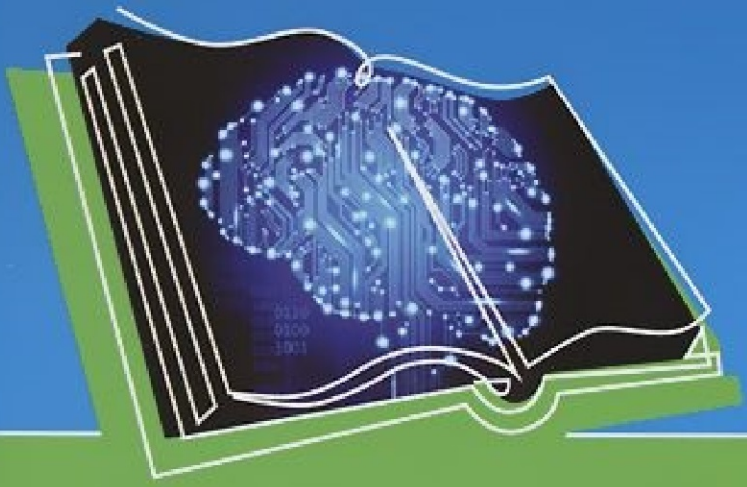
Высшее профессиональное образование

Учебник

Б. Я. Советов
В. В. Цехановский
В. Д. Чертовской

ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ И ТЕХНОЛОГИИ



БАКАЛАВРИАТ

ИНФОРМАТИКА
И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

ACADEMA

Высшее профессиональное образование
БАКАЛАВРИАТ

Б. Я. СОВЕТОВ, В. В. ЦЕХАНОВСКИЙ,
В. Д. ЧЕРТОВСКОЙ

ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ И ТЕХНОЛОГИИ

Учебник

Допущено
Учебно-методическим объединением
вузов по университетскому политехническому
образованию в качестве учебника для студентов
высших учебных заведений, обучающихся
по направлению подготовки 230400
«Информационные системы и технологии»

ACADEMIA

Москва
Издательский центр «Академия»
2013

УДК 004.89(075.8)
ББК 32.813я73
С56

Рецензенты:
директор Санкт-Петербургского института информатики и автоматизации Российской академии наук, д-р техн. наук, проф. *Р. М. Юсупов*;
проф. кафедры вычислительных систем и сетей Санкт-Петербургского государственного университета аэрокосмического приборостроения, д-р техн. наук *М. Б. Игнатьев*

Советов Б. Я.

С56 Интеллектуальные системы и технологии: учебник для студ. учреждений высш. проф. образования / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. — М. : Издательский центр «Академия», 2013. — 320 с. — (Сер. Бакалавриат).

ISBN 978-5-7695-9572-1

Учебник создан в соответствии с Федеральным государственным образовательным стандартом по направлению подготовки 230400 «Информационные системы и технологии» (квалификация «бакалавр»).

Рассмотрен комплекс проблем по разработке, функционированию и проектированию систем и технологий искусственного интеллекта. Изложены теоретические и прикладные вопросы представления знаний в информационных системах, идеология построения интеллектуальных систем и технологий. Раскрыт математический аппарат представления знаний. Рассмотрены возможности и пути использования искусственного интеллекта при проектировании информационных систем, новые аспекты представления знаний на основе искусственных нейронных сетей, расчетно-логических систем, генетических алгоритмов, мультиагентных систем. Большое внимание уделено использованию прикладных интеллектуальных технологий. Приведены примеры построения интеллектуальных систем.

Для студентов учреждений высшего профессионального образования

УДК 004.89(075.8)
ББК 32.813я73

ISBN 978-5-7695-9572-1

ВВЕДЕНИЕ

В последнее время интеллектуальные системы (ИС) получили распространение в системах управления, имеющих сложную структуру с обратной связью, и привели к интересным результатам. Стало возможным не только учесть и «переложить» на компьютер богатейшие знания лиц, принимающих решения (ЛПР), и экспертов, но и использовать полученные знания для обучения и повышения квалификации специалистов. Одновременно был осуществлен прорыв в системе программирования путем создания новых декларативных языков, программных продуктов и приложений. Это относится прежде всего к экспертным системам [5, 13, 14, 18, 20, 50, 53, 57, 58].

Авторы предприняли попытку отразить современные достижения в области интеллектуальных систем на основе информационных технологий, что обеспечивает переход к промышленным методам и средствам работы с информацией в различных сферах человеческой деятельности. Это дает возможность расширения существующих методов и средств реализации систем искусственного интеллекта и перехода к принципиально новым, основанным на высоких технологиях.

Стремительный рост сложности информационных процессов в различных предметных областях требует использования новых интеллектуальных подходов, основанных на обработке знаний. Однако успешному внедрению интеллектуальных информационных систем препятствует слабый уровень использования информационных технологий. Исходя из этого, основная цель данной технологии — достижение уровня реальных промышленных приложений в области технологий искусственного интеллекта.

Фундаментальные положения ИС все время пополняются новыми результатами [4, 9, 26, 30, 36, 55, 56] во многих публикациях. Большинство из них — в виде статей в различных, часто трудно доступных для студентов журналах. Статьи чаще всего преследуют научные, а не учебные цели. Сложность восприятия, а тем более — систематизация новых знаний студентами связаны с большим разнообразием классов интеллектуальных систем, при этом в каждом, особенно новом, классе используются специфические терминология и математический аппарат.

Таким образом, имеется настоятельная необходимость систематизации материалов по интеллектуальным системам.

В данном учебнике впервые системно рассмотрены такие классы, как искусственные нейронные сети, генетические алгоритмы, многоагентные системы, системы на естественном языке, адаптивные системы.

Учебник состоит из введения, трех разделов, объединяющих девять глав, и приложения.

Раздел I (главы 1, 2) посвящен описанию состава и структуры интеллектуальных систем.

В разделе II (главы 3—7) рассмотрены технологии создания интеллектуальных систем.

Раздел III (главы 8, 9) раскрывает прикладные технологии в рамках систем поддержки и принятия решений (СППР).

В приложении в качестве примера приведена программная реализация экспертных систем на различных языках программирования.

Методически учебник включает контрольные вопросы по каждой главе, что даст возможность студенту проверить качество усвоения материала.

Авторы надеются, что данный учебник будет способствовать повышению качества подготовки бакалавров, а также будет полезен всем читателям, интересующимся современным состоянием и перспективами развития информационных систем и технологий.

РАЗДЕЛ I
ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ

Глава 1
ИНТЕЛЛЕКТУАЛЬНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ

1.1. Понятие искусственного интеллекта

При изучении курса «Интеллектуальные системы и технологии» пользуются специальной терминологией [4, 9, 12, 16, 26, 30, 32, 34—36, 42, 51, 54—56, 62, 66, 70, 72—74], изложенной далее.

Под ***системой*** понимается совокупность взаимодействующих структурных элементов в соответствии с поставленной целью.

Структура — совокупность элементов и их связей.

Таким образом, понятие «структура» отражает статику процесса, а система — его динамику.

Интеллект — мыслительная способность человека.

Мышление — способность человека с помощью размышлений и последовательных мыслительных действий получать желаемые результаты.

Под ***искусственным интеллектом*** понимают создание вычислительной системы, имитирующей человеческие навыки, системы, которая перерабатывала бы информацию на уровне и по законам человеческого мозга (план-максимум).

Интеллектуальная система (ИС) — техническая и программная система, способная решать задачи, которые традиционно считались творческими.

Обратим внимание на то, что речь идет не о ручном, а о компьютерном варианте информационных систем. ИС оперирует следующими понятиями.

Данные — совокупность объективных сведений.

Информация — сведения, неизвестные ранее получателю информации, пополняющие его знания, подтверждающие или опровергающие положения и соответствующие убеждения.

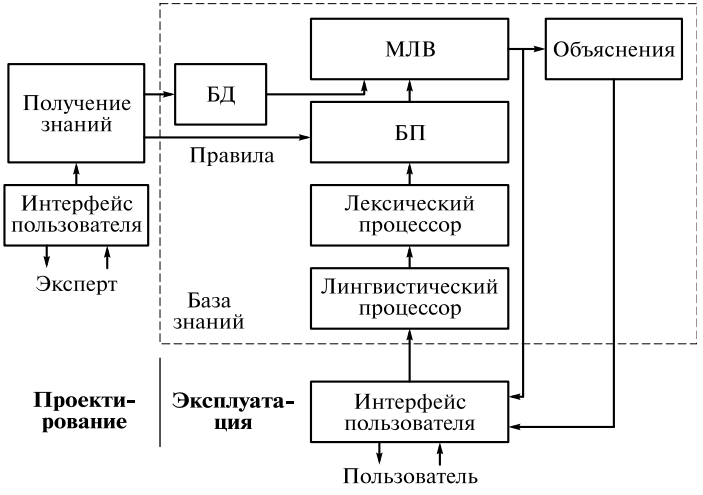


Рис. 1.1. Схема интеллектуальной системы:

МЛВ — машина логического вывода; БД — база данных; БП — база правил

Знания — это совокупность фактов, закономерностей и эвристических правил, с помощью которых решается поставленная задача.

Для любой системы выделяют процессы проектирования и эксплуатации.

Процесс проектирования по своему назначению является интеллектуальным и включает формирование цели, технологию и инструментарий. В проектировании присутствует много ручных операций, хотя в последнее время все шире используются системы автоматизированного проектирования.

Когда говорят об интеллектуальных системах, то имеют в виду интеллектуальность, проявляющуюся в процессе эксплуатации системы.

Общая схема интеллектуальный системы показана на рис. 1.1.

Блок Интерфейс пользователя предназначен для связи компьютера с пользователем, для которого предпочтительным языком «разговора» является естественный язык или близкий к нему.

Следует отметить специфическое понятие «естественный язык», под которым понимается фактически искусственный язык, полученный на этапе концептуализации из естественного человеческого языка (русского, английского или любого другого) путем удаления неоднозначностей (синонимов, омонимов, идиоматических выражений). Следует заметить, что такой язык в теории автоматизированных систем носит название «искусственный язык» или «информационный язык».

Основу ИС составляют блоки База данных (БД), База правил (БП), Машина логического вывода (МЛВ), Объяснения. БД хранит исходные данные. В базе правил фиксируются знания и опыт эксперта. МЛВ выводит результат, взаимодействуя с БД и БП. Объяснения выводят систему правил, использованных для полученного конкретного результата, на естественном языке.

Все три «компьютерных» блока должны быть описаны математически. Правила записываются через интерфейс эксперта (см. рис. 1.1) в виде сложных правильно построенных формул (ППФ) при посредничестве инженера по знаниям. Форма, в которой записываются ППФ, определяется *синтаксисом*, а истинностное значение алгоритмов логического вывода *семантикой*, или смыслом.

По запросу пользователя компьютер с помощью блока Объяснения может дать ответ на вопрос «как» (с помощью набора каких правил) получен результат и «почему» компьютер задает пользователю уточняющие вопросы, связанные, как правило, с данными.

Для стыковки интерфейса пользователя, работающего на естественном языке, с перечисленными блоками используют лингвистический и лексический процессоры.

Лингвистический процессор — непосредственно работает с естественным языком, преобразуя результат в «машинный вид».

Лексический процессор — словарный состав информационного языка.

Иногда включают и синтаксический процессор, при этом под синтаксисом понимают правила сочетания слов внутри предложения и построения предложений.

В блоке Получение знаний выделяют два понятия:

- если информация поступает из книг (документов), то говорят о выявлении знаний;
- если информация получается на основе работы эксперта, то говорят об извлечении знаний.

Преобразование инженером по знаниям полученной информации называется *получением знаний*.

Сложность заключается в том, что эксперт может не владеть языками программирования, в то время как инженер по знаниям может недостаточно ориентироваться в данной предметной области. При этом в наиболее развитых моделях выделяют экстенсиональную и интенсиональную компоненты.

Экстенсиональная компонента — различные сведения о предметной области (данные).

Интенсиональная компонента — схемы связей между атрибутами, отражающие основные закономерности системы в предметной области решаемых функциональных задач. Для разработчиков моделей данные — это схема БД. Для представителей ИИ — это знания о проблемной области.

1.2. Классификация интеллектуальных систем

Понятие «интеллект» многогранно, поэтому существует значительное количество разновидностей интеллектуальных систем, которые целесообразно классифицировать, т.е. разделять множество на подмножества по принятому классификационному признаку.

Из множеств классификаций интеллектуальных систем [3, 10, 60] воспользуемся классификацией, показанной на рис. 1.2.

Системы на естественном языке СЕЯ специфичны и предназначены преимущественно для таких целей, как машинный перевод, генерация документов, автоматическое аннотирование и реферирование.

Экспертные системы ЭС предполагают высокую степень формализации процессов на этапе концептуализации.

Разновидностью экспертных систем можно считать расчетно-логические системы РЛС, оперирующими с функциями вместо правил.

Искусственные нейронные сети ИНС фактически представляют собой разновидность систем автоматического управления, использующие свойства нейрона.

Системы с генетическими алгоритмами СГА относятся к разновидности эволюционных эвристических методов.

Многоагентные системы МАС преследуют цель согласования теорий баз данных и баз знаний.

Интеллектуальные системы управления ИСУ используют режим адаптации к изменяющимся параметрам и целям эксплуатации (функционирования) систем.

В дальнейшем будут рассмотрены все перечисленные выше классы систем.

По характеру математического описания интеллектуальные системы можно разделить на непрерывные и дискретные (рис. 1.3).

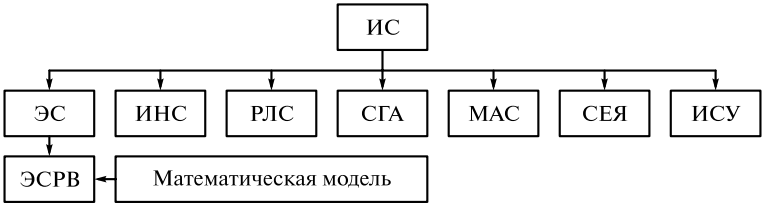


Рис. 1.2. Классификация интеллектуальных систем (ИС):

ЭС — экспертные системы; ИНС — искусственные нейронные сети; РЛС — расчетно-логические системы; СГА — системы с генетическими алгоритмами; МАС — многоагентные системы; СЕЯ — системы на естественном языке; ИСУ — интеллектуальные системы управления; ЭСРВ — экспертные системы реального времени

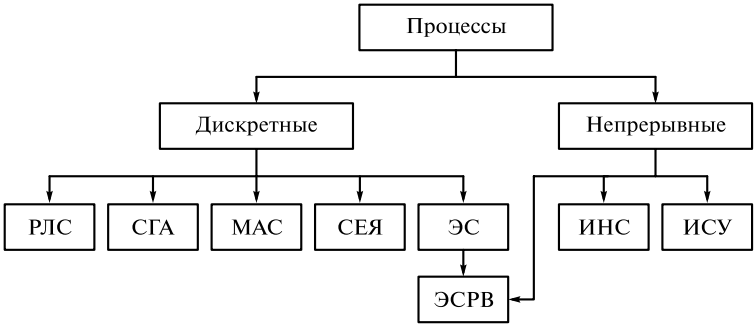


Рис. 1.3. Методы математического описания

Непрерывные процессы имеют место в ИНС, ИСУ, РЛС и описываются дифференциальными уравнениями с соответствующими критериями, функциональными (формульными) зависимостями. В этом случае используются преимущественно числовые данные, хотя они могут быть искусственно организованы в виде такой структуры, как базы данных.

Интеллектуальные процессы человека чаще связаны с дискретными данными и знаниями, опирающимися на информационные языки. В связи с этим ИС оперируют с дискретными величинами (классы ЭС, СГА, МАС, СЕЯ). Однако и в этих классах не существует однородного математического описания. Здесь используются такие методы, как логическое описание, нечеткие множества, функциональные семантические сети, контекстно-зависимые грамматики.

Имеется к тому же все большая необходимость связи дискретных и непрерывных величин, что привело к появлению гибридных моделей. Сюда относятся, прежде всего, экспертные системы реального времени (ЭСРВ), в которых имеется непрерывная составляющая, описываемая дифференциальными или разностными уравнениями.

1.3. Технология проектирования и эксплуатации интеллектуальных систем

Технология — порядок (последовательность) взаимодействия структурных элементов системы [1, 22, 44, 49, 52, 65, 68].

Понятия «технология» и «алгоритм» схожи. Будем полагать, что алгоритм относится к процессам меньшего масштаба. Иными словами, технология может состоять из совокупности алгоритмов.

Интегральный характер носит технология проектирования и эксплуатации системы. Эта технология включает в себя следующие этапы:

1) идентификация;
2) концептуализация;
3) формализация;
4) реализация (выполнение);
5) отладка и тестирование;
6) опытная эксплуатация.

Идентификация. Цель этапа — получение словесной модели и структуры системы. На данном этапе можно выделить такие работы:

- формирование целей проектирования;
- формирование коллектива разработчиков;
- определение ресурсов для разработки;
- идентификация проблемы.

При *формировании целей проектирования* следует отличать цель построения системы от цели, имеющей место в спроектированной системе при ее работе.

Цель работы спроектированной системы — выработка решений-советов, анализ и учет последствий принимаемых решений.

Целями проектирования могут быть формализация трудноформализуемых задач, неформальных знаний эксперта; улучшение качества решений экспертов; автоматизация интеллектуальных аспектов работы руководителя.

Далее сосредоточимся на последней цели, в которой фактически сочетаются две первые цели. Среди руководителей выделим конечного пользователя (КП) — руководителя, который будет использовать создаваемую систему.

На этапе *формирования коллектива разработчиков* участвуют несколько экспертов (руководителей, конечных пользователей) в роли учителей и инженер по знаниям в качестве ученика, осваивающего словесную модель.

При *определении ресурсов для разработки* следует учитывать ограничения в процессе проектирования, к которым относятся размеры финансирования работ; сроки выполнения проекта (предварительные результаты — через 3 мес, первые уточненные результаты — через 6 мес); источники знаний; вычислительные средства.

В процессе *идентификации проблемы* собственно идентификация предполагает получение ответов на следующие вопросы:

- какой класс задач решает система;
- как эти задачи могут быть определены;
- каковы основные понятия в предметной области;
- каковы трудности в решении выделенного класса задач?

В самой процедуре идентификации выделяют процессы извлечения знаний эксперта (в беседе с инженером по знаниям), выявление знаний инженером по знаниям из книг и документов, приобретение знаний (трансформация вербальной модели знаний в формальную).

Возможны следующие подходы к извлечению знаний:

а) долговременная работа с экспертами;

б) оперативное создание прототипа системы путем получения инженером по знаниям первоначальных знаний и предварительной структуризации процедуры извлечения;

в) анализ начальных и полученных знаний.

В качестве методов извлечения можно использовать:

а) опрос эксперта по общему вопроснику с наводящими вопросами;

б) структуризованный опрос;

в) самонаблюдение с размышлением эксперта вслух;

г) самоотчет эксперта при выполнении работы;

д) критический обзор извлеченных знаний в целях уменьшения грубых ошибок и формирования метазнаний.

Детально процесс извлечения знаний описан в работе [13].

В процессе извлечения знания получаются чаще всего в неформальном словесном виде. Основная задача инженера по знаниям — перейти от словесного описания к формальному с учетом используемого программного инструмента реализации (язык программирования, оболочка ЭС).

Концептуализация. Цель этапа — получение системы ключевых понятий, необходимых для последующего формального описания процессов в системе.

На этом этапе исходят из понятия «черный ящик», для которого проводится детализация входов и выходов, выявление алгоритмов, разделение системы на подсистемы, установление информационных связей между подсистемами. Иными словами, формируется структура исследуемой системы.

Полезно составить словарь (онтологию) используемых терминов информационного языка, обращая особое внимание на исключение неоднозначностей (синонимы, омонимы, идиоматические выражения). Для синонимов часто составляют специальный словарь.

Саму систему правил (дискретная составляющая описания управляющей части системы) выявляют в процессе беседы инженера по знаниям с экспертом — конечным пользователем по схеме: «Ваше подразделение не выполнило план с начала месяца и за прошедший день — ваши действия?». Как правило, формулировка ответов будет иметь вид «Если недостаточно такого-то ресурса, то действие таково».

Целесообразно составить протокол рассуждений и действий эксперта; определить связь системы правил с непрерывной составляющей описания управляющей части системы

Само действие КП может характеризоваться непрерывным описанием или переходом к нему.

Формализация. Цель этапа — переход от вербального (словесного) описания процессов в системе к их описанию на некотором формальном языке.

Формальное представление связано со структуризацией. В общем случае выделяют структуризацию исходной задачи, предметной области, базы правил, приложения.

Структуризация исходной задачи подразумевает выявление многоуровневой системы управления, разделение системы на функциональные и программные модули и определение правил «сборки» системы из модулей.

Структуризация предметной области предполагает формирование иерархии классов понятий с их свойствами (летательный аппарат как исходный класс, самолет — родительский класс).

Структуризация базы правил предполагает построение иерархии правил.

Структуризация приложений подразумевает иерархию рабочих пространств и программных шаблонов.

Одновременно оценивается достоверность исходных данных и правил. Для этих целей можно использовать аппарат факторов уверенности или нечетких переменных [35].

Здесь могут решаться следующие проблемы:

1) выбор алгебры оперирования с нечеткими переменными;

2) построение модели настройки на конкретного КП и анализ его предпочтений;

3) сопоставление предпочтениям ЛПР критериев оптимальности.

Реализация (выполнение). Цель этапа — создание прототипа системы. Это соответствует этапу технического проектирования АСУ. На нем проверяются основные технические решения, при этом могут не учитываться сложные причинно-временны́е соотношения, неточность данных. Проводится корректировка правил, дополнительные детали в которые «добавляются» разработчиком после отладки и тестирования системы, учета замечаний пользователя.

Таким образом, реализация является в общем случае процедурой итеративной.

Отладка и тестирование. На этом этапе система реализуется в полном объеме, что соответствует этапу рабочего проектирования АСУ.

Тестирование системы понимается в широком смысле, предполагая тестирование выявленных алгоритмов до их реализации и тестирование алгоритмов в форме реализованных на компьютере программ.

Первая разновидность тестирования — это тестирование исходных данных (в том числе их достоверности), логическое тестирование (системы продукций с точки зрения избыточных, цикличных или конфликтных пропущенных правил, пропущенных и пересекающихся правил, пересекающихся правил, несогласующихся условий, достоверности алгоритмов), концептуальное тестирование (проверка удовлетворения общих требований, предъявленных к системе).

Проводится тестирование отдельных программных модулей и шаблонов, программы в целом, разрабатываются входные сценарии, результаты введения которых в программу известны, для проверки работы системы правил.

Для упрощения тестирования отдельных модулей полезна их автономная отладка в виде компонент или шаблонов.

Процедура стыковки программ с данными внешних устройств (например, датчиками) называется ***инспекцией***. Для процедуры инспекции могут разрабатываться специальные программы (поиска местонахождения элементов на основе их типов, принадлежности к классу, атрибутов). Они должны позволять получать характеристики внешних устройств как в автономном режиме, так и при их работе в системе в целом.

Для отладки используются специальные средства (сообщение об ошибках работы системы, пошаговое выполнение алгоритма, подсветка возбужденного правила), придаваемые, как правило, базовым программным продуктам. Отладочные средства тесно взаимодействуют со средствами инспекции.

Опытная эксплуатация. Разработанная система предъявляется заказчику, который оценивает ее обычно по признакам пригодности системы как суммы полезности системы и удобства работы с ней.

Полезность характеризуется степенью удовлетворения требований (потребностей) пользователя, прежде всего, в сбойных ситуациях.

Удобство работы определяется свойствами интерфейсов пользователя, включая гибкость (настройка на различных пользователей или на изменение квалификации одного и того же пользователя), устойчивость к ошибкам оператора (целостность системы).

Выделяют тренажерное тестирование, пилотное сопровождение и практическую эксплуатацию.

К тренажерному (модельному) тестированию предъявляются более жесткие требования, чем к практической эксплуатации (к надежности системы в условиях более частой поломки оборудования, «зашумления» первичных данных датчиков, отказов, в том числе компьютера и периферийных средств).

Пилотное сопровождение предполагает два варианта:

1) подключение к системе входов реальных систем без замыкания обратных связей;

2) поэтапное замыкание обратных связей.

Система постепенно переводится от работы с моделью внешней среды к работе с самой средой.

После завершения пилотного сопровождения, в процессе которого выявляются неполадки и пожелания пользователя, устраняются замечания и начинается эксплуатация системы в полном объеме. Пример прикладного проектирования интеллектуальной системы приведен в подраз. 4.5.

Отметим, что в разных классах интеллектуальных систем детальность проработки отдельных этапов технологии может быть разной. Детальная этапность имеет место для экспертных систем, поскольку это наиболее развитые системы.

Технологию проектирования и эксплуатации интеллектуальных систем можно представить в виде совокупности трех технологий: создание, использование и функционирование интеллектуальных систем.

Технология создания определяется этапами идентификации, концептуализации и формализации. Этап концептуализации специфичен и характерен только для интеллектуальных систем. Он связан с понятием «онтология», которое подробно рассмотрим позднее.

Технология использования, представленная соответствующими инструкциями, формируется на этапах реализации, отладки и корректируется на этапе опытной эксплуатации.

Технология функционирования связана с этапом формализации.

Технологию создания (проектирования) можно трактовать и иначе. На стадии идентификации определяется структура системы в соответствии с ее назначением с учетом особенностей системы, на стадиях концептуализации и формализации — функциональное математическое наполнение структурных элементов (технология функционирования).

При рассмотрении структур классов систем будем придерживаться уровня их развития (исследования).

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Дайте определение понятий «система» и «структура».
2. Что такое интеллект, искусственный интеллект, интеллектуальная система?
3. Приведите общую схему интеллектуальной системы и опишите назначение отдельных блоков.
4. Как классифицируют интеллектуальные системы? Каково назначение отдельных классов?
5. Дайте определение технологии. Чем технология отличается от алгоритма?

Глава 2

КЛАССЫ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

2.1. Экспертные системы

В описании классов интеллектуальных систем следует выделить две составляющих: назначение и структуру; технологию создания систем. В данной главе рассмотрим первую составляющую большинства классов систем. Технологию построения последних обсудим в последующих главах. Исключение составят расчетно-логические системы и системы с генетическими алгоритмами. Теория этих систем только начала складываться. Технология их создания находится практически в зачаточном состоянии и поэтому освещается в данной главе.

Экспертная система (ЭС) — вычислительная система, использующая знания эксперта и процедуры логического вывода и позволяющая дать объяснения полученным результатам. Синонимом ЭС является термин «система, базирующаяся на знаниях» [21, 43, 46, 59, 62].

На начальных этапах развития рассматривались статические ЭС. В таких системах время впрямую не фигурировало.

Изначально предполагалось, что схема ЭС будет иметь вид, показанный на рис. 1.1. Статическая ЭС без блока «база правил» называется оболочкой ЭС (аналог СУБД). Наиболее известная оболочка GURU. Однако выяснилось, особенно после появления оболочки экспертной системы GURU, что построение «переводчика» с естественного языка на язык компьютера является сложной, вполне самостоятельной проблемой. Появление декларативных логических языков типа ПРОЛОГ (Prolog), а затем языков, использующих ПРОЛОГ-идеи, позволило упростить задачу «перевода» при условии использования текстовых элементов из заранее составленного естественного языка после выполнения этапа концептуализации. При этом структура интеллектуальной системы трансформировалась — исчезли блоки лингвистического и лексического процессоров (рис. 2.1).

Первые ЭС были разомкнутыми и использовались чаще всего в диагностике. Сюда следует отнести медицинские системы — MYCIN и EMYCIN. Их характеристиками являются простота общения с ПК, возможность наращивания модулей, интеграция неоднородных данных, решение многокритериальных задач с учетом предпочтений

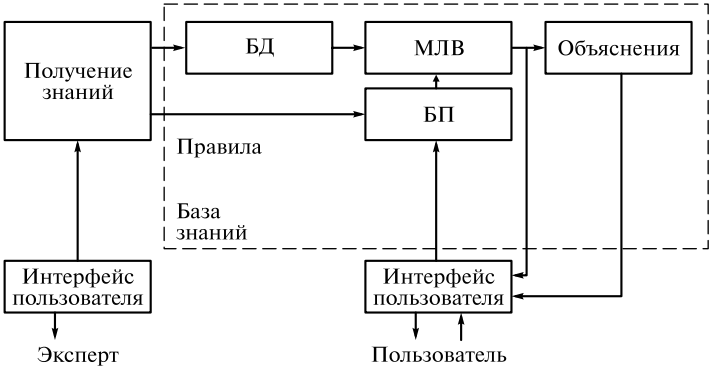


Рис. 2.1. Схема экспертной системы:
МЛВ — машина логического вывода

лица, принимающего решения (ЛПР), документальность, конфиденциальность, унификация формы знаний, независимость механизма логического вывода, возможность представления объяснений, возможность символьного описания процессов.

Такие ЭС могут быть использованы как элементы управляющей части системы управления. Место таких систем в цикле управления показано на рис. 2.2.

Однако статические системы имеют ограниченную сферу применения. Они не позволяют учесть, сохранить и выполнить анализ изменяющихся во времени данных, поступающих от внешних источников; работать с асинхронными процессами; обеспечить механизм

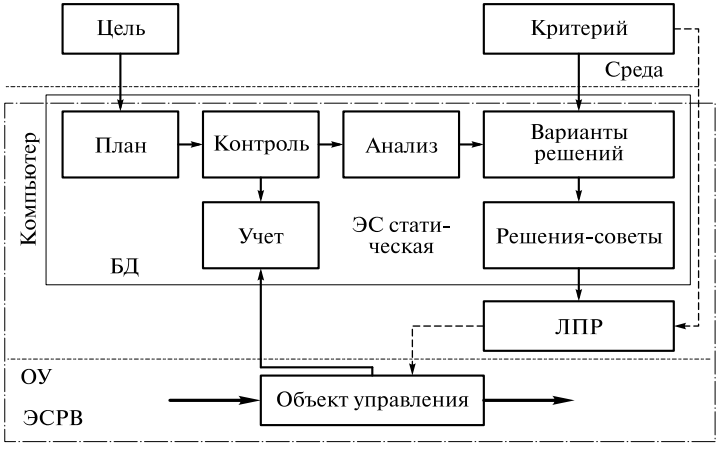


Рис. 2.2. Информационно-советующая система

рассуждений при ограниченных ресурсах; обеспечить прогноз последствий принимаемых решений-советов; обеспечить создание и поддержку интерфейсов пользователей, уровень их защиты для различных категорий пользователей.

Экспертные системы, реализованные в области управления, называются ЭС *реального времени* (ЭСРВ).

Следует отметить, что первоначально ЭСРВ работала с моделью объекта управления (см. рис. 1.1). В настоящее время в ЭСРВ включают реальный объект управления, информация о состоянии которого поставляется системой датчиков и отражается на табло (рис. 2.3). Их место в цикле управления показано на рис. 2.2.

Фирма Gensym в 1988 г. выпустила очень удачный программный продукт под названием G2, получивший в настоящее время очень широкое распространение.

С помощью ЭСРВ создаются эффективные системы управления непрерывными производствами и процессами в химии, фармакологии, производстве цемента, продуктов питания, авиакосмических исследованиях, транспортировке и переработке нефти и газа, управлении тепловыми и атомными станциями, системах связи, финансовых операциях.

Создание ЭС, даже статических, достаточно дорого: от 0,1 до 0,2 млн долларов при трудоемкости от 0,5 до 10 человеколет при количестве учитываемых параметров свыше тысячи. В то же время ис-

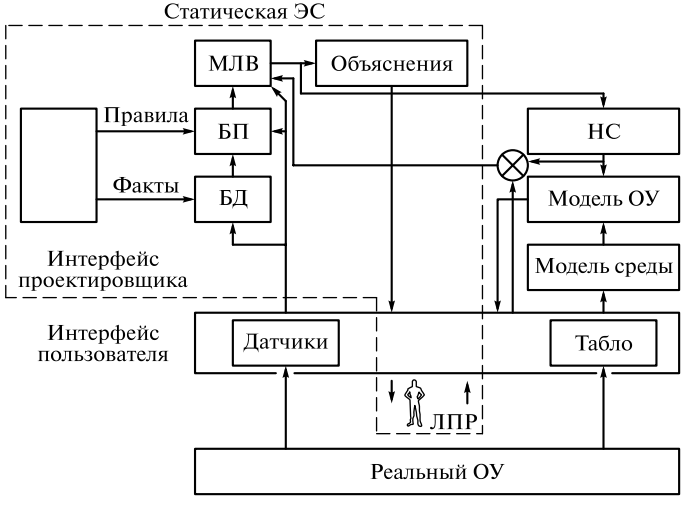


Рис. 2.3. Современная экспертная система реального времени:

БП — база правил; МЛВ — машина логического вывода (дискретная составляющая описания управляющей части); НС — непрерывная составляющая описания управляющей части

пользование ЭС и ЭСРВ в настоящее время приносит значительный экономический эффект. Если в 1988 г. доход от продажи ЭС составлял 3 млн долларов, то в 1993 г. уже 55 млн долларов.

Фирма Sira сократила затраты на строительство трубопровода в Австралии на 40 млн долларов, применив управление трубопроводом с помощью экспертной системы, реализованной с помощью оболочки G2.

В последнее время наиболее значительная доля промышленных ЭС относится к ЭСРВ (в частности, G2). Причинами активного создания ЭСРВ послужили специализация программных средств, повышающая эффективность их использования и сокращающая сроки разработки приложений; использование ПРОЛОГ-идей, что снизило требования к программным приложениям; возможность интегрирования ЭС с другими информационными технологиями; открытость и переносимость разработок путем использования стандартов; применение архитектуры клиент-сервер.

В рамках теории ЭС должны быть решены следующие основные проблемы:

1) возможно ли достичь конечного результата (цели) на основе выбранных правил и данных (проблема выводимости);

2) если возможно достижение цели, то как это сделать (проблема извлечения результата);

3) как оценить достоверность результата, если исходные данные и правила не полностью достоверны (проблема достоверности).

Первая и вторая проблемы решаются сочетанием аппарата описания знаний и аппарата вывода. Для решения третьей проблемы используются различные методы, учитывающие неточность описания (фактор уверенности, нечеткие множества). Технология решения этих проблем более подробно рассмотрена в гл. 4.

2.2. Искусственные нейронные сети

Начиная с 1990-х гг. работы велись в плане поисков методов обучения и построения адаптивных систем. На определенном этапе становления искусственные нейронные сети (ИНС) развивались в противовес ЭСРВ для устранения следующих недостатков:

- сложное сочетание непрерывных и дискретных процессов;
- сложность учета нелинейного характера процесса управления;
- недостаточная адаптация ЭС реального времени к различного рода неопределенностям (параметров структуры).

В дальнейшем выяснилось, что сами ИНС имеют достаточно сложную структуру [25] и их применение для малоразмерных систем управления проблематично.

Основой ИНС является базовый процессорный элемент, который построен как имитация естественного биологического нейрона.

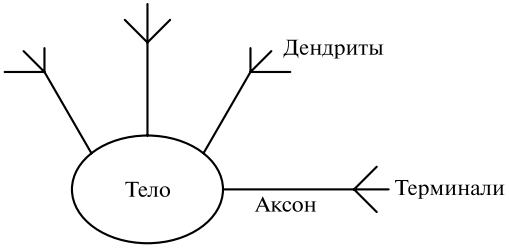


Рис. 2.4. Биологический нейрон

Схема *биологического нейрона* показана на рис. 2.4.

Нейрон (клетка) работает следующим образом. Через дендриты поступают сигналы химического, электрохимического, электрического свойства и раздражают тело клетки. Сигналы преобразуются в разность потенциалов. Если эта разность превышает порог, клетка возбуждается и электрический сигнал через отросток (аксон) и нервные окончания (терминали) передается следующей клетке. Совокупность дендритов и терминалей образует *синопсис*.

Приведенная модель клетки служит прототипом для базового процессорного элемента (БПЭ).

Схема *базового процессорного элемента*, или ADALINE, показана на рис. 2.5. Здесь w_i ($-1 \le w_i \le 1$) — некоторые веса, которые могут изменяться; r_0 — некоторый порог срабатывания элемента; $f(s)$ — соматическая функция, или функция активации. Наиболее распространенными являются следующие виды функций:

- линейные $f(s) = s$;
- экспоненциальная $f(s) = \mathbf{e}^{-\frac{s^2}{\sigma^2}}$, где σ — числовой параметр;
- логистическая кривая $f(s) = \dfrac{1}{1+\mathbf{e}^{-s}}$.

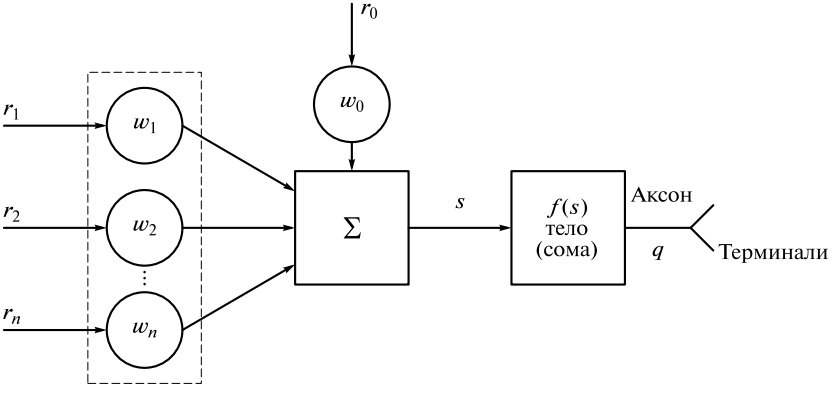


Рис. 2.5. Базовый процессорный элемент (БПЭ)

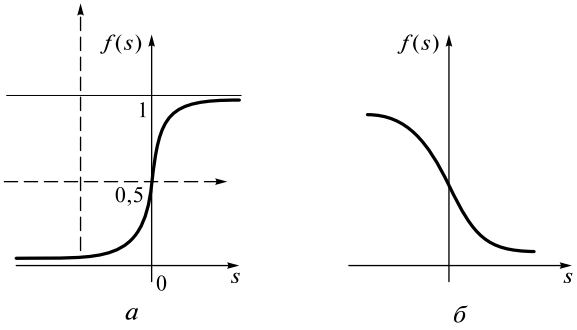


Рис. 2.6. Сигмоидальная функция

Логистическая кривая встречается наиболее часто. Она практически бесконечно дифференцируема и ее производная $f'(s) = 1 + f(s)$. Эту кривую называют также сигмоидальной функцией. Она имеет вид, показанный на рис. 2.6, и может быть записана в такой форме: $f(s) = 1/[-1 + \exp(-s)]$. Если взять два числовых значения $s = s_1$ и $s = s_2$ при $s_1 > s_2$, то первая кривая будет круче в районе $s = 0$. В общем виде $f(s) = b + (1/[1 + \exp(-\{s - a\})])$, где a и b — константы, обеспечиваемые соответствующими смещениями r_0; при $a < 0$ и $b < 0$

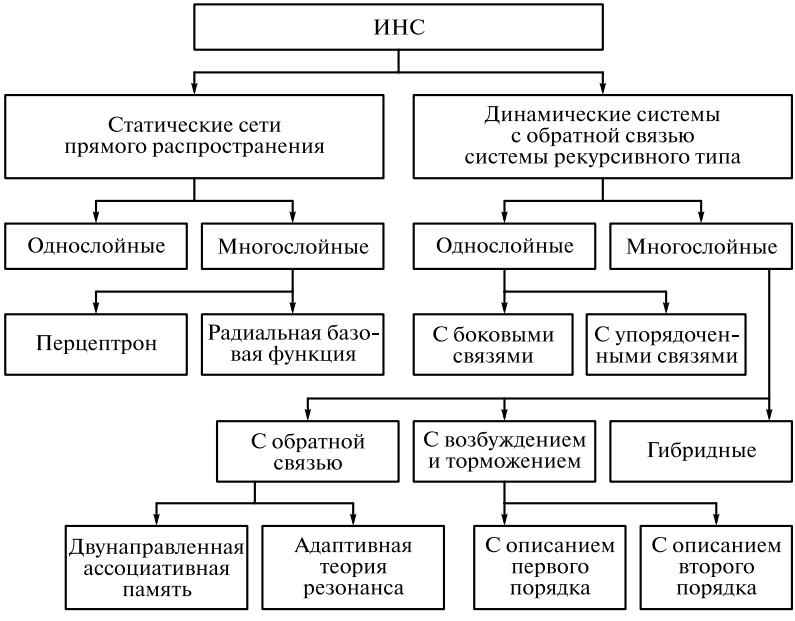


Рис. 2.7. Классификация ИНС

начало координат сместится так, как показано пунктиром на рис. 2.6, *а*. При $f(s) = -1/[1 + \exp(-s)]$ кривая получает вид, показанный на рис. 2.6, *б*. Таким образом, с помощью весов вид функции может быть изменен. Этим свойством пользуются при применении ИНС в диагностике.

В простейшем случае БПЭ описывается выражением

$$q = f(s) = f\left(\sum_{i=1}^{n} w_i r_i + w_0 r_0\right). \qquad (2.1)$$

В дальнейшем изложении БПЭ будет использоваться чаще всего в системах автоматического управления. В качестве регулируемых параметров используются w_i, w_0, $f(s)$. Веса могут изменяться в процессе работы системы, тогда как функции устанавливаются чаще всего только в процессе проектирования.

На основе БПЭ могут быть построены различные виды искусственных нейронных сетей.

Вариант классификации ИНС показан на рис. 2.7. БПЭ представляет собой одномерный элемент ADALINE. Его возможности при линейной соматической функции ограничены. В связи с этим предложили многомерный однослойный БПЭ MADALINE (рис. 2.8). Он описывается в векторно-матричной форме

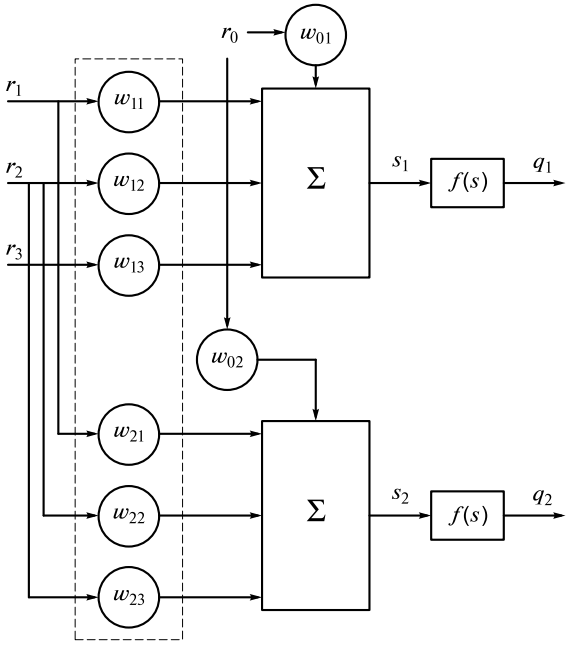


Рис. 2.8. Многомерный БПЭ (MADALINE)

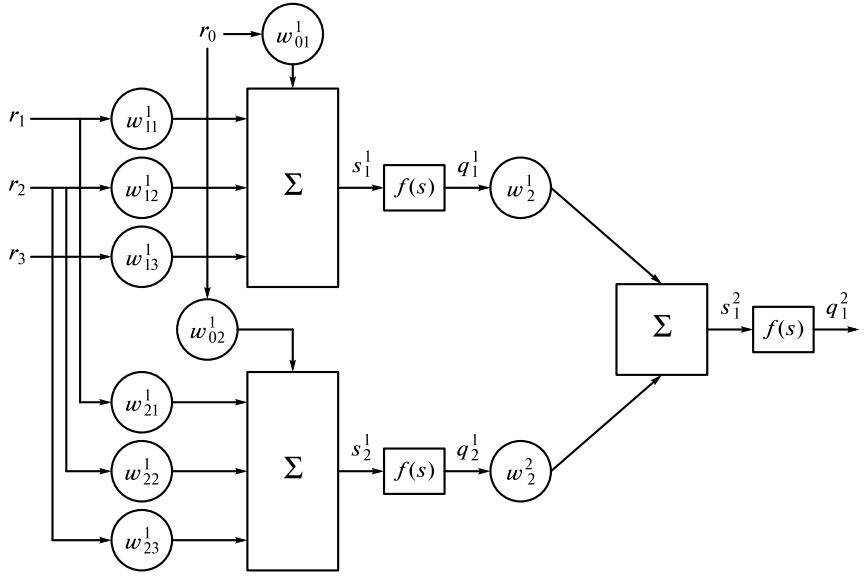


Рис. 2.9. Многослойный БПЭ

$$q = f(s) = f\left(W_i R_i + W_0 R_0\right). \tag{2.2}$$

В то же время имеется необходимость, как покажем позднее, в многослойной сети, простейшая структура которой представлена на рис. 2.9.

Очень часто во втором слое и далее может не использоваться функция суммы, а в качестве соматической функции применяют линейную функцию. В связи с этим часто слой определяется набором соответствующих весов.

Иногда используют упрощенное представление ИНС, в частности на рис. 2.10 таким образом показана однослойная структура (обычно рассматриваются сети без боковых связей), а на рис. 2.11 — сеть Хоп-

Рис. 2.10. Упрощенное представление однослойной структуры

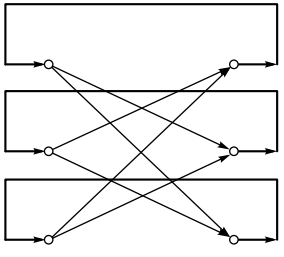


Рис. 2 11. Упрощенное представление сети Хопфилда

филда. Она является сетью с обратной связью и по свойствам близка к системам автоматического управления.

Многослойная структура рассматривается позднее.

ИНС могут быть использованы в следующих областях: распознавание образов; параллельные вычисления; оптимизация процессов; обучение; идентификация и управление.

2.3. Расчетно-логические системы, системы с генетическими алгоритмами

Структура расчетно-логических систем. Расчетно-логические системы (РЛС) в значительной степени похожи на экспертные системы. РЛС используют для числовых расчетов. Описание знаний осуществляется с помощью *функциональных семантических сетей*.

Возможности таких систем чаще всего иллюстрируют на плоской геометрии. Элемент семантической сети в общем случае отражается выражением

$$x^*y^*z = 0,$$

где x, y, z — элементы; * — некоторая операция.

Пусть a, b, c — стороны треугольника, p — его периметр, тогда $a + b + c - p = 0$.

Если A, B, C — углы треугольника, то $A + B + C - \pi = 0$.

Элементы функциональной семантической сети можно представить в виде табл. 2.1.

Таблица 2.1. **Функциональная семантическая сеть**

Обозначение	Математическая запись	Параметры отношения
$F1$	$A + B + C - \pi = 0$	A, B, C
$F2$	$\sin A/a - \sin B/b = 0$	A, B, a, b
$F3$	$\sin B/b - \sin C/c = 0$	B, C, b, c
$F4$	$\sin A/a - \sin C/c = 0$	A, C, a, c
$F5$	$a^2 - b^2 - c^2 + 2bc\cos A = 0$	A, a, b, c
$F6$	$b^2 - a^2 - c^2 + 2ac\cos B = 0$	B, a, b, c
$F7$	$c^2 - a^2 - b^2 + 2ac\cos C = 0$	C, a, b, c
$F8$	$a + b + c - p = 0$	a, b, c, p

Продолжение табл. 2.1

Обозначение	Математическая запись	Параметры отношения
$F9$	$S^2 - (p(p-a)(p-b)(p-c))^{1/2} = 0$	S, p, a, b, c
$F10$	$S - h_a a/2 = 0$	S, a
$F11$	$S - h_b c/2 = 0$	S, b
$F12$	$S - h_c c/2 = 0$	S, c
$F13$	$h_a - c\sin B = 0$	B, h_a, c
$F14$	$h_a - b\sin C = 0$	C, h_a, b
$F15$	$h_b - a\sin C = 0$	C, h_b, a
$F16$	$h_b - c\sin A = 0$	A, h_b, c
$F17$	$h_c - b\sin A = 0$	A, h_c, b
$F18$	$h_c - a\sin B = 0$	B, h_c, a
$F19$	$a - b\cos C - c\cos B$	B, C, a, b, c
$F20$	$b - a\cos C - c\cos A$	A, C, a, b, c
$F21$	$c - a\cos B - b\cos A$	A, B, a, b, c
$F22$	$\sin(A/2) - ((p-b)(p-c)/bc)^{1/2} = 0$	A, b, c, p
$F23$	$\sin(B/2) - ((p-a)(p-c)/ac)^{1/2} = 0$	B, a, c, p
$F24$	$\sin(C/2) - ((p-a)(p-b)/ab)^{1/2} = 0$	C, a, b, p
$F25$	$\cos(A/2) - (p(p-a)/bc)^{1/2} = 0$	A, a, b, c, p
$F26$	$\cos(B/2) - (p(p-b)/ac)^{1/2} = 0$	B, a, b, c, p
$F27$	$\cos(C/2) - (p(p-c)/ab)^{1/2} = 0$	C, a, b, c, p
$F28$	$\mathrm{tg}(A/2) - ((p-b)(p-c)/p(p-a))^{1/2} = 0$	A, a, b, c, p
$F29$	$\mathrm{tg}(B/2) - ((p-a)(p-c)/p(p-b))^{1/2} = 0$	B, a, b, c, p
$F30$	$\mathrm{tg}(C/2) - ((p-a)(p-c)/p(p-b))^{1/2} = 0$	C, a, b, c, p
$F31$	$\sin(A) - (p(p-a)(p-b)(p-c))^{1/2}/bc = 0$	A, a, b, c, p
$F32$	$\sin(B) - (p(p-a)(p-b)(p-c))^{1/2}/ac = 0$	B, a, b, c, p
$F33$	$\sin(C) - (p(p-a)(p-b)(p-c))^{1/2}/ab = 0$	C, a, b, c, p
$F34$	$\omega_a - (bc(a+b+c)(a+b-c))^{1/2}/(b+c) = 0$	ω_a, a, b, c

Окончание табл. 2.1

Обозначение	Математическая запись	Параметры отношения
*F*35	$\omega_b - (ac(a + b + c)(a + b - c))^{1/2}/(a + c) = 0$	ω_b, *a*, *b*, *c*
*F*36	$\omega_c - (ab(a + b + c)(a + b\text{-}c))^{1/2}/(a + b) = 0$	ω_c, *a*, *b*, *c*
*F*37	$m_a - (2b^2 + 2c^2 - b^2)^{1/2}/2 = 0$	m_a, *a*, *b*, *c*
*F*38	$m_b - (2a^2 + 2c^2 - b^2)^{1/2}/2 = 0$	m_a, *a*, *b*, *c*
*F*39	$m_c - (2a^2 + 2b^2 - c^2)^{1/2}/2 = 0$	m_c, *a*, *b*, *c*
*F*40	$D - 2R$	*D*, *R*
*F*41	$2Rh_a - bc = 0$	*R*, *b*, *c*, h_a

Здесь h_a — высота треугольника; *D*, *R* — диаметр и радиус описанной окружности.

Эта же табл. 2.1 может быть представлена в графической форме, фрагмент которой показан на рис. 2.12.

Имеет место следующая постановка задачи: «Даны некоторые функциональные отношения и исходные данные. Требуется найти функциональное решение (достичь цели)».

Например, пусть даны сторона и два угла треугольника, необходимо найти с помощью функциональных зависимостей площадь треугольника.

В общем случае задача ставится так: «Имеется некоторая цель. Необходимо определить, можно ли ее достичь и как при имеющихся функциональных отношениях». Задача похожа на задачу выводимости для ЭС. Однако здесь нет твердого правила выводимости и уверенного алгоритма получения результата (поиска). В общем случае может оказаться, что цель недостижима. Возможны циклы при поиске результатов. Задача поиска фактически использует переборный алгоритм.

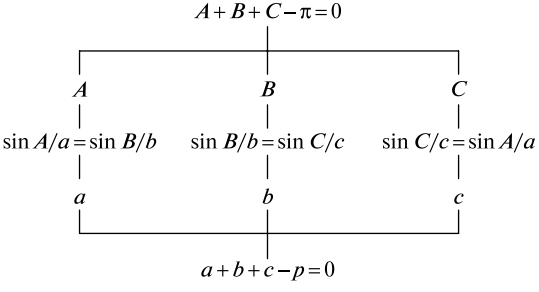


Рис. 2.12. Фрагмент функциональной семантической сети

Технология функционирования расчетно-логических систем. Известны два алгоритма поиска: метод обратной волны и метод прямой волны. Они являются в определенном смысле аналогами метода обратного и прямого поиска для экспертных систем.

Метод обратной волны. Первоначально выделяют четыре множества, или списка (некоторые из них могут быть пустыми):

$S1$ — список параметров, которые должны быть определены;

$S2$ — список параметров, для которых есть соответствующая функция (имеется разрешение);

$S3$ — список разрешений, включаемых в план поисков (в случае неоднозначного решения задачи);

$S4$ — список критериев для оценки вариантов неоднозначных решений.

Алгоритм оперирует этими списками.

Этап 1. Исходная цель размещается в списке $S1$. Затем параметры этого списка размещаются в список $S2$ и отыскивают необходимую функцию F_i. При этом возможны несколько решений и в этом случае следует выбрать функцию с наименьшим числом аргументов. Если все аргументы известны, то план решения найден и следует перейти к вычислению. Если какой-либо аргумент в функции неизвестен, его снова записывают в список $S1$ и повторяют этап. Переход к этапу 2.

Этап 2. Упорядочить список $S1$ от наибольшего к наименьшему числу аргументов.

Этап 3. Выбрать из списка $S1$ параметр с наименьшим числом аргументов и поместить в список $S2$. Если в списке $S1$ последний параметр и список пусты, поиск плана решения закончен и следует перейти к поиску нового плана решения или к вычислениям. Иначе — поместить неизвестные аргументы в список $S1$ и перейти к этапу 2.

Рассмотрим этот алгоритм на примере.

Пример 2.1. Пусть дана одна сторона с треугольника и два угла A и B. Необходимо найти площадь S. Процесс решения показан в табл. 2.2.

Таблица 2.2. **Процесс решения**

Обозначение	Математическая запись	Список определяемых параметров	Список разрешенных параметров	Список разрешений	Список критериев	Шаги
$F1$	A^*, B^*, C	$S1$	$S2$	$S3$	$S4$	
$F2$	A^*, B^*, a, b	S				Исходное состояние

Окончание табл. 2.2

Обозначение	Математическая запись	Список определяемых параметров	Список разрешенных параметров	Список разрешений	Список критериев	Шаги
$F3$	B^*, C, b, c^*	h_c	S	$F12$	$L12$	Шаг 1
$F4$	A^*, C, a, c^*					Шаг 2
$F5$	A^*, a, b, c^*	b	h_c	$F17$	$L17$	Шаг 3
$F6$	B^*, a, b, c^*					Шаг 2
$F7$	C, a, b, c^*	C	b	$F3$	$L3$	Шаг 3
$F8$	a, b, c^*, p					Шаг 2
$F9$	S, a, b, c^*		C	$F1$	$L1$	Шаг 3
$F10$	S, a					
$F11$	S, b					
$F12$	S, c^*					
$F13$	B^*, h_a, c^*					
$F14$	C, h_a, b					
$F15$	C, h_b, a					
$F16$	A^*, h_b, c^*					
$F17$	A^*, h_c, b					
$F18$	B^*, h_c, a					

Поскольку список $S1$ пуст, то план решения найден и схема вычислений будет выглядеть так: $< A^*, B^*, c^* > \rightarrow F(1, C) \rightarrow C \rightarrow \rightarrow F(3, b) \rightarrow b \rightarrow F(17, h_c) \rightarrow h_c \rightarrow F(12, S) \rightarrow S$, где $F(12, S)$ есть математическое разрешение отношения $F(12)$ относительно S.

Метод прямой волны. Рассмотрим его на примере 2.1. Обозначим через *np* число параметров в математическом отношении, *kp* — число параметров, найденных на данном шаге применением соответствующего оператора *F*, *unkp* — число неизвестных параметров.

Составим таблицу решения (табл. 2.3). После каждого шага, на котором находится какой-либо аргумент, пересчитываются значения *unkp* и поиск повторяется до нахождения целевого параметра. Таким образом, для функциональных семантических сетей набор отношений может быть неполным. Могут возникнуть две ситуации: тупик (решения нет) и наличие циклов при нахождении решений.

Таблица 2.3. **Порядок решения**

Обозначение	Математическая запись	*пр*	Шаги алгоритма							
			1		2		3		4	
			unkp	*kp*	*unkp*	*kp*	*unkp*	*kp*	*unkp*	*kp*
*F*1	*A**, *B**, *C*	3	1	1(C)	0	0	0	0	0	0
*F*2	*A**, *B**, *a*, *b*	4	2	0	2	0	0	0	0	0
*F*3	*B**, *C*, *b*, *c**	4	2	0	1	1(*b*)	0	0	0	0
*F*4	*A**, *C*, *a*, *c**	4	2	0	1	1(*a*)	0	0	0	0
*F*5	*A**, *a*, *b*, *c**	4	2	0	2	0	0	0	0	0
*F*6	*B**, *a*, *b*, *c**	4	2	0	2	0	0	0	0	0
*F*7	*C*, *a*, *b*, *c**	4	2	0	2	0	0	0	0	0
*F*8	*a*, *b*, *c**, *p*	4	3	0	3	0	1	1(*p*)	0	0
*F*9	*S*, *p*, *a*, *b*, *c**	5	4	0	4	0	2	0	1	1(*S*)

В обоих случаях должны быть предусмотрены специальные программные приемы, позволяющие выходить прежде всего из циклов. Нетрудно заметить, что сфера применения РЛС ограничена.

Структура систем с генетическими алгоритмами. Эти системы [15, 38, 47] появились в процессе поиска новых методов математических расчетов, поскольку возможности как детерминированных, так и статистических методов оказались недостаточными для описания вновь появляющихся классов систем.

В методах эффективного поиска сформировалось понятие «эволюционные вычисления» (рис. 2.13), гогда оказалось полезным сочетать детерминированную составляющую алгоритмов со случайной составляющей. В методах эффективного поиска выделяются три группы:

- методы, основанные на математических вычислениях — это все виды программирования. Серьезным ограничением здесь являются требования дифференцируемости функции. Если эти требования не соблюдены, то используются специальные методы программирования. Сложным в общем случае может оказаться многоэкстремальный характер кривой выбора решения;
- перечислительные, или переборные, методы (целочисленное, динамическое программирование) характеризуются высокой размерностью задач, большим объемом вычислений;
- методы, использующие элемент случайности (случайный поиск) не дают гарантии оптимальности решений и получения более хорошего результата, чем в предыдущих методах.

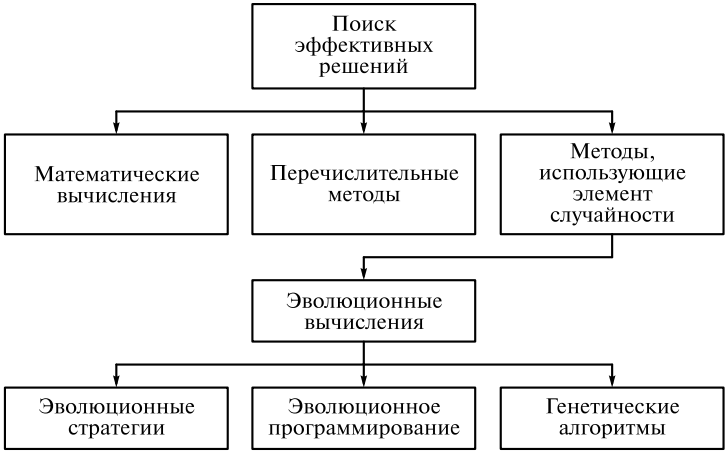


Рис. 2.13. Методы поиска оптимальных решений

Методы эволюционных вычислений, в свою очередь, также можно классифицировать:

- эволюционные стратегии по своей сути похожи на метод генетических алгоритмов, однако целевая функция может меняться от поколения к поколению. Основной признак изменения — мутация;
- эволюционное программирование базируется на теории конечных автоматов, которые должны реагировать на изменения внешней среды;
- в генетических алгоритмах используются приемы из генетики. Процесс заменяется некоторым кодом. Используется специфическая терминология (код и его структура называются *генотипом*, конкретная интерпретация кода, т.е. переход от задачи к коду, — *фенотипом*; набор конкретных кодов — *популяцией*; индивидуальный код — *хромосомой* (особью, индивидом); каждый шаг в алгоритме — *поколением*; особи предыдущего поколения — *родителями*, последующего поколения — *потомками*).

Общая цель названных методов — адаптация к изменяющимся условиям. В то же время общая цель генетических алгоритмов — описание процесса развития популяций.

Технология формирования систем с генетическими алгоритмами. В самом общем виде последовательность работы с генетическим алгоритмом (ГА) показана на рис. 2.14.

В качестве критерия останова могут использоваться достижение критерием оптимума, достижение заданного количества поколений, малое изменений целевой функции.

ГА — вариант решения в виде битовой строки фиксированной длины, манипулирование с которой производится в отсутствии связей с ее смысловой интерпретацией.

Возможности ГА можно проиллюстрировать на примере.

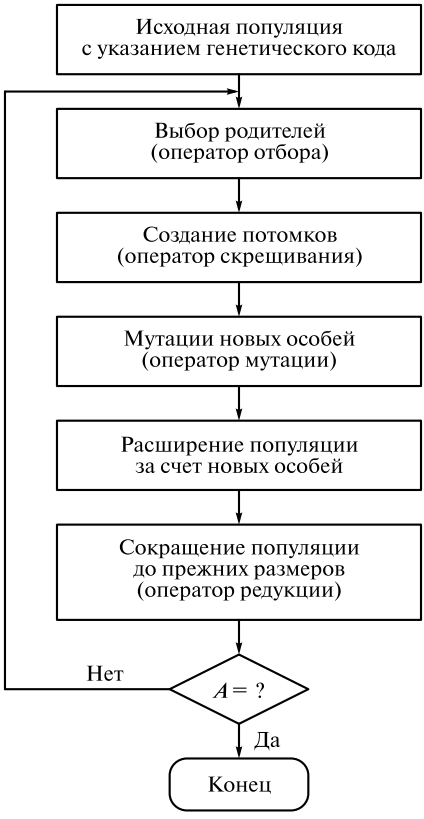


Рис. 2.14. Общая схема генетического алгоритма:
A — критерий остановки

Пример 2.2. Имеется пять городов, которые коммивояжер должен посетить по одному разу. Начало и конец обхода совпадают. Задано расстояние между городами в виде матрицы. Требуется найти такой обход, чтобы суммарное расстояние было минимальным.

В данном примере примем следующую матрицу расстояний:

$$\begin{pmatrix} 0 & 4 & 6 & 2 & 9 \\ 4 & 0 & 3 & 2 & 9 \\ 6 & 3 & 0 & 5 & 9 \\ 2 & 2 & 5 & 0 & 8 \\ 9 & 9 & 9 & 8 & 0 \end{pmatrix}.$$

Если решить эту задачу точным методом, то минимальное расстояние равно 25, а порядок обхода — 514235. Перейдем к решению с помощью метода ГА.

На этапе скрещивания код элементов пар разделяется на части, а затем происходит обмен между элементами пары. Правила обмена могут быть самыми разными, в том числе при формировании различных поколений.

Например, правила обмена могут быть следующими. Пусть имеется пара родителей, обладающих кодами

1	2	3	4	5
2	4	5	2	1

1	4	5	2	3
5	2	3	4	1

Разделим коды на три части. Поменяем местами среднюю часть кода. Затем вставляем числа, начиная со второго элемента среднего фрагмента, из родительской пары, пропуская уже имеющиеся элементы в новой перестановке. Воспользовавшись данным правилом для табл. 2.4, построим табл. 2.5. Учтем, что для шестой строки осуществляется мутация (в виде перестановки) второго и третьего элементов. Возвращаем популяцию в прежние размеры, для чего удаляем особи 3, 4, 5, 7. При этом получается табл. 2.6 для второго поколения. При использовании генетических алгоритмов допустимо повторение отдельных особей. Появилось новое поколение, но уверенности, что решение оптимально, нет.

Следующее поколение: пусть родительские пары 1 и 4, 2 и 3. При скрещивании в первой паре отделим первые три элемента кода, а во второй — первые два элемента кода. Далее используем прежнее правило. Пусть мутация имеет место для четвертого и пятого элементов строки 7. Получим табл. 2.7.

Удаляем особи 4, 5, 6, 7. Получим среднее 28,75 и min 28.

Таким образом, два последних поколения не привели к оптимальному результату. При большом объеме кодов может возникнуть вопрос об остановке вычислений.

Для СГА характерно большое количество разновидностей (рис. 2.15), поскольку различные этапы ГА (см. рис. 2.14) имеют разновидности.

В общем подходе имеются два класса: популяция рассматривается целиком; популяция делится на блоки, и скрещивание происходит внутри блока. В некоторых поколениях возможно межблочное скрещивание.

Популяция может быть фиксированная (детерминированная) и случайная (недетерминированная); в последнем случае величина ее может быть вероятностная.

Отбор может происходить целиком и по группам.

В скрещивании (кроссинговере) выделяют одноточечный и многоточечный варианты. В последнее время используется понятие «элитизм», когда часть лучших особей переходят в следующее поколение.

Таблица 2.4. **1-е поколение**

Номер строки	Код	Целевая функция	Вероятность участия в скрещивании
1	12345	29	32/122
2	21435	29	32/122
3	54312	32	29/122
4	43125	32	29/122
Сумма 122; среднее 30,5			

Таблица 2.5. **Результат 1**

Номер строки	Родители	Потомки	Целевая функция
5	12345	54312	32
6	54321	(12354) 13254	28
7	21435	43125	32
8	43125	21435	29

Таблица 2.6. **2-е поколение**

Номер строки	Код	Целевая функция	Вероятность участия в скрещивании
1	12345	29	28/115
2	21435	29	28/115
3	13254	28	29/115
4	21435	29	28/115
Сумма 115; среднее 28,75			

Таблица 2.7. **Результат 2**

Номер строки	Родители	Потомки	Целевая функция
5	1 2 3 | 4 5	2 1 4 | 3 5	29
6	2 1 4 | 3 5	1 2 3 | 4 5	29
7	2 1 | 4 3 5	1 3 | 4 5 2	29
8	1 3 | 2 5 4	2 1 | 3 5 4	29

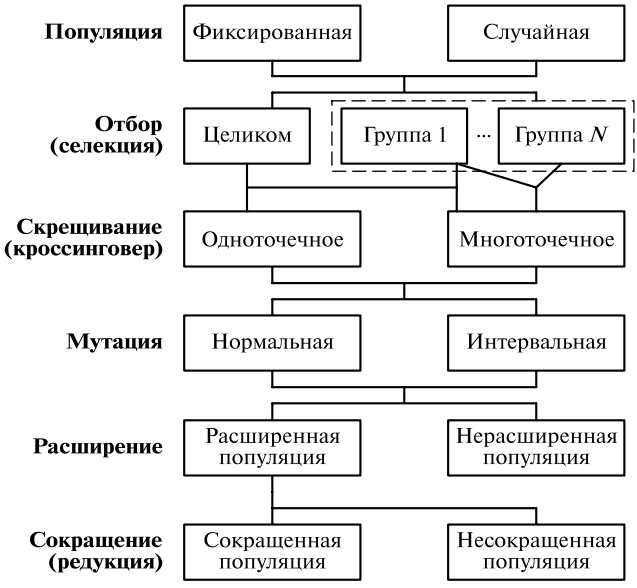


Рис. 2.15. Набор разновидностей систем с генетическими алгоритмами

Мутации могут определяться нормальным законом — нормальная мутация, некоторым интервалом (границами) — интервальная мутация. В последнее время вместо мутации используют понятие «рекомбинация» — обмен частью кода у потомков.

Детерминированность в ГА проявляется в процессах отбора, расширения и наследования, тогда как случайность — в процессе мутации. В системе с генетическим алгоритмами еще не сложилась однозначная терминология. В работах [15, 38, 47] используемые методы называют по-другому. Скрещивание и мутацию называют рекомбинацией, отбор — селекцией, кроссинговер — одно- и двухточечным, упорядоченным, частично-соответствующим, циклическим, универсальным.

В упорядоченном методе точка разреза выбирается случайно, а затем идет обмен. Частично-соответствующий метод похож на упорядоченный, но замена ведется лишь для повторяющихся генов. В циклическом методе замены проводятся циклами (строительными блоками). В универсальном операторе используется маска.

В выборе пар используют такие методы: случайный, выбор лучших особей, близкое и дальнее родство с применением кода Грея или расстояния Хэмминга.

Отбор (селекция, репродукция) осуществляется на основе рулетки, заданной шкалы (критерия), турнирно (выбирается группа, в которой проводится турнир).

После скрещивания образуется расширенная популяция (этап расширения). Далее популяция приводится к прежней размерности (сокращение/редукция) по принятому критерию эффективности членов популяции.

Из приведенного перечисления методов видно, что решения на каждом этапе алгоритма, приведенного на рис. 2.14, носят неформальный и неоднозначный характер. Чтобы частично упорядочить методы, сократить время поиска и как можно больше приблизиться к экстремуму, был предложен ряд правил.

В их основе по-прежнему лежат неформальные процедуры. В то же время такое упорядочение можно считать попыткой формирования начал теории генетических алгоритмов.

Для этого введено понятие «шаблон». Шаблон — подмножество хромосом, содержащих знаки *** (звездочки). На основании шаблона можно построить набор хромосом, если заменить * на 0 или 1. Шаблоны называют *изоморфными*, если из них можно получить одинаковые хромосомы. Для шаблона ** изоморфны четыре хромосомы. Для шаблона длиной L можно получить 3^L хромосом и 2^L неизоморфных хромосом. Короткий шаблон называют *строительным блоком*. Для шаблона ***1 строительный блок — элемент 1.

Шаблон имеет два параметра: $o(H)$ порядок, $\delta(H)$ длина. Для H = (0***) величина $o(H) = 1$. Длина шаблона — расстояние между первой и последней позициями нулей и единиц. Так, $\delta(011*1***) = 5 - 1 = 4$.

Хромосома i выбирается с вероятностью

$$Pr(i) = f_i(x)\sum_{j=1}^{N} f_j(x),$$

где $f_i(x)$ — значение целевой функции i-й хромосомы.

Пусть $m(H, t)$ — число хромосом в популяции P^t. Тогда в следующей генерации популяции P^{t+1} будет $m(H, t + 1)$ хромосом:

$$m(H,t+1) = m(H,t)\,N_p\,f(H)\Big/\sum_{j=1}^{N} f_j(x) = m(H,t)f(H)/f_{\text{ср}}(x),$$

где $f(H)$ — среднее значение целевой функции шаблона; $f_{\text{ср}}(x)$ — среднее значение целевой функции популяции; N_p — размер непересекающихся популяций.

Если целевая функция шаблона выше среднего значения целевой функции популяции на величину $cf_{\text{ср}}(x)$, c = 1, 2, ..., то

$$m(H, t + 1) = m(H, t)(f_{\text{ср}}(x) + cf_{\text{ср}}(x))/f_{\text{ср}}(x) = m(H, t)(1 + c).$$

Тогда

$$m(H, t) = m(H, 0)(1 + c)^t.$$

Число хороших хромосом растет. Тогда шаблон с целевой функцией выше среднего копируется в следующую генерацию, а шаблон с целевой функцией ниже среднего устраняется.

Вероятность выживания шаблона длиной H для оператора кроссинговера

$$Pr(s) = 1 - \delta(H)/(L - 1).$$

Для H = (**1***0*) величина $\delta(H) = 7 - 3 = 4$, $L - 1 = 7$, $Pr(s) = 1 - 4/7 = 3/7$, а вероятность разрушения — 4/7.

Возможно вычислить вероятности и для других операторов [15].

Нетрудно заметить, что и после введения начала теории генетических алгоритмов (ГА) в их построении остается еще очень много неформальных моментов.

Достоинствами ГА являются их простота и отсутствие связей с физической сущностью описываемого процесса после перехода к соответствующим кодам.

Метод генетических алгоритмов можно использовать в ИНС, если в каждом слое ИНС количество слоев и элементов неизвестно. Эта процедура достаточно трудоемка в описании, поэтому рассмотрим лишь идею такого расчета.

Таким образом, ставится задача определения числа слоев и количества БПЭ в каждом слое для ИНС. На входе имеются входные и выходные БПЭ, набор обучающих примеров или значений. Необходимо определить количество промежуточных слоев, количество БПЭ в каждом слое, значения весов во всех слоях, вид функции активизации в каждом слое.

Последнюю задачу часто упрощают выбором сигмоидальной функции во всех слоях. В качестве параметра выбирают наклон функции.

Тогда общая задача решается в два этапа:

1) определение числа слоев и количества БПЭ в каждом слое;

2) определение весов и наклонов функций.

Рассмотрим второй этап. Пусть имеется ИНС, показанная на рис. 2.16. (здесь P_i — наклон сигмоидальной функции):

$$f(x) = \frac{1}{1 + \mathbf{e}^{p_i^x}} P_i > 0;$$

$$-1 \le w_{ij} \le 1.$$

Исходная популяция выбирается на основе равномерного распределения и должна содержать не менее 100 кодов (особей). Элитные особи сразу попадают в следующее поколение.

Пусть имеются $P = 0{,}95$ (вероятность скрещивания); Э = $(1 - P)N$ (число элитных особей); N (общее число особей). Мутация происходит с вероятностью $P_м = 0{,}01$. В данном случае полагаем, что в каждом поколении меняются одно значение P_i и один из весов.

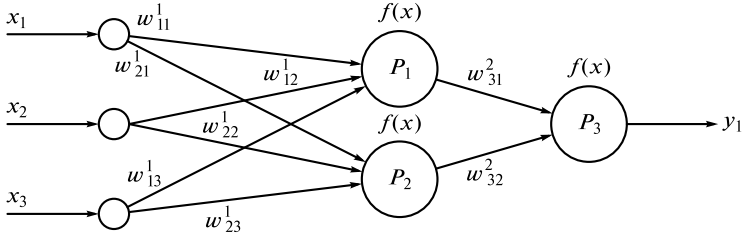


Рис. 2.16. Выбор параметров ИНС с помощью метода генетических алгоритмов

Таблица для начала работы генетического алгоритма имеет следующий вид:

w^1_{11}	w^1_{12}	w^1_{31}	P_1	w^1_{21}	w^1_{22}	w^1_{11}	P_2	w^1_{31}	w^1_{32}	P_3

В силу объемности расчетов дальнейшие шаги не рассматриваются.

Аналогичный подход может быть использован и на первом этапе.

Генетические алгоритмы хорошо работают с числом кодов не менее 100. Поскольку алгоритм переборный, даже при использовании специальных программных пакетов решение занимает много времени.

Иногда выходят из положения распараллеливанием. Множество кодов разбивается на группы, отыскивается набор лучших элементов в каждой группе и с ними проводится новое вычисление. Подобный прием позволяет больше приблизиться к получению оптимального значения.

2.4. Мультиагентные системы

Теория мультиагентных систем только начала складываться. Показанные на рис. 2.17 предметные области развивались практически автономно.

Область систем искусственного интеллекта (СИИ) с позиции принятия решений сильно ограничена. При этом рассматриваются дискретные процессы. В то же время появились адаптивные непрерывные системы.

Адаптация возможна к изменению параметров объекта управления (самонастраивающаяся система), к изменению структуры ОУ (самонастраивающаяся система).

Возникла задача интеграции предметных областей, характеризующихся кривой *1* на рис. 2.17. Напрямую решить такую задачу оказа-

лось сложно. Поэтому пока изучают интеграцию предметных областей, ограниченных кривой *2*.

Считается, что эта интеграция является началом решения первого понимания термина «искусственный интеллект». Одним из таких методов интеграции является метод мультиагентных (многоагентных) систем (МАС).

В мультиагентных системах используют модальную логику — логику достоверности. Достоверность может иметь, как уже отмечалось, три разновидности:

1) аподиктическая (наибольшая достоверность) — целое больше части;

2) ассерторическая (утвердительная логика) — «яблоко разделено на части»;

3) проблематичная (фактор уверенности, нечетные переменные).

Модальная логика не может «работать» без пропозициональной логики (логики предложений, в частности, логики правил).

Термин «агент» происходит от латинского глагола *agere*, что означает «действовать», «двигать», «править», «управлять». В энциклопедическом словаре Ф. А. Брокгауза и И. А. Ефрона [12] читаем: «агент — деятель, лицо, действующее по поручению или полномочию другого».

Для успешной работы агент должен обладать достаточными интеллектуальными способностями, чтобы в сфере своих задач замещать действия человека-владельца, должен иметь возможность взаимодействовать с владельцем или пользователем для получения соответствующих заданий и передачи результатов, должен ориентироваться в среде своего существования и принимать необходимые решения.

Агент — лицо, действующее по поручению (вне компьютерного понимания) другого лица.

Рис. 2.17. Соотношение предметных областей

С компьютерной точки зрения *агент* — вычислительный процесс, который реализует автономную коммуникационную функциональность приложений. С точки зрения систем искусственного интеллекта целесообразно иметь интеллектуального агента.

Поскольку общепринятого определения *интеллектуального агента* (ИА) до настоящего времени не выработано, то следует говорить о классе агентных объектов (сущностей), включающем в себя множество видов [62, 70].

Отталкиваясь от определений данных в [37], будем считать, что ИА — это программный, или аппаратный, объект (сущность), автономно функционирующий для достижения целей, поставленных перед ним владельцем или пользователем, и обладающий определенными интеллектуальными способностями. Уровень этих способностей, необходимых для достижения поставленных перед ИА целей, можно определить, пользуясь классификацией Д. А. Поспелова. Если классифицировать среды, в которых должны действовать агенты, то получается следующая схема (рис. 2.18).

Для замкнутых сред может быть построено конечное исчерпывающее описание. Функционирующие в таких средах агенты могут обладать полным знанием о среде и ее свойствах или получить эту информацию в процессе своего взаимодействия со средой. Трансформируемые среды могут изменять свои характеристики и реакции на действия агентов и зависят от тех действий, которые агенты совершают в среде.

Вид математического аппарата, позволяющего описать поведение агента в соответствующей среде, и является мерой его интеллектуальной сложности (или разумности). Представим эту классификацию в виде табл. 2.8.

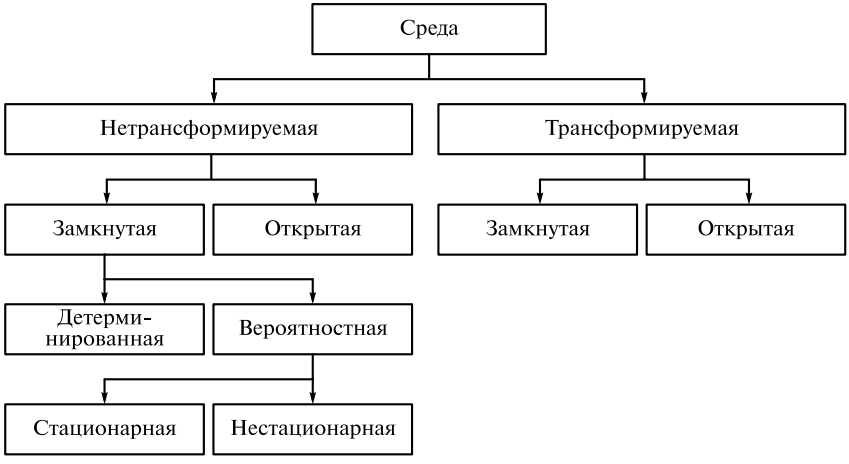


Рис. 2.18. Классификация сред

Таблица 2.8. **Мера интеллектуальной разумности**

Тип среды функционирования	Метод математического описания	Вид ИА
Замкнутая детерминированная	Автоматные грамматики, конечные автоматы	Автоматные агенты
Замкнутая вероятностная	Вероятностные автоматы	Вероятностные автоматные агенты
Нетрансформируемая открытая	Контекстно-свободные грамматики (КСГ), сценарии, магазинные автоматы	КСГ-агенты
Трансформируемая замкнутая	Контекстно-зависимые грамматики (линейные автоматы)	КЗГ-агенты
Трансформируемая открытая	Семиотические системы	Семиотические агенты

Две базовые характеристики — автономность и целенаправленность — позволяют отличать ИА от других программных и аппаратных объектов (модулей, подпрограмм, процедур). Наличие целесообразности поведения требует, чтобы ИА обладал свойством реактивности. Такой уровень интеллекта соответствует рефлекторному поведению животного. Если же ИА обладает знаниями о среде, собственных целях и способах их достижения, то такой агент может быть назван разумным. Следовательно, может быть проведена граница между интеллектуальными и неинтеллектуальными агентами.

Значительные усилия по стандартизации агентных систем и технологий предпринимает организация Foundation for Intelligent Physical Agents (FIPA), которая является международной организацией, выполняющей работу по разработке открытых спецификаций, поддерживающих интероперабельность агентов и агентных приложений. В январе 2000 г. членами FIPA были 56 фирм и университетов из 17 стран. Согласно спецификации абстрактной архитектуры агент определяется как вычислительный процесс, который реализует автономную коммуникационную функциональность приложений. Это определение неконструктивно относительно интеллектуальных способностей агента и поэтому целесообразно использовать классификацию в табл. 2.9, данную Nwana [70].

Таблица 2.9. **Классификация агентных систем**

Характеристика	Тип агента			
	Простой	Смышленый (smart)	Интеллектуальный (intelligent)	Действительно (truly) интеллектуальный
Автономное выполнение	+		+	+
Взаимодействие с другими агентами и/или пользователями	+	+	+	+
Слежение за окружением	+	+	+	+
Способность использовать абстракции		+	+	+
Способность использовать предметные знания		+	+	
Возможность адаптивного поведения для достижения целей			+	+
Обучение из окружения			+	+
Толерантность к ошибкам и/или неверным входным сигналам			+	
Real-time-исполнение			+	
Естественно-языковое взаимодействие			+	

Сформировался довольно большой список свойств, которыми должны обладать ИА:

- автономность (autonomy, autonomous functioning), т. е. способность к самостоятельному формированию целей и функционированию с самоконтролем своих действий и внутреннего состояния;
- общественное поведение (social ability, social behavior) — способность согласовывать свое поведение с поведением других агентов

в условиях определенной среды и правил поведения путем обмена сообщениями на языке коммуникации;

- реактивность (reactivity) — способность воспринимать состояние внешней среды (среды функционирования и множества других агентов) и своевременно приспосабливаться (адаптироваться) к происходящим изменениям;
- активность (pro-activity) — способность проявлять инициативу, т.е. самостоятельно генерировать цели и действовать рационально для их достижения, а не только пассивно реагировать на внешние события;
- базовые знания (basic knowledge) — постоянная часть знаний агента о себе, о среде, а также постоянные знания о других агентах, которые не изменяются в рамках жизненного цикла агента;
- убеждения (beliefs) — переменная часть знаний агента о среде и других агентах, которая может изменяться во времени, но агент может об этом не знать и продолжать использовать их для своих целей;
- желания (desires) — состояния и/или ситуации, достижение которых является желательным и важным для агента, однако которые могут быть противоречивыми и не всегда достигнутыми;
- цели (goals) — совокупность состояний, на достижение которых направлено текущее поведение агента;
- намерения (intentions) — это то, что агент обязан сделать в силу своих обязательств по отношению к другим агентам, или то, что следует из его желаний (т.е. непротиворечивое подмножество желаний, выбранное по тем или иным причинам и совместимое с принятыми на себя обязательствами);
- обязательства (commitments) — задачи, которые берет на себя агент по просьбе и/или поручению других агентов.

К этому набору свойств могут добавляться такие:

- благожелательность (benevolence) — готовность агентов помогать друг другу и решать именно те задачи, которые им поручат владелец или пользователь;
- правдивость (veracity) — свойство агента не оперировать заведомо ложной информацией;
- рациональность (rationality) — способность агента действовать так, чтобы достигать своих целей, а не избегать их достижения, по крайней мере, в рамках своих знаний и убеждений.

Введенные таким образом необходимые свойства интеллектуальных агентов требуют некоторого обсуждения, так как на искусственный технический объект переносятся социальные свойства, присущие обычно поведению человека.

Общественное поведение агентов может принимать разные формы, которые могут быть классифицированы по уровням взаимодействия.

Уровень 0 — *связность*. Устанавливается извне владельцем или пользователем и не воспринимается самими агентами.

Уровень 1 — *координация*. Агенты способны создать ситуацию, позволяющую другим агентам оказаться в нужном месте в нужное время, чтобы в результате их деятельность осуществлялась эффективно.

Уровень 2 — *кооперация*. Агенты допускают, чтобы их поведение частично определялось поведением других агентов, когда они совместно пытаются достичь некоторой общей цели. Такой процесс для своей реализации должен осознаваться всеми участвующими в нем агентами.

Уровень 3 — *сотрудничество*. Предполагается реальная совместная работа агентов, в процессе выполнения которой может выиграть каждый.

Уровень 4 — *образование союза*. Предусматривается продолжительная во времени совместная деятельность, в ходе которой агенты создают и поддерживают условия существования союза.

Базовые знания являются необходимым традиционным компонентом для всех интеллектуальных систем, убеждения же должны быть определенным образом интерпретированы в структуре МАС.

Интеллектуальная система (агент) может воспринимать правила формирования выводов, базовые шкалы и веса критериев, функции или отношения предпочтения как истинные.

Убеждения можно разделить на три класса:

1) внутренние убеждения агента — это алгоритмы, сценарии, оценки, заложенные в него при разработке или внесенные в процессе эксплуатации владельцем или пользователем;

2) индуктивные убеждения, возникающие в результате анализа состояний среды, формируемые продукционными правилами следующего вида: если наблюдается факт X, то убеждение Z;

3) коммуникативные убеждения, возникающие в результате связи с другими агентами, формируемые продукционными правилами следующего вида: если A сообщает о факте x и A — заслуживающий доверия источник, то убеждение Z.

Приписывание ИА желаний и намерений, по мнению авторов, не является бесспорным, так как в реальных технических системах они могут вступить в противоречие с целями и обязательствами, предусмотренными для них человеком-разработчиком, и привести к потере управляемости и непредсказуемому поведению систем.

В последние годы термин МАС применяют к широкому спектру программных систем, состоящих из различных автономных и полуавтономных компонентов [16, 32, 34, 42, 66, 72, 74].

Дж. Люгер определяет МАС как вычислительную программу, решатели которой расположены в некоторой среде, каждый из них способен к гибким, автономным и социально организованным действиям и направлениям на предопределенной реалии или цели.

Анализ работ наиболее авторитетных в этой области исследователей показывает, что МАС должны обладать четырьмя важнейшими свойствами: ситуативностью, автономностью, гибкостью и социальностью.

Ситуативность ИА понимается как способность воспринимать окружающую его среду (окружение) и действовать в этой среде, при возможности изменяя ее в своих целях. Примером таких интеллектуальных агентов могут служить мобильные роботы, участвующие в соревнованиях ROBOCUP, которые должны взаимодействовать с мячом, партнерами по команде и противниками, не зная заранее размещения и намерений других игроков.

Автономность ИА означает его способность взаимодействовать со средой без прямого участия других агентов, для чего он должен уметь контролировать свое внутреннее состояние и выполняемые действия.

Гибкость агента демонстрируется отзывчивостью или предусмотрительностью (в зависимости от ситуации). Отзывчивый агент получает стимулы от своего окружения и вовремя отвечает на них соответствующим образом. Предусмотрительный агент не просто реагирует на ситуацию в окружающей среде, но и адаптируется, целенаправленно действует и выбирает альтернативы в различных ситуациях.

Агент обладает свойством *социальности*, если он может соответствующим образом взаимодействовать с другими программными или человеческими агентами. ИА — лишь составляющая сложного процесса решения проблем в соответствующей среде.

Представляется, что не совсем правильно трактовать многоагентную систему как чисто программную систему, поскольку агенты могут быть сложными программно-аппаратными компонентами или гибридами, например с использованием нейронных сетей, где традиционное понятие «программы» оказывается в определенной степени размытым.

В. Б. Тарасов [52] дает формализованное определение МАС, не детализируя содержание входящих в формулу составляющих

$$MAC = (A, E, R, ORG, ACT, COM, EV),$$

где A — множество агентов; E — множество сред, находящихся в определенных отношениях R и взаимодействующих друг с другом, формирующих некоторую организацию ORG, обладающих набором индивидуальных и совместимых действий ACT (стратегия поведения и поступков), включая возможные коммуникативные действия COM и возможность эволюции EV.

МАС может рассматриваться как сильно связанная сеть решателей, совместно работающих над проблемами, которые могут выходить за рамки возможностей индивидуальных агентов.

Исходя из изложенного можно дать такое определение: МАС — совокупность взаимосвязанных агентов, как программных так и аппаратных, способных взаимодействовать друг с другом и с окружающей средой, обладающих определенными интеллектуальными спо-

Рис. 2.19. Классификация МАС

собностями и возможностью индивидуальных и совместных действий.

Интеллектуальными МАС будем называть системы, обладающие всеми признаками ИНС в соответствии с определением К. А. Пупкова.

Классифицировать МАС можно по целому ряду признаков (рис. 2.19).

Важным признаком является способ решения проблем. Системы распределенного решения проблем содержат агентов, специально разработанных для решения определенного круга задач или достижения известных целей. В этом случае все агенты определяются априорно (на стадии проектирования) и должны взаимодействовать согласованно и непротиворечиво (благожелательно).

Открытые МАС могут содержать переменные множеств агентов, входящих в систему и выходящих из нее, причем возможно своекорыстное (столкновение интересов) поведение агентов.

Понятие «агентно-ориентированная система» (АОС) представляется более широким, чем понятие МАС. M. Wooldridge характеризует АОС как систему, в которой ключевой используемой абстракцией является агент.

Агентно-ориентированные системы трактуем как гибридные системы, содержащие, наряду с МАС, и другие системы (ЭС, обучающие и тестирующие системы, распределенные объектные приложения).

АОС в настоящее время являются глобально распределенными и взаимосвязанными совокупностями программных и аппаратных систем, содержащими сотни и тысячи компонентов, что и определяет возрастающую сложность их разработки, внедрения и эксплуатации.

2.5. Системы на естественном языке

Здесь следует выделить процедуры поиска, автоматического аннотирования и реферирования [19, 41, 46, 59].

В свою очередь, в поисковой системе следует выделить ручной режим, характерный для библиотек, и автоматизированный режим — для электронных документов [64, 69]. Правда, в современных библиотеках все шире используется автоматизированный поиск.

Поисковые системы. Попытка организации простого общения пользователя с компьютером на естественном языке при взаимодействии с программным продуктом была предпринята в пакете GURU 2.0.

Предварительно в словарь раздела «Естественный язык» было введено 500 английских слов. В словаре отсутствуют такие неоднозначности, как омонимы и идиоматические выражения, а для синонимов предполагается дополнительный словарь.

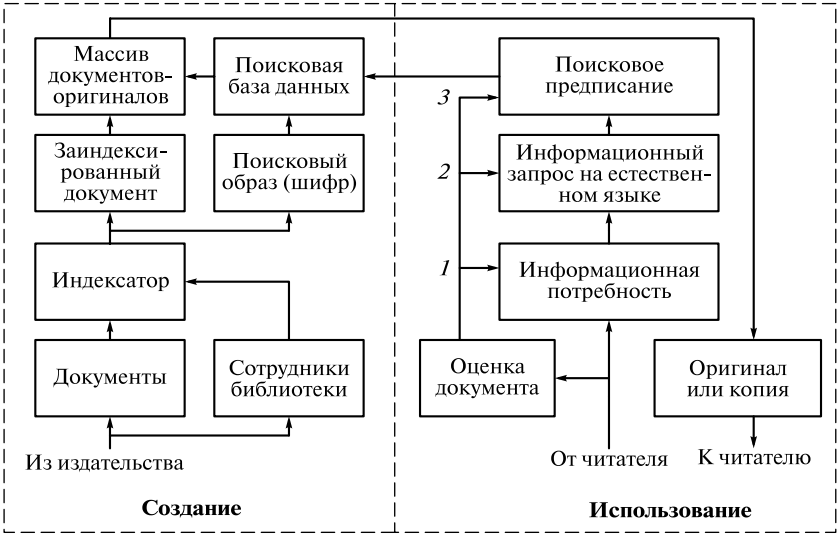


Рис. 2.20. Схема работы библиотеки:

1 — пертинентность; *2* — релевантность; *3* — смысловое соответствие

Возможно пополнение словаря новыми словами. Этот процесс облегчается, когда в словаре уже есть однокоренные слова. При вводе нового слова предусмотрено исправление ошибок (пропущенные и лишние буквы, переставленные буквы). Затем проводится лексический и синтаксический анализ слова и при отсутствии ошибок анализа слово включается в словарь.

В описании названного раздела GURU не указан метод представления знаний. Косвенная информация позволяет предположить, что используется дескриптивная логика. Иллюстрация работы GURU на естественном языке приведена в приложении 1.

Чтобы понять автоматизированный поиск, полезно рассмотреть процедуру ручного поиска. Прежде всего рассмотрим поиск научной литературы, который занимает до половины времени, отводимого исследователю на новые разработки. Схема работы библиотеки приведена на рис. 2.20.

В процедуре создания документ (книга, журнал, отчет), поступающий из издательства, получает индекс, или шифр, по специальному классификатору (блок индексатор). В соответствии с индексом формируется и массив оригиналов, размещаемых в хранилищах.

При пользовании библиотекой читатель формирует поисковое предписание (запрос на естественном языке), в котором указывает название документа, его класс, а в ряде случаев — форму представления ответа (оригинал, аннотация, реферат). Далее проводится индексация — переход от естественного языка на язык-шифр, удобный

для функционирования системы. По шифру осуществляется поиск документа, который представляется читателю-заказчику в форме, указанной в запросе.

Следовательно, система выполняет аналитико-синтетическую обработку документа, хранение и поиск, размножение и репродуцирование.

Наибольшую сложность представляет операция аналитико-синтетической обработки документа, в которую входят индексирование, удовлетворение интересов читателя, аннотирование и реферирование. Трудности заключаются в значительной доле неформальных процедур.

Так, для удовлетворения интересов читателя (см. рис. 2.20) вводятся следующие понятия [54].

Информационная потребность (ИП) — потребность в знании, свойственная каждому субъекту и получающая выражение в форме информационных запросов.

Информационный запрос (ИЗ) — сформулированное на естественном языке осознанное требование читателя и адресованное поисковой системе.

ИЗ не всегда полно отражает потребность и может уточняться, приближаясь к ИП. Если уточнение проводится n раз, то процесс имеет описание:

$$\text{ИП} = \lim_{n\to\infty}(\text{ИЗ})^n.$$

ИЗ далее трансформируется в *поисковое предписание* (ПП), являющееся смысловым содержанием информационного запроса. Оценка полученного документа может производиться по степени смысловой близости между содержанием документа и информационным запросом (релевантность) и по степени соответствия документа информационной потребности (пертинентность). Выявление релевантности и пертинентности проводится путем попарного сравнения. Затруднения здесь часто вызываются нечетко сформулированным запросом.

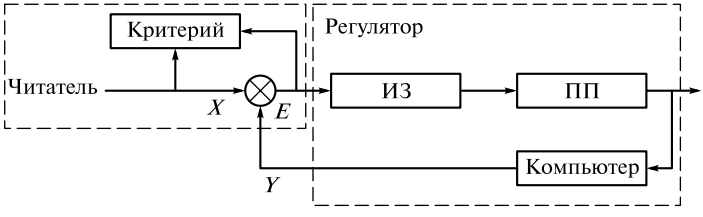


Рис. 2.21. Модель поисковой системы:

X — объективная потребность, входной сигнал; Y — ответ читателю, выходной сигнал; $E = (X - Y)$ — разностный сигнал

На основании сказанного модель поисковой системы можно представить в виде, показанном на рис. 2.21. Сигнал E должен быть сведен к минимуму или нулю на основе принятого критерия качества. Система работает в итеративном режиме при разной скорости работы каналов. На выходе Y, наряду с пертинентными документами, могут иметь место непертинентные документы, обусловливающие ошибку E. Непертинентные документы можно разделить на документы, выданные ошибочно вследствие двух причин:

1) неадекватности выражения информационной потребности в запросе, что служит источником для коррекции информационного запроса;

2) неправильного преобразования информационного запроса в поисковое предписание (коррекция поискового предписания).

В системе могут использоваться элементы обучения.

Из рис. 2.21 видно, что поисковая система по структуре в общем случае многомерна и напоминает структуру системы автоматического управления, однако характеризуется специфичностью математического описания. Это относится прежде всего к информационно-поисковым языкам.

Информационные языки. К таким языкам предъявляются следующие требования. Язык должен быть понятен человеку и компьютеру; обладать в достаточной мере формализованной структурой; быть точным, но не громоздким по структуре во избежание увеличения времени поиска; обладать возможностью модификации по мере изменения языка документов или потребителей документов.

Информационный язык может быть естественным и искусственным.

Естественный язык не имеет строго формализованной структуры и характеризуется неоднозначностью описания (синонимы, омонимы, идиоматические выражения). В связи с этим возникла необходимость создания искусственных информационных языков.

В структуре языка могут быть выделены четыре основные понятия: набор символов, слова, выражения, предложения: *символы* — неделимые знаки, с помощью которых строятся слова; *слова* — наименования единиц языка, образующиеся путем определенного расположения символов; из слов образуются *выражения*, а из выражений — *предложения*.

Словарь, словарный состав языка образуют *лексику* как совокупность слов, входящих в состав языка. Имеют место ключевые слова, под которыми понимаются наиболее существенные слова и словосочетания, выражающие основное смысловое содержание документа и информационного запроса.

Дескрипторы — нормативные ключевые слова, которые отобраны из основного словарного состава языка