виртуализация серверов (полная виртуализация и паравиртуализация); виртуализация на уровне ОС; виртуализация приложений.

Виртуализация серверов подразумевает запуск на одном физическом сервере нескольких виртуальных серверов.

Полная виртуализация (Full Virtualization). Используются немодифицированные экземпляры гостевых ОС, для поддержки работы которых служит общий слой эмулирования их исполнения сверху основной ОС, в роли которой выступает обычная ОС (рисунок 11.1).

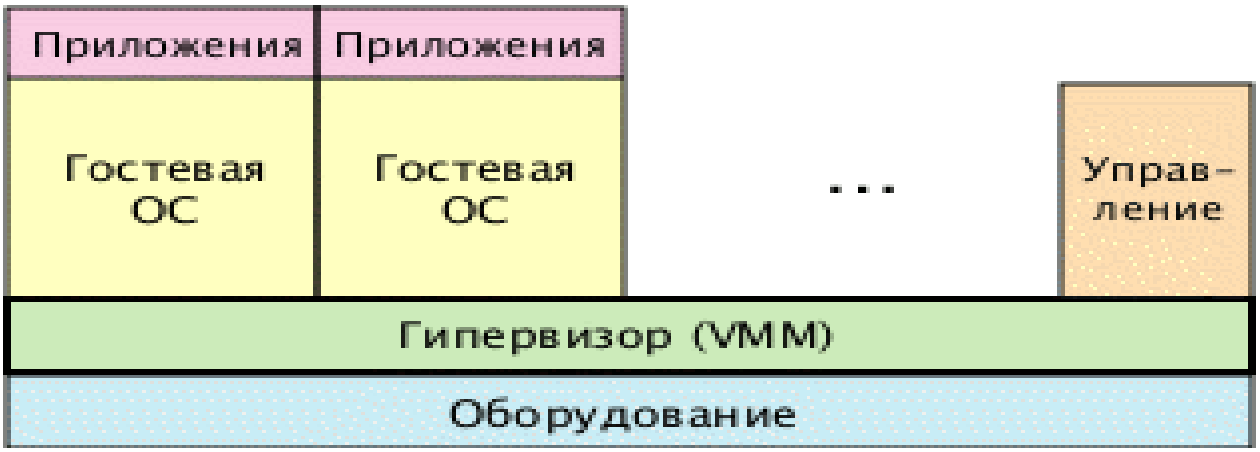

Рисунок 11.1 – Полная виртуализация

Данная технология применяется в средствах VMware Workstation, VMware Server (Parallels Desktop, Parallels Server, MS Virtual PC, MS Virtual Server). К достоинствам этого подхода можно отнести простоту реализации, универсальность и надежность решения; функции управления берет на себя хост-ОС. Недостатки − высокие накладные расходы на используемые аппаратные ресурсы, отсутствие учета особенностей гостевых ОС, меньшая гибкость в использовании аппаратных средств.

Паравиртуализация (paravirtualization). Она включает в себя модификация ядра гостевой ОС и выполняется так, что в нее включается новый набор интерфейсов (API), через который можно напрямую работать с аппаратурой, не конфликтуя с другими ВМ. При этом нет необходимости задействовать полноценную ОС в качестве хостового ПО, функции которого в этом случае исполняет специализированная система, получившая название гипервизора (hypervisor). Данный вариант является наиболее используемым направлением развития серверных технологий виртуализации и применяется в продуктах VMware ESX Server, Xen (и решениях других поставщиков на базе этой технологии), MC Hyper-V [1; 2].

Виртуализация на уровне ядра ОС (operating system-level virtualization). Данный вариант включает использование одного ядра хостовой ОС для создания независимых параллельно работающих операционных сред. Достоинства – высокая эффективность использования аппаратных ресурсов, низкие накладные технические расходы, хорошая управляемость, минимизация расходов на приобретение лицензий. Недостатки – возможна реализация только однородных вычислительных сред.

Виртуализация приложений подразумевает применение модели сильной изоляции прикладных программ с управляемым взаимодействием с ОС, при которой виртуализируется каждый экземпляр приложений, все его основные компоненты: файлы (включая системные), реестр, шрифты, INI-файлы, COM-объекты, службы (рисунок 11.2) [1; 2].

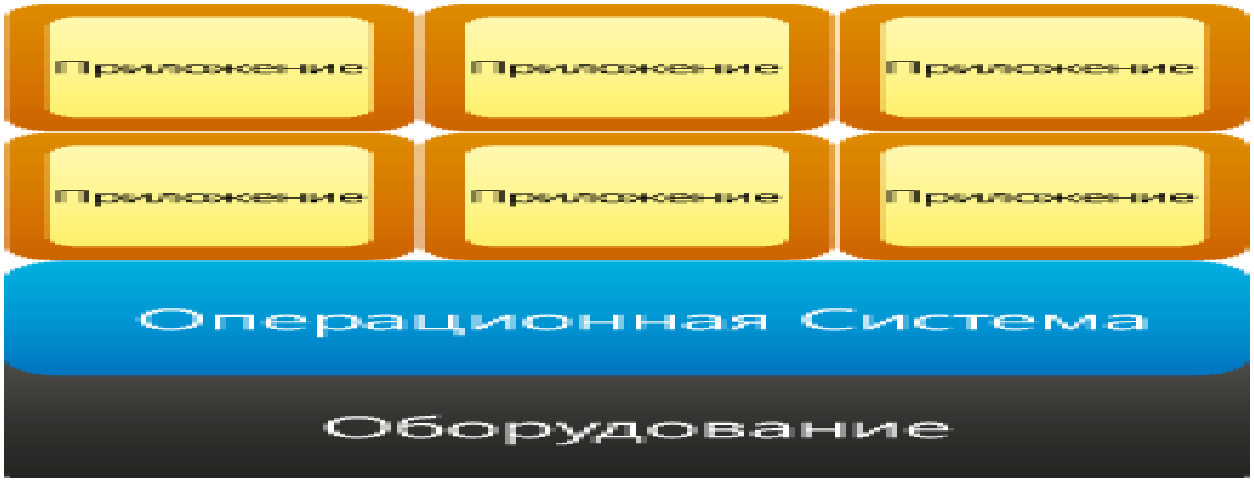

Рисунок 11.2 – Виртуализация приложений

Приложение исполняется без процедуры инсталляции в традиционном ее понимании и может запускаться прямо с внешних носителей. С точки зрения ИТ-отдела, такой подход имеет очевидные преимущества: ускорение развертывания настольных систем и возможность управления ими, тестировании приложений на совместимость.

11.3 Использование облачных вычислений в организации

«Облако» – это пространство, куда пользователь (или организация) обращаются, когда нужно использовать определенную технологию, и использует ее до тех пор, пока она необходимо. Для этого не нужно ничего устанавливать на компьютер, не надо платить за технологию, когда ею не пользуются. Три условия, при которых данное использование определяет, является ли конкретная услуга облачной: ресурсы доступны через веб-браузер или с использованием специального интерфейса прикладного программирования (API) для доступа к веб-службам; для начала использования услуги не требуется никаких капитальных затрат; пользователь платит только за то, что он использует, и оплачивает только то время, в течение которого он ресурсы ОВ использует [2].

Облако может быть приложением, доступ к которому осуществляется через Интернет, или сервером, к которому пользователь обращается, когда ему это необходимо. Для пользователей SaaS-систем не имеет значения, где установлено программное обеспечение, какую операционную систему оно использует и на каком языке написано: PHP, Java или .NET. SalesForce.com − это вариант SaaS. Представляет систему управления взаимоотношениями с клиентами (CRM), разработанную для среднего или большого предприятия. SalesForce.com предназначен для специалистов отделов продаж и позволяет им отслеживать потенциальных клиентов и менеджеров по закупкам, определять их место в организации процесса продаж, управлять всем процессом заключения сделки от первого контакта до заключения контракта и расчетов и т. д. [2; 3]. Средства SaaS имеют некие определяющие характеристики, такие как: доступность через веб-браузер, доступность по требованию. Условия оплаты зависят от использования. Многопользовательское приложение – это серверная программа, которая поддерживает одновременную работу нескольких клиентов с использованием одного экземпляра серверной программы. Эта модель предлагает поставщику SaaS очевидные преимущества в том, что она позволяет предоставлять конечным пользователям такие функции, как поддержка большего числа клиентов на меньшем количестве аппаратных средств; ускоренная и упрощенная процедура развертывания обновлений приложений и систем безопасности; более надежная архитектура [2].

Среды PaaS предоставляют инфраструктуру, а также функционально завершенные среды обслуживания и разработки приложений для развертывания определяемого пользователем веб-приложения. Он программируется с использованием платформы разработки приложений, предоставляемой поставщиком, и предоставляет пользователю не заботиться обо всех деталях развертывания готового приложения [2].

Важным бизнес-аргументом в пользу «облаков» является преобразование капитальных затрат в операционные. В данном случае предприятиям больше не нужно инвестировать в собственные ИТ-центры и размещенное в них оборудование. Вместо этого достаточно арендовать необходимые ресурсы у специализированного поставщика ОВ и платить арендную плату за те мощности, которые фактически используются за конкретное время. В этом случае снижаются риски, связанные с невозможностью спрогнозировать выполнение ИТ-проектов: если проект окажется неудачным, достаточно будет вернуть арендованные мощности подрядчику и заплатить за время их использования. Риск остаться с ненужными аппаратно-программными средствами уходит в прошлое.

Каждый разработчик знает, сколько трудностей может вызвать перемещение приложения с опытной площадки на производственную: изменения параметров конфигурации, различия в версиях программного обеспечения и разных моделях используемого оборудования могут вызвать непредсказуемые ошибки, исправление которых может значительно изменить время выполнения ИТ-проекта. В условиях технологии «облаков» конфигурация виртуальной машины при разработке и промышленной эксплуатации практически идентична,

это устраняет проблемы, связанные с различиями в аппаратном и программном обеспечении разрабатываемого ИТ-проекта [2]. Это создает новую «платформу для бизнес-операций», которая позволит компаниям изменить свои бизнес-модели и найти мощные, ранее неиспользуемые способы взаимодействия с клиентами, поставщиками и торговыми партнерами [4].

Когда некий специалист выбирал решение для бухгалтерского учета, он рассматривал SaaS-систему «Мой бизнес» наряду с традиционными полнофункциональными системами бухгалтерского учета. 1С оттолкнула своей новизной, решение от Buchsoft оказалось недостаточно удобным. В то же время с обеими системами была связана дополнительная проблема: они жестко привязаны к конкретному рабочему месту. Система «Мой бизнес» не имела этого недостатка: вы можете управлять компанией и вести учет из любого места, даже находясь в поездке. По словам специалиста, эксплуатация системы «Мой бизнес» обходится в 10–20 раз дешевле, чем услуги бухгалтера или аутсорсинговой компании, специализирующейся на бухгалтерских услугах: вместо 40 тысяч рублей в месяц за услуги такой компании достаточно заплатить всего 2 тысячи рублей за аренду ресурса ОВ «Мой бизнес». И вообще не нужно беспокоиться об администрировании и обновлениях программного обеспечения [1].

Четвертой волне автоматизации бизнеса − внедрению систем управления человеческим взаимодействием (HIMS) − способствуют технологии. Формирующаяся форма организации – «Облачное предприятие», которое изменит структуру и стиль управления и станет органичной сетью, а не иерархическим монолитом, разделенным на отделы. Облако – это не просто технология: это новая платформа для человеческого общения в бизнесе, что, в свою очередь, требует новых организационных структур, стилей управления и новых моделей командного поведения.

Экономичность и результативность – платить столько, сколько используется, позволять дорогостоящие, мощные компьютеры и современные программы. Технология «Облако» позволяет учитывать и оплачивать только фактически потребленные ресурсы сразу после их использования.

Аренда ресурсов. Обычные серверы усредненной компании загружены на 10–15 %. В отдельные периоды времени возникает потребность в дополнительных вычислительных ресурсах (конец отчетного периода), в другие (отпускной период) эти дорогостоящие ресурсы простаивают. Используя необходимый объем вычислительных ресурсов технологии ОВ в нужный момент времени, компании снижают затраты на оборудование и его обслуживание. Это позволяет компании, особенно новой, отказаться от приобретения дорогостоящих ИТ-активов в пользу не аренды, а оперативного потребления по мере необходимости, снижая при этом затраты на обслуживание своих систем и получая гарантии уровня обслуживания от поставщика ОВ.

Аренда программного обеспечения. Вместо того чтобы приобретать пакеты программного обеспечения для каждого локального пользователя, компании покупают или арендует необходимые программы в «облаке». Эти программы будут использоваться только теми пользователями, которым эти программы

нужны в их работе. Более того, стоимость программ, ориентированных на доступ в Интернет, значительно ниже, чем их аналогов для персональных компьютеров. Если программы используются не часто, то их можно просто арендовать за почасовую плату. Затраты на обновление программ и поддержание их в рабочем состоянии на рабочем месте сводятся к нулю.

Для поставщика ОВ ИТ-услуг экономический смысл «облака» заключается в эффективности масштабирования (поддерживать большой однородный дата-центр дешевле, чем множество небольших разнородных) и сглаживании нагрузки (когда потребителей много, маловероятно, что всем им потребуется пиковая мощность одновременно) [2].

Разработчики программного обеспечения также выигрывают от перехода в «облако»: им стало проще, быстрее и дешевле разрабатывать, тестировать под нагрузкой и предлагать свои решения заказчикам – это можно делать непосредственно в облаке с минимальными затратами. К тому же ОВ является эффективным инструментом для увеличения прибыли и расширения каналов продаж для независимых производителей программного обеспечения в форме модели SaaS. Такой подход позволяет организовать динамическое предоставление услуг, когда пользователи могут производить платежи по факту и корректировать объем своих ресурсов в зависимости от реальных потребностей [1; 2].

Что делать, если отсутствуют финансовые средства для развертывания бизнеса в момент, когда возникает проблема покупки нового оборудования? Если компания управляет собственной инфраструктурой, ей потребуются инвестиции для каждой новой сети устройств хранения данных (Storage Area Network, SAN) или каждого приобретенного сервера. Для ввода в эксплуатацию потребуется достаточное время – с момента принятия решения до размещения заказа, его оплаты, физической доставки, приемки, установки, установки программного обеспечения, тестирования и ввода в эксплуатацию.

Если компания управляет собственной инфраструктурой, ей может потребоваться платить за недвижимость или электричество, не используя большую часть платных ресурсов. Это неэффективная трата денег. При использовании облачной инфраструктуры ни один из компонентов не является проблемой, поскольку ОВ позволяют [1; 2]:

− добавлять мощности в облачную инфраструктуру только в тот момент, когда это необходимо. Нет никаких затрат, связанных с распределением ресурсов, поэтому нет необходимости беспокоиться о синхронизации потребностей в мощности и бюджетных потребностях, можно быстро увеличить вычислительную мощность;

− не беспокоиться об оборудовании, на котором все это работает. Возможно, вы обнаруживаете, что физический сервер, на котором фактически выполнялась работа, полностью вышел из строя. При правильном выборе инструментов можно добиться автоматического восстановления оборудования после аварийных ситуаций;

− если требования к пропускной способности изменятся, может потребоваться другая конфигурация виртуального оборудования. В этом случае компания отказывается от виртуального сервера и получает другой. В то же время нет необходимости думать о перепродаже, утилизации или беспокоиться о вреде для окружающей среды;

− нет необходимости платить за недвижимость и электроэнергию, которыми компания не пользуется. Поскольку используется лишь малая часть аппаратных мощностей, эффективность использования физического пространства, необходимого для удовлетворения потребностей в обработке информации, повышается. Она не оплачивается за всю стойку серверов, потребляющих электроэнергию, где большинство циклов процессора простаивают. ОВ конкурируют с двумя подходами к ИТ-инфраструктуре: внутренняя ИТ-инфраструктура и поддержка, аутсорсинг (передача функций службам, управляемым сторонними организациями).

Наибольшей выигрыш от использования облачной инфраструктуры для поддержания собственной ИТ-инфраструктуры компании даже не технологический, а финансовый. Модель оплаты только за фактически потребленные услуги, принятая в облачной инфраструктуре, делает этот подход намного эффективнее, чем модель «предоплаты за все», типичная для внутренней ИТ-инфраструктуры ситуация [2].

Обсуждая экономический эффект облачных вычислений, чаще всего говорят о тех положительных экономических последствиях, которые связаны с экономией за счет масштаба. Владельцы крупных дата-центров именно из-за масштаба своих сайтов (насчитывающих десятки тысяч серверов) способны достичь специфических преимуществ, которые недоступны владельцам серверных сайтов малого (до 100 серверов) и среднего (около 1000 серверов) размера. В отличие от своих мелких и средних конкурентов владельцы крупных дата-центров способны добиться 3–7-кратной экономии на электроэнергии и сетевой инфраструктуре, снижая затраты на человеческий персонал, когда специалист обслуживает не десятки или сотни, а тысячи серверов [1; 3].

Вторая причина, по которой крупные объекты экономически более эффективны, заключается в том, что они меньше страдают от неравномерной нагрузки, вызванной ежедневным, годовым и отраслевым балансом. В то время как узкоспециализированные частные сайты создаются с учетом максимальной пиковой нагрузки, крупные центры обработки данных могут позволить себе равномерно распределять ресурсы между пользователями, чьи системы имеют разные циклы нагрузки на вычислительные ресурсы. Таким образом, крупный облачный провайдер может позволить себе сэкономить до 30–40 % на одном оборудовании по сравнению с компанией с парком из 1000 серверов [2].

При всей убедительности аргумента об экономии за счет масштаба его привлекательность несколько снижается из-за одного важного фактора: владельцы крупных сайтов хотят не только сэкономить, но и заработать. В то же время в настоящее время валовая прибыль тех же Amazon Web Services достигает около 50 %. Этот показатель будет снижаться из-за усиления конкуренции

со стороны других крупных облачных провайдеров, таких как Microsoft. Учитывая высокий порог входа (в США на строительство крупного дата-центра на 50 тысяч серверов требуется около 200 млн долл.), совершенной конкуренции на этом рынке ожидать нельзя, и здесь будут преобладать крупные поставщики [1; 2].

11.4 Поставщики услуг облачных вычислений

Западные поставщики. Первой в продвижении модели IaaS считается компания Amazon, которая предлагает два основных IaaS-продукта: EC2 (Elastic Compute Cloud) и S3 (Simple Storage Service). EC2 представляет собой Xen-хостинг со статическими VPS-характеристиками, которые не расширяются (хотя подобные сервисы уже предоставляют подобную технологию auto scaling). Хранилище S3 имеет интерфейс webDAV и поддерживает работу со многими языками программирования [1; 2; 3].

Среди других инфрасервисных компаний можно отметить GoGrid, которая имеет очень удобный интерфейс для управления VPS, а также cloud storage с поддержкой протоколов SCP, FTP, SAMBA/CIFS, RSYNC, причем размер хранилища масштабируется в процессе работы, имеется управление посредством API.

Enomaly представляет собой решение для развертывания и управления виртуальными приложениями в «облаке», а управление услугами осуществляется через браузер. Дополнением является автоматическое масштабирование виртуальных машин под текущую нагрузку, а также автобалансировка нагрузки. Среди поддерживаемых виртуальных ОС поддерживаются Linux, Windows, Solaris и BSD Guests. Для виртуализации применяют не только Xen, но и KVM, а также VMware. Eucalyptus представляет собой программный комплекс с открытым кодом для реализации cloud computing на кластерных системах. Также интерфейс совместим с Amazon EC2 [1; 2].

Самым известным примером ОВ платформы модели PaaS является AppEngine от Google, которая предлагает хостинг для веб-приложений с возможностью покупать дополнительные вычислительные ресурсы (например, для тестирования высоких нагрузок). Для запуска приложений Google AppEngine на виртуальных кластерных системах была разработана платформа AppScale, не имеющая тем не менее никакого отношения к Google.

В системах веб-поиска и контекстной рекламы компании Yahoo используется платформа Hadoop, ориентированная на передачу больших объемов данных между сетевыми серверами. На базе Hadoop построены HBase (аналог базы данных Google BigTable), а также HDFS (Hadoop Distributed File System, аналог Google File System) [1].

Представителем модели PaaS являются программы компании Mosso [2]:

− Cloud Sites − веб-хостинг (Linux, Windows, Mail) для нагрузочных веб-проектов с возможностью расширять базовые бесплатные возможности за дополнительную плату (трафик, хранилище данных, вычислительная мощность).

− Cloud Files − файловый cloud-хостинг с ежемесячной погигабайтной оплатой за объем хранимых файлов. Управление осуществляется через браузер либо посредством API (PHP, Python, Java, NET, Ruby).

− Cloud Servers − почасовая аренда серверов (RAM в час) с возможностью выбора серверной ОС. Можно изменять характеристики сервера, но не в режиме реального времени. В скором времени разработчики обещают сделать API для управления серверами.

В основе облачной инфраструктуры компании Microsoft − операционная система *Windows Azure*. Windows Azure создает единую среду, включающую облачные аналоги серверных продуктов Microsoft (реляционная база данных SQL Azure, являющаяся прототипом SQL Server, а также средства Exchange Online, SharePoint Online и Microsoft Dynamics CRM Online) и инструменты разработки (NET Framework, Visual Studio, оснащенная в версии 2010 года набором Windows Azure Tools). Программист, создающий сайт в среде Visual Studio 2010, может разместить свой сайт в Windows Azure [1].

Развитием логики модели SaaS является концепция WaaS (Workplace as a Service − рабочее место как услуга). Используя ее, клиент получает в распоряжение полностью оснащенное всем необходимым для работы виртуальное рабочее место. По опубликованным данным SoftCloud спросом пользуются SaaS-приложения (в порядке убывания популярности): почта, коммуникации (VoIP), антиспам и антивирус, Helpdesk, управление проектами, дистанционное обучение, CRM, хранение и резервирование данных [1; 2].

Похожими являются программные продукты: MobileMe (Apple), Azure (Microsoft) и LotusLive (IBM). Функции данных сервисов в том, что они предоставляют пользователям доступ к хранению своих данных (контакты, почта, файлы), а также поддерживают совместную работу нескольких пользователей с общими документами.

Конкуренция в облачной сфере привела к появлению бесплатных сервисов, например, две транснациональные ИТ-компании: Microsoft, Google. Обе компании выпустили наборы сервисов, позволяющих работать с документами. У Google − Google Docs, у Microsoft − Office Web Apps.

Google Apps − это среда, которая предоставляет средства совместной работы: ставший распространенным почтовый сервис − GMail, клиент обмена быстрыми сообщениями − Google Talk (этот сервис пригоден для общения с любым jabber-пользователем), календарь − Google Calendar, средства для работы с документами и электронными таблицами − Google Docs & Spreadsheets, «центральная страница» − место для размещения той информации, которая будет общей для всех пользователей, редактор страниц от Google, который позволяет создать и опубликовать текущую информацию [2].

Массово используемым образцом PaaS является продукт Google App Engine. Для работы с инструментом Google App Engine пишутся приложения в среде Python, используя инфраструктуры разработки, предоставляемые Google, и инструменты, предназначенные для работы с файловой системой и хранили-

щами данных Google. Этот подход хорошо работает для приложений, которые необходимо быстро развертывать и которые не предъявляют существенных требований к интеграции. Недостатком подхода PaaS является его привязка к конкретному поставщику. Например, если вы работаете с Google, то должны писать свои приложения на языке программирования Python, используя при этом специфический интерфейс прикладного программирования (API) от Google.

Идея IaaS является наиболее распространенной. В работах [1; 2; 3] приведено немало примеров, иллюстрирующих этот подход на примере одного из ключевых игроков в этой области − Amazon Web Services (AWS). У AWS есть ряд конкурентов, которые предлагают иные подходы к решению проблемы IaaS. Эти альтернативные подходы позиционируются для различных типов клиентов, использующих облачные инфраструктуры.

Продукт AWS основывается на чистой виртуализации от Amazon, которая имеет в собственности все оборудование и управляет своей сетевой инфраструктурой, а клиентам принадлежит все, что содержится в гостевых операционных системах, работающих на виртуальных машинах. Пользователь запрашивает виртуальные экземпляры по требованию и освобождает их по мере того, как они перестают быть ему нужными. Одно из преимуществ ОВ компания Amazon видит в том, что обязуется не выделять на виртуализацию больше ресурсов, чем нужно пользователю.

Продукт AppNexus − это другой подход к данной проблеме. По аналогии с AWS, AppNexus позволяет получить доступ к серверам по мере необходимости. Однако AppNexus предлагает выделенные серверы с виртуализацией, реализованной иначе. При этом пользователю можно быть уверенным в том, что его приложения не конкурируют за ресурсы с другими приложениями. Ему обеспечивается выполнение требований полного контроля над физическими ресурсами сервера [1].

В последнее время крупнейшие облачные компании активно перестраивают свою стратегию с учетом «гибридизации» облачных вычислений. Так, Amazon Web Services планирует развиваться не только путем наращивания собственных услуг, но и путем создания сообщества партнеров и поставщиков, сервисы которых будут интегрироваться с ОВ от Amazon. Таким образом, Amazon Web Services будет становиться платформой все более открытой для интеграции – а это прямой путь к гибридной модели.

Еще более последовательно в сторону гибридной модели движется MS, облачная стратегия которой предполагает возможность размещения вычислительных мощностей по выбору: на собственной площадке, в публичном «облаке» или у сервис-провайдера. Это позволяет комбинировать элементы публичного и приватного «облака» в тех соотношениях, которые наиболее удобны для компании. В рамках этой стратегии MS добавила возможность помещения самостоятельно сформированных образов виртуальных машин в »облаке» Windows Azure [1].

IBM предлагает собственные SaaS-решения (Lotus Live, Tivoli Live), публичные IaaS-решения на базе собственных вычислительных ресурсов IBM (Enterprise и Enterprise+), элементы PaaS-платформы (совместно с Amazon и на базе SmartCloud Enterprise). Кроме того, IBM предлагает целый спектр инструментов для развертывания приватных «облаков», который представляет собой знакомые продукты IBM для управления корпоративными вычислительными ресурсами, из линейки продуктов Tivoli.

Предложения по построению приватного «облака» HP (HP I CloudSystem) по сути являются технологиями развертывания корпоративных дата-центров, в то время как предложения по аренде вычислительных ресурсов HP (HP Enterprise Cloud Services-Compute) лишь отдаленно напоминают облачные сервисы Amazon, позволяющие использовать заказанные ресурсы через несколько минут после оплаты по кредитной карте [1; 2].

Компания Oracle обладает сильными позициями на рынке инфраструктурных решений для развертывания приватных «облаков» (благодаря интегрированной аппаратно-программной платформе Exalogic, основанной на серверных разработках Sun Microsystems), на рынке арендуемых SaaS-приложений (старая линейка решений On Demand). Oracle также предлагает возможность использования своей СУБД на публичной площадке Amazon Web Services.

Поставщики в СНГ. Подобно CloudSigma, российские предложения по облачному хостингу ориентированы преимущественно на масштабирование в пределах отдельно взятой виртуальной машины. Некоторые провайдеры, например, Slidebar/Parking, предлагают возможность вынесения в «облако» корпоративной инфраструктуры, однако таких мощных инструментов для построения, администрирования и мониторинга облачной инфраструктуры, как у Amazon, в России пока предложить не могут [1; 2].

Но российские облачные провайдеры позволяют увеличивать и уменьшать количество доступной оперативной памяти и дискового пространства и платить только лишь за фактически использованное процессорное время. Наиболее популярным среди российских облачных хостинг-провайдеров является предоставление пользователям возможности самостоятельно определить количество необходимых ресурсов с помощью «ползунка»: пользователь выбирает объем необходимой памяти, дисковое пространство, объем доступных ресурсов CPU. После этого он получат виртуальную машину с заданными характеристиками. Если в течение времени потребности в вычислительных ресурсах снизятся или увеличатся, характеристики виртуальной машины можно будет изменить (с соответствующей корректировкой оплаты).

11.5 Защита информации в технологии облачных вычислений

В связи с широким использованием технологий облачных вычислений (ОВ) актуальна проблема ЗИ для них. Причины, обуславливающие возникновения уязвимостей в среде облачных вычислений, следующие: объем обрабатываемой информации постоянно увеличивается; доступ к ресурсам вычисли-

тельных комплексов получает большее число пользователей; низкий уровень компьютерной грамотности пользователей, недостаточная квалификация системных администраторов. Одной из важнейших задач компьютерной безопасности является борьба с вредоносным ПО и в, частности, подзадача его обнаружения [7].

Все методики обнаружения можно разделить на 2 типа: методики обнаружения известного вредоносного ПО (ВПО) и методики обнаружения неизвестного ВПО. Среди общих рисков для облачных ИТКС можно выделить: кража учетных записей; атаки на гипервизор; атаки на виртуальную инфраструктуру; блокирование сегмента облачной ИТКС; кража вычислительных ресурсов ИТКС; нарушение механизма динамической балансировки ИТКС [7].

Суть подхода в области создания защищенных систем сводится к созданию изолированной информационной среды, реализации политики многоуровневого разделения доступа, расширенных механизмов идентификации-аутентификации. Рассмотрим отдельные результаты в этой области. В работе [5] по ЗИ в среде облачных вычислений получено:

– математическая модель представления программного обеспечения (ПО) в терминах теории графов и теории множеств, позволяющая анализировать процесс его выполнения;

– способ формального описания классифицирующего признака ПО;

– алгоритм классификации ПО, обладающее и не обладающее заданным признаком;

– подход к оценке подобия различных экземпляров программного обеспечения, основанный на мере Дамерау – Левенштейна;

– методика верификации ПО на наличие деструктивных свойств для сред ОВ, использующая предложенный подход к оценке подобия различных экземпляров.

В работе [6] предложена формализация и контроль информационного взаимодействия в форме виртуальных соединений с помощью межсетевых экранов. Разработанная модель учитывает динамический характер выделяемых ресурсов и структуру протоколов сетевого взаимодействия, что позволяет осуществлять разграничение доступа с учетом текущего состояния защищаемой среды. Входом модели является поток сетевых пакетов, которые поступают в межсетевые экраны системы ЗИ в среде ОВ, а выходом является разделение пакетов на виртуальные соединения, классификация каждого дана на принадлежность соединению и определения подмножества правил фильтрации для них.

Разработанный алгоритм классификации виртуальных соединений, использующий технологии организации параллельных вычислений и структуру стека TCP/IP, позволил повысить производительность межсетевых экранов и более эффективно использовать вычислительные ресурсы аппаратных платформ, что, в свою очередь, снижает потери от наличия средств разграничения доступа в среде облачных вычислений.

Резюмируя вышеизложенное, констатируем. В качестве тенденций развития интеллектуальных технологий и ИСЗИ для КИС и ОВ можно представить следующее [5; 6]:

– развитие инструментальных программных комплексов с интеллектуальной ППР по выбору эффективных методов, моделей и алгоритмов в КИС и ОВ;

– развитие технологий многоагентных систем для обнаружения атак, противодействия угрозам нарушения ИБ, оценки уровня защищенности информации в КИС и ОВ.

– разработка теоретических основ, моделей и средств защиты облачной инструментальной платформы проектирования интеллектуальных систем на основе семантических технологий.

Процедура парольной аутентификации пользователя среды ОВ и сервера ОВ происходит в момент обращения пользователя за услугами ОВ. В случае успешной аутентификации пользователь ОВ может передать задание для выполнения, иначе он получает сообщение об ошибке аутентификации. Верификация заданий перед их отправкой для вычисления на клиенты ОВ не позволяет передать небезопасный код для исполнения в ОВ [5].

Рассмотрим модули, входящие в состав «Координатора безопасности», – модуль «Идентификация/аутентификация клиентов ОВ» и модуль «Создание и проверка ЭЦП». Аутентификация производится на основе инфраструктуры открытых ключей, путем обмена сертификатами клиента ОВГ и сервера ОВГ. Корневым центром сертификации является самоподписанный центр сертификации, входящий в состав модуля «Идентификация/аутентификация клиентов ОВ». В случае неуспешной аутентификации считается, что ключи клиента скомпрометированы. Пометка о ненадежности данного вычислительного узла делается в базе данных клиентов ОВ. Требуется переиздание сертификата для данного клиента ОВ. В случае успешного прохождения процедуры аутентификации клиент ОВГ включается в состав подоблака для проведения вычислений [8].

Для обеспечения безопасной передачи кода задания и результата вычислений между Координатором ОВ и клиентом ОВ используется шифрование. Основными требованиями, предъявляющимися к способу шифрования, являются высокая скорость шифрования, безопасное распределение секретных ключей, отсутствие проблемы хранения секретных ключей клиентов ОВ. Предъявленным требованиям удовлетворяет технология «Цифрового конверта», заключающаяся в генерации отправителем сеансового секретного ключа для шифрования данных и шифрования сеансового ключа на открытом ключе получателя.

Недостатком цифрового конверта является то, что получатель не знает, кто именно отправил ему сообщение, так как открытый ключ может быть доступен каждому. Для идентификации отправителя применяется ЭЦП. Использование ЭЦП гарантирует неотказуемость от авторства и целостность передаваемого сообщения [4; 7]. За генерацию и проверку ЭЦП отвечает «Модуль создания и проверки ЭЦП», входящий в состав «Координатора безопасности» [8].

11.6 Интеллектуальное управление на базе облачных вычислений

Структура и стиль менеджмента ранее адаптировались к фабрике как к средоточию экономической активности, следует теперь адаптировать менеджмент к ОВ. Зарождающаяся форма организации − «Облачное предприятие» (Cloud Enterprise) − изменит структуру и стиль управления и станет органической сетью, а не иерархическим, разделенным на департаменты, монолитом. ОВ дают возможность нескольким компаниям объединиться и действовать в качестве единой системы не только ради производительности, но и, что важнее, ради *скорости* реакции и инноваций. В ОВ лидеры не отдают команды, а передают информацию, доверяя компетентности участников команды и добиваясь ответственности благодаря прозрачности отношений [4].

Старые методы прогнозирования, планирования сверху вниз, «управления и контроля», хорошо служившие миру во время Промышленной эры, где производительность ставилась во главу угла, оказываются бесполезными в децентрализованном, аутсорсинговом деловом мире XXI века, где все ускоряющиеся перемены − единственное, что остается постоянным, а хорошо информированных потребителей от самых лучших товаров и услуг отделяет всего один щелчок мыши [7].

Власть перешла от производителей к потребителям, которые могут выдвигать требования: что они хотят получить, когда и где. Потребитель становится определяющим, и компании должны преобразиться: стать не продавцами для своих клиентов, а покупателями для них. Бизнес-процессы включают в себя создание и пересмотр организационной стратегии, процессы бюджетирования, принятие управленческих решений, распределение капитала и внутреннее сотрудничество.

Сложная адаптивная система (Complex Adaptive System − CAS) − это динамическая сеть из многих агентов (которые могут быть представлены клетками, видами, отдельными особями, командами, фирмами, нациями), работающих параллельно, постоянно действующих и реагирующих на то, что делают другие агенты. Управление в CAS обычно сильно рассеяно и децентрализовано [7].

Представим ряд организационных принципов «морской звезды», которые соответствуют определению гибкого предприятия в ОВ: проекты могут выдвигаться в любой части организации, их предлагают даже внешние партнеры; никто не отвечает за все, если вывести из строя подразделение, организация в целом быстро восстановится; участники функционируют автономно, что делает возможным масштабирование рабочей силы; роли аморфны и постоянно меняются; задачи выполняются, «когда потребуется»; знания и власть распределены; интеллект разбросан по всей организации; рабочие группы общаются напрямую, а не по иерархии; ключевые решения принимаются совместно, прямо на месте. По словам профессора технологии из Университета Индианы доктора Куртиса Бонка, «…мы вступаем в век активного вовлечения сотрудников в управление, широко распределенного лидерства без явного лидера при мгновенной коммуникации и технологии, которые все это делают возможным» [4].

Природные команды обладают чертами, которые обычно отсутствуют в организационных командах: коллективное лидерство: любой участник группы может взять лидерство на себя; кластеризация: вовлечение в деятельность многих с помощью нескольких. Биокоманды решают проблемы и учатся с помощью быстрых экспериментов и эволюции.

ОВ связаны не только с технологией: это новая платформа для человеческого общения в бизнесе, которая, в свою очередь, требует новых организационных структур, стилей управления и новых моделей командного поведения. С появлением ОВ, социальных сетей, мобильной связи и развитием сотрудничества само значение слова «команда» изменилось. ОВ предоставляют новую платформу для бизнеса, благодаря которой становятся возможными операционные инновации и неизвестные ранее формы сотрудничества. Это открывает далеко идущие возможности для инноваций в управлении и формировании новых конкурентных преимуществ.

Наступило время «экономики идей», в которой нематериальные активы (знания) являются основным ресурсом. Чтобы попасть в нее, лидерам компаний не нужно прибегать к услугам футурологов. Им следует создавать сценарии возможного будущего, которые одновременно оптимистичны и вероятны. С помощью новой адаптивной бизнес-платформы, сформированной в ОВ, экспериментируя, они могут прокладывать свою дорогу вперед [4].

Интеллектуализация управления в ОВ. Web 3.0 (интеллектуальная сеть) становится очередным этапом использования Интернета и поможет работать с информаций. Онтология формирует семантику, создав новые возможности для интеллектуальных агентов выполнять запросы пользователей [2; 7]. Открытое извлечение информации (Information Extraction − IE) обеспечит работу новых форм поиска, освободив пользователей от задачи по исследованию документов, выданных поисковой машиной. Сейчас применяются серверы исполнения деловых регламентов (BRE − Business Rules Engine). Но чтобы справиться со сложностью БП, связывающих несколько предприятий или цепочку создания инноваций в Web 3.0, компании потребуют создания интеллектуальных процессов, превосходящих серверы исполнения деловых регламентов [4;7].

Распределенный ИИ − DAI (Distributed Artificial Intelligence) основывается на агентных технологиях. Стандартный программный агент имеет три свойства: автономность, способность реагировать и способность выйти на связь. Добавив к этому способность планировать и ставить цели, поддерживать модели представлений, рассуждать о действиях и повышать уровень знаний и качество работы через обучение, получим главные компоненты интеллектуального агента [4; 7].

Интеллектуальные агенты могут быть интегрированы в структуры ОВ, содержащие конкретные функции по решению задач, обработки данных и управления. Они поддерживают соединение информации и технологий, основанных на знаниях, выполняют процесс логических рассуждений (например, включение в них деловых регламентов); позволяют включить функцию обучения и самосовершенствования как на уровне инфраструктуры (адаптивная

маршрутизация), так и на уровне приложения (адаптивные пользовательские интерфейсы). Интеллектуальные агенты будут использоваться для сбора бизнес-аналитики BI (Business Intelligence) и процессов обработки сложных событий − CEP (Complex Event Processing). Как и информационные системы бухгалтерии, системы бизнес-аналитики предприятия устарели. В Web 1.0 важным показателем было количество посещений определенной страницы. Сейчас важно количество связей в социальных сетях, количество отправленных сообщений и время, проведенное на конкретном сайте [4; 7].

Получение правильной информации и непрерывный анализ в реальном времени в ОВ − это следующая сложная задача для корпоративного интеллекта, особенно для того, чтобы найти ценную информацию и «управлять репутацией». Для этого необходимо переходить от «поиска в данных» к «поиску в блогах». Требуется выйти за пределы поисковика Google, обработать интернет-шум, чтобы понять, что же происходит в отрасли, оценить ситуацию о товарах и услугах компании (на рынке), т. е. нужна аналитика Web 3.0.

Системы роевого интеллекта − это многоагентные системы, состоящие из простых агентов, взаимодействующих друг с другом и окружающей средой. Агенты следуют простым правилам, и, хотя нет никакой централизованной контрольной структуры, описывающей, как должны себя вести отдельные агенты, взаимодействия между ними приводят к появлению интеллектуального поведения, неизвестного отдельным агентам (стаи птиц, косяки рыб) [4].

Используя обработку сложных событий для корпоративного интеллекта, можно создать обратную связь между ним и системой управления бизнес-процессами, которая по обратной связи воздействует на корпоративный интеллект. Когда компании выводят управление бизнес-процессами в сложную деловую экосистему, формирующуюся вокруг цепочки создания ценности, ценность обработки сложных событий становится опорой для корпоративного интеллекта и анализа процессов в реальном времени, необходимых для того, чтобы создать и совершенствовать постоянно меняющиеся бизнес-процессы.

Управление и инжиниринг сервисами знаний − Service Science Management and Engineering (SSME) − термин, используемый IBM Research в своих разработках в области сервисных систем. HP создала «Научный центр систем и сервисов». Oracle Corp. присоединилась к IBM для создания индустриального консорциума под названием Service Research and Innovation Initiative. В Европейском союзе создана рабочая группа по вопросам науки о сервисах − NESSI (Networked European Software and Services Initiative). В Калифорнийском университете Беркли реализована программа SSME. Все это происходит потому, что в сфере услуг заняты более 50 % рабочей силы в Бразилии, России, Японии и Германии, а также 75 % рабочей силы в США и Великобритании [4; 7].

11.7 Структура ЦОД и туманные вычисления

Новый тип серверов XXI века − модульные, называемые Blade-серверами, или серверами-лезвиями (blade − лезвие). Преимущества Blade-серверов изго-

товители описывают посредством правила «1234», которое трактуется следующим образом. «По сравнению с обычными серверами при сравнимой производительности Blade-серверы занимают в два раза меньше места, потребляют в три раза меньше энергии и обходятся в четыре раза дешевле» (рисунок 11.3) [1].

По определению, данному аналитической компанией IDC, *Blade-сервер или лезвие* – это модульная одноплатная компьютерная система, включающая процессор и память. Лезвия вставляются в специальное шасси с объединительной панелью (backplane), обеспечивающей им подключение к сети и подачу электропитания [1].

Рисунок 11.3 – Конструкция Blade-серверов

Это шасси с лезвиями является Blade-системой. Оно выполнено в конструктиве для установки в стандартную 19-дюймовую стойку и в зависимости от модели и производителя занимает в ней 3U, 6U или 10U (один U − unit, или монтажная единица, равен 1,75 дюйма). За счет общего использования таких компонентов, как источники питания, сетевые карты и жесткие диски, Blade-серверы обеспечивают более высокую плотность размещения вычислительной мощности в стойке по сравнению с обычными серверами высотой 1U и 2U [1].

Существующие сетевые протоколы вычислительных ЦОД Fiber Channel, Infiniband, традиционный Ethernet имеют ограниченные возможности по управлению трафиком и QoS.

Усовершенствованные варианты протокола Ethernet – Converged Enhanced Ethernet и Cisco Data Center Ethernet включают расширения по управлению потоками на основе приоритетов, разделению пропускной способности, управлению перегрузками и логическим состоянием полос передачи данных, по обеспечению передачи без потерь, а также по одновременному использованию нескольких параллельных путей передачи данных между узлами.

Основные недостатки данных решений – сложная децентрализованная схема управления потоками данных, основанная на множестве закрытых протоколов, значительная стоимость сетевого оборудования, сложность его расширения. Отдельно следует заметить, что данные решения используют реактив-

ную схему управления потоками, когда решения по коммутации принимаются во время передачи потока.

Перспективным направлением технологий вычислительных ЦОД является использование программно-конфигурируемых сетей (ПКС – Software Defined Network). Принципы ПКС впервые возникли в исследовательских лабораториях Стэнфорда и Беркли и развиваются в рамках консорциума Open Network Foundation и европейского проекта OFELIA. Известен опыт компаний Google и Amazon по внедрению ПКС в ЦОД. В основе подхода ПКС лежит возможность динамически управлять пересылкой данных в сети с помощью открытого протокола OpenFlow.

Все активные сетевые устройства объединяются под управлением сетевой операционной системы (СОС), которая обеспечивает приложениям доступ к управлению сетью. Сетевые контроллеры могут быть централизованными, использовать общие абстракции для пересылки пакетов. За счет управления пересылкой данных в сети использование ПКС в ЦОД позволяет реализовать схемы одновременной многопутевой передачи данных, управление потоками на основе приоритетов, виртуализацию сети, обеспечить QoS, эффективно распределить нагрузку на сеть.

Открытость стандарта OpenFlow и вынесение логики управления в отдельный контроллер упрощает программное и аппаратное обеспечение активного сетевого оборудования, что позволит в будущем производителям снизить его стоимость, уменьшатся затраты на создание ЦОД. Централизация и открытость средств управления позволяет гибко и эффективно адаптировать работу ЦОД под возникающие потребности бизнеса, что ускоряет внедрение инноваций и обеспечивает конкурентоспособность компаний.

Для работы технологий Интернета вещей можно использовать и туманные вычисления (ТВ – Fog Computing). Под ТВ подразумевается приближение «облака» к земле, в данном случае «туман» – это разновидность облачных сервисов, расположенных в окружающей нас среде. Fog Computing не альтернатива, а дополнение к Cloud Computing, и могут возникнуть ситуации их совместного действия (например, выполнение аналитического приложения), и в таком случае Cloud окажет услугу Fog [9].

ТВ дополняют ОВ и обеспечивают взаимодействие умных вещей между собой и облачными ЦОД в виде трехуровневой иерархической структуры. Верхний уровень занимают тысячи облачных ЦОД, предоставляющих ресурсы, необходимые для выполнения аналитических, программных приложений IoT. Уровнем ниже располагаются десятки тысяч распределенных управляющих ЦОД, в которых содержится «интеллект» Fog Computing, а на нижнем уровне находятся миллионы вычислительных устройств умных вещей.

Fog Computing – виртуализированная платформа, поддерживающая три типа сервисов, межмашинные коммуникации M2M, вычисления, хранение и сеть. Задача Fog Computing заключается в обеспечении взаимодействия миллиардов устройств между собой и с ОВ [9].

Парадигма Fog Computing отличается от Cloud Computing [9].

1. Распределение вычислительной мощности и реальное время. Вычислительные ресурсы могут быть размещены на периферии Сети, в Fog Computing может произойти конвергенция двух систем – управления бизнесом и технологическими системами.

2. Географическое распределение компонентов. Модель распределения сервисов в Fog Computing менее централизована, чем для облаков, а отдельные устройства могут быть связаны между собой потоками данных и предоставлять друг другу сервисы.

3. Большой объем внешних данных. Устройства могут в реальном времени генерировать гигантские объемы данных.

4. Сложная топология. Миллионы географически распределенных узлов могут создавать разнообразные и не детерминированные заранее связи.

4. Мобильность и гетерогенность. Потребуется использование протокола маршрутизации LISP (Locator/ID Separation Protocol), который позволяет разделить функциональность IP-адресов на две части: идентификаторы хостов и локаторы маршрутизации.

Выводы

На основании вышеизложенного и с учетом источников [1–10] заключаем.

1. Под облачными вычислениями обычно понимают возможность получения необходимых вычислительных мощностей по запросу из сети. ОВ – это не только технологическая инновация в ИТ, но и способ создания новых бизнес-моделей. ОВ базируются на тысячах серверов, размещенных в дата-центрах. С понятием ОВ связывают такие технологии, как инфраструктура как услуга (IaaS); платформа как услуга (PaaS); ПО как услуга (SaaS). Позже появились технологии: аналитика как услуга (AnaaS); рабочий стол как услуга (DaaS); функция как услуга (FaaS); хранилище как услуга (STaaS); контейнер как услуга (CaaS); БД как услуга (DBaaS); аутентификация как услуга (AaaS).

2. Три условия, по которым определятся, является ли тот или иной сервис облачным: сервис доступен через веб-браузер для доступа к веб-сервисам; чтобы начать пользоваться сервисом не требуется затрат капитала; пользователь платит только за то, чем он пользуется, и оплачивает только то время, в течение которого он этим пользуется. ОВ создают новую «платформу для бизнес-операций», которая даст возможность компаниям изменить их бизнес-модели и найти мощные, ранее недоступные способы взаимодействия с потребителями, поставщиками и торговыми партнерами.

3. Первопроходцем в IaaS считается компания Amazon, которая предлагает два IaaS-продукта: EC2 (Elastic Compute Cloud) и S3 (Simple Storage Service). Примером платформы PaaS является AppEngine от Google. В системах веб-поиска компании Yahoo используется платформа Hadoop. В центре облачной инфраструктуры Microsoft находится ОС Windows Azure. IBM предлагает SaaS-решения (Lotus Live, Tivoli Live).

4. Причины, обуславливающие возникновения уязвимостей в среде ОВ: объем обрабатываемой информации увеличивается; доступ к ресурсам вычислительных комплексов получает большее число пользователей; низкий уровень компьютерной грамотности пользователей; недостаточная квалификация системных администраторов. Среди общих рисков для облачных ИТКС можно выделить: кража учетных записей; атаки на гипервизор; атаки на виртуальную инфраструктуру; блокирование сегмента облачной ИТКС; кража вычислительных ресурсов ИТКС; нарушение механизма динамической балансировки ИТКС.

5. Суть подхода в области создания защищенных систем ОВ сводится к созданию изолированной информационной среды, реализации политики многоуровневого разделения доступа, расширенных механизмов аутентификации-идентификации. Потребитель становится определяющим, и компании должны преобразиться: стать не продавцами для своих клиентов, а покупателями для них; использовать технологии ОВ. Мир вступает в век активного вовлечения сотрудников в управление, широко распределенного лидерства без явного лидера при мгновенной коммуникации и технологии, которые все это делают возможным.

6. Интеллектуализация управления на базе ОВ связана с способностью облачных систем делать доступными данные о транзакциях активных БП , происходящих в реальном времени, позволит организациям реагировать на события в реальном времени. Повышается эффективность процесса принятия решений для деятельности. Автоматизация функций: создание представления клиента во всех каналах бизнеса, открытие комплексных счетов, изменение адресов в нескольких линиях бизнеса или работа с потерянными и украденными кредитными картами − вот примеры возможного применения обработки информации в реальном времени, доступной всем участникам бизнеса в ОВ.

7. Перспективным направлением технологий вычислительных ЦОД является использование программно-конфигурируемых сетей (ПКС – Software Defined Network). Все активные сетевые устройства объединяются под управлением сетевой операционной системы (СОС), которая обеспечивает приложениям доступ к управлению сетью. Открытость стандарта OpenFlow и вынесение логики управления в отдельный контроллер упрощает программное и аппаратное обеспечение активного сетевого оборудования.

8. Туманные вычисления дополняют ОВ и обеспечивают взаимодействие умных вещей между собой и облачными ЦОД в виде трехуровневой иерархической структуры. Верхний уровень занимают облачные ЦОД, предоставляющих ресурсы, необходимые для выполнения аналитических, программных приложений IoT. Уровнем ниже располагаются десятки тысяч распределенных управляющих ЦОД, в которых содержится «интеллект» Fog Computing, а на нижнем уровне находятся миллионы вычислительных устройств умных вещей.

Контрольные вопросы

1. Дайте определение ОВ.
2. Укажите три этапа развития облачных вычислений.
3. Какие технологии связывают облачные вычисления?
4. Охарактеризуйте технологию SaaS.
5. Охарактеризуйте технологию PaaS.
6. Назовите основных западных поставщиков ОВ.
7. Назовите основных поставщиков ОВ в СНГ.
8. Назовите четвертую волну автоматизации бизнеса.
9. В чем заключается эффективность от использования ОВ?
10. Поясните угрозы ЗИ в среде ОВ.
11. Как адаптировать менеджмент к ОВ?
12. В чем заключается интеллектуализация управления в ОВ?
13. Поясните структуру ЦОДа.
14. Иерархия интеллектуального управления для умных вещей.

Литература, используемая в главе

1. Клементьев, И. П. Введение в облачные вычисления. / И. П. Клементьев, В. А. Устинов. Екатеринбург : Изд-во УрУ, 2012. – 242 с.

2. Ридз, Дж. Облачные вычисления / Дж. Ридз. – СПб. : БХВ, 2011. – 288 с.

3. Облачные сервисы. Взгляд из России; под ред. Е. Гребнева. – М. : Cnews, 2011. – 282 с.

4. Фингар, П. Облачные вычисления − бизнес-платформа XXI века / П. Фингар. − М. : Аквамариновая Книга, 2011. – 256 с.

5. Туманов, Ю. М. Защита сред облачных вычислений путем верификации программного обеспечения на наличие деструктивных свойств : автореф. дис. 05.13.19. / Ю. М. Туманов. – М, 2012. – 18 с.

6. Лукашин, А. А. Система защиты информационного взаимодействия в среде облачных вычислений : автореф. дис. 05.13.19 / А. А. Лукашин. – СПб., 2012. – 18 с.

7. Вишняков, В. А. Информационный менеджмент : учеб. пособие / В. А. Вишняков. – Минск : Изд-во МИУ, 2015. – 305 с.

8. Вишняков, В. А. Информационное управление и безопасность: методы, модели, программно-аппаратные решения / В. А. Вишняков. – Минск : Из-во МИУ, 2014. – 287с.

9. Черняк, Л. Платформа Интернета вещей / Л. Черняк // Открытые системы. СУБД. – 2012. – № 7. – С. 1–6.

10. XaaS «Что угодно как услуга» [Электронный ресурс]. – Режим доступа: https://translated.turbopages.org/proxy_u/en-ru.ru.5eba7ac5-66424188-1382b1a0-74722d776562/https/kinsta.com/blog/xaas/. – Дата доступа: 15.05.2024.

Заключение

В заключение поговорим об ограничении ИС и их перспективах в будущем. Попытки создания машинного интеллекта, близкого к человеческому (ЧИ), потерпели ряд неудач, хотя ученые считали, что эта проблема решается с увеличением мощности и памяти компьютеров. Это было связано с несколькими причинами. Рассмотрим их подробнее [1].

Обучение и адаптация. Человеческий интеллект обладает редким даром – способностью к обучению и адаптацией к изменениям. В нем происходит обработка сигналов от пяти сенсорных систем: зрительной, слуховой, осязательной, обонятельной и вкусовой, причем по общим принципам в коре мозга. Так создается и поддерживается модель мира. Базовой основой интеллекта не являются каналы поступления информации, а ее запоминание и обработка, что создает образы модели мира. Так, например, письменный и разговорный языки воспринимаются на уровне чувств по-разному, но на уровне центральной обработки в неокортексе (основы коры мозга) они работают почти одинаково с точки зрения понимания смысла.

Другой характер решений. Аналогия между компьютером и мозгом человека еще некорректна потому, поскольку нейроны мозга работают гораздо медленнее (на пять и более порядков). Но при решении задачи распознавания мозг решает эту задачу за 100 шагов, а компьютер − за миллион и более, независимо от скорости и распараллеливания операций. Мозг просто извлекает из памяти одно из похожих готовых решений. В компьютерах тоже есть память, но человеческая отличается по ряду признаков. Это отличительное запоминание характеризуется: последовательностями хранения элементов внешнего мина, а не отдельными элементами, последовательности вспоминаются ассоциативно и хранятся в инвариантной форме иерархически. Запоминание о событиях мира хранится у человека в серийной форме, и также может быть воспроизведено. Запоминание событий, книг, песен – пример такого вида памяти. Это справедливо и для сенсорных запоминаний. Все воспоминания человека хранятся в синоптических связях (аксонах) между нейронами. Нейронные синапсы хранят миллионы воспоминаний, используемых редко или не используемых совсем. В тоже время в памяти компьютера редко хранятся последовательности сигналов.

Ассоциативный характер памяти естественного интеллекта способен воспроизвести последовательности событий в условиях неполной или искаженной информации как для пространственной, так и временной последовательности входных сигналов. В любой момент времени часть информации может активизировать целое, в этом суть ассоциаций. Еще одна особенность естественной памяти − ее инвариантность. При копировании в компьютере искажение даже одного бита приводит к фатальному сбою программы. Естественный интеллект запоминает важные составляющие модели мира, не привязывается к незначительным его элементам. Хотя искусственная ассоциативная память реализована на базе нейронных сетей, в ней не задействованы инвариантные представления. Как это формируется в коре мозга, остается нерешенной задачей.

Динамичность внешнего мира. Еще философ Платон, решая проблему восприятия мира человеком, заметил, что в мире нет двух одинаковых вещей и они несовершенны по своей природе. Он высказал мистическое предположение, что человек при рождении получает душу, которая в другом измерении познает модели мира (Формы). Обучение и понимание возможны за счет сопоставления несовершенных форм реального мира с идеальными Формами. Но если отбросить элементы мистики, Платон имел в виду инвариантность мышления.

Творчество. Его можно определит как способности прогнозирования на основе аналогий. Прогнозирование – это применение инвариантных последовательностей к новым ситуациям. При решении сложных задач сначала надо настроиться на то, что решение возможно. Второе, не надо останавливать блуждание мысли, не зацикливаться на одном решении. Рассмотреть задачу под другим углом, не впадать в заблуждения.

Сознание распадается на обычное и квалию. Первое связывают с декларативной памятью и способностью пересказать прошлое. У людей, страдающих синестезией, размыты различия между чувствами восприятия. Звуки могут иметь цвет или вкус. Зрительное, слуховое и осязательное восприятия имеют одинаковые принципы формирования сигналов, которые по разному обрабатываются в коре мозга. Зрительный нерв, состоящий из миллиона волокон, воспринимает преимущественно пространственную информацию, а слуховой, включающий 30 тысяч волокон, выражает приоритет временной информации. Эти различия связаны с тем, что называют квалия.

С сознанием связывают понятия души и разума. Некоторые наши мысли не связаны с окружающим миром и являются продуктами сформированной нашей модели мира. Даже наше тело является частью мира. Мозг познает мир через сигналы сенсорной системы. Разум независим от тела, является продуктом деятельности мозга. Это подтверждается травмами (болит несуществующая конечность). Люди, имеющие болезнь мозга, теряют разум (болезнь Айцгеймера), хотя тело здорово.

Воображение это обратная связь от органов чувств вверх в слоях коры мозга, а потом вниз. Оно основано на разворачивании прогнозов и восприятии их в качестве входных сигналов. По сути это вариант планирования. Восприятие не только построено на информации, получаемой от органов чувств, а может генерироваться моделью памяти. Многие особенности мира последовательны и формируют одинаковые модели (время суток, периоды года). Но формирование модели может иметь особенности, связанные с окружающей средой. Люди Востока и Запада по-разному воспринимают пространство и объекты. Не существует абсолютной универсальной модели мира для большинства людей. У каждого индивидуума она зависит от среды, воспитания, образования и т. д.

Развитие интеллекта. Три составляющие мозга − это ганглии (двигательная система), мозжечок (временные последовательности) и гиппоками (конкретные события). Первое отличие ЧИ от животных – сложная иерархическая структура мира в процессе длительного обучения. Второе отличие – язык, система речи. В развитии естественного интеллекта выделяются три этапа. На

первом знания передавались через ДНК, на втором через модифицируемые нервные системы, особи воспринимали модель мира и адаптировались к ней в течение жизни. Третьего развития интеллекта достиг только человек за счет системы речи. Это позволяет передавать знания потомкам и получать их от предков.

Ограничения интеллектуальных систем. Роботы и интеллектуальные системы обладают существенным недостатком − инвариантные задачи для них практически неразрешимы. Внешний мир обладает динамичностью. Одним из подходов для познания такой динамичности в естественном интеллекте является нахождение инвариантной структуры для переменного потока информации. Однако такая структура еще не является базой для прогнозов. Необходимо ее постоянное сопоставление с текущей реальностью.

Сопоставление прогнозов. Три особенности естественной памяти (сохранение последовательности символов от органов чувств, ассоциативное запоминание и инвариантное представление) являются основой предсказания будущего на основе знания о прошлом. Процесс размышления связан с составлением прогнозов о важных структурах окружающего мира. Процесс восприятия действительности мира это комбинация восприятий и прогнозов мозга на основе воспоминаний. Прогнозы это мысли, а в сочетании с сенсорными сигналами − восприятие мира и поведение в нем. Прогнозирование, а не поведение является свидетельством разума.

Иерархическая структура мира соответствует иерархической структуре коры мозга человека. Близкая к ней модель фреймов. Человеку от природы не дано знание образов окружающего мира, языка. Но встроенный механизм самообучения, который выявляет иерархические структуры и запоминает их. Запоминание последовательностей это основной компонент формирования репрезентаций объектов мира. Мозг хранит иерархии последовательностей. О мере изучения структуры объектов с высших слоев мозга они перемещаются вниз, освобождая верхние слои для более сложных связей и структур. Со временем знания передвигаются вниз по иерархии, структуры высших порядков делают человека экспертом.

Рассмотренные отдельные ограничения интеллектуальных систем требуют создания иерархической многосвязанной ассоциативной памяти либо в кремнии (на данный момент нет таких технологий), либо программными средствами (глубинные нейронные системы).

Новые интеллектуальные системы (НИС) [1; 2] будут отличаться от человека. Но они будут иметь память и обработчик, аналогичные тем, что существуют в человеческом мозге. НИС будут требовать обучения (как дети), в процессе которого будет создаваться модель их собственного мира (с помощью человека). Потом НИС будут строить аналогии на основании прошлого, прогнозировать будущие события и решать новые задачи. Такие НИС могут встраиваться в сложные системы (машины, корабли, самолеты, здания). Память может быть удалена от сенсоров. Ее интеллект может определяться прогностической способностью иерархической памяти.

Создание человекоподобных роботов проблематично. Во-первых, человек сложное биологическое и социальное создание, а не только интеллект. Во-вторых, создание таких систем экономически нецелесообразно. НИС будут обладать эквивалентом ЧИ и набором сенсорных систем. Главной трудностью при создании НИС будет система иерархической памяти, которая решаема при помощи связанности и емкости. Для создания такой памяти необходима емкость порядка $8 \cdot 10^4$ млрд байт, что эквивалентно емкости нескольких десятков современных винчестеров. Кстати, для создания НИС память может быть и меньшей. Со связанностью сложней. Клетка ЧМ связана с 5–10 тысячами других. Для современных кремниевых технологий эта задача пока не решаема.

Применениями НИС могут быть система распознавания и перевода речи, восприятия и обработка зрительных сцен в реальном времени, управление сложными движущимися системами со слуховым и зрительным восприятием, многофункциональные системы безопасности.

Отдельные показатели НИС превзойдут ЧМ по быстроте, емкости, реплицитивности и сенсорным устройствам. Быстродействие кремниевого разума будет на пять-шесть порядков выше биологического, что приведет к более быстрому решению задач, принятию решений, созданию огромных хранилищ информации. Большая емкость приведет к более глубокому пониманию отдельных сущностей мира. Развитие человеческого разума требует десятилетий, кремниевый можно будет воспроизводить намного быстрее.

НИС могли бы воспринимать мир всеми существующими системами ощущений, например, гидролокаторами, радарами, приборами ночного видения и т. д. Посредством НИС с такими сенсорами можно эффективно управлять климатом, миграцией животных, распространением бактерий и вирусов. НИС могут обладать сверхтонкими ощущениями для использования в медицине, фармакологии и биологии. Еще один вид НИС может работать в виртуальном мире математики и физики, понимать многомерные пространства.

1. Хоккинс, Д. Об интеллекте / Д. Хоккинс, С. Бэйксли. – М. : Вильямс, 2015. – 240 с.

2. Вишняков, В. А. Развитие интеллектуального управления с использованием облачных технологий / В. А. Вишняков // Информатика, 2016. – № 2. – С. 113–120.

Св. план 2024, поз. 16

Учебное издание

Вишняков Владимир Анатольевич

ИНТЕЛЛЕКТУАЛЬНЫЕ ТЕХНОЛОГИИ В ИНФОКОММУНИКАЦИЯХ

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

В авторской редакции

Корректор *Е. Н. Батурчик*
Компьютерная правка, оригинал-макет *А. А. Лущикова*

Подписано в печать 26.11.2024. Формат 60×84 1/16. Бумага офсетная. Гарнитура «Таймс».
Отпечатано на ризографе. Усл. печ. л. 15,69. Уч.-изд. л. 17,5. Тираж 50 экз. Заказ 93.

Издатель и полиграфическое исполнение: учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий №1/238 от 24.03.2014,
№2/113 от 07.04.2014, №3/615 от 07.04.2014.
Ул. П. Бровки, 6, 220013, г. Минск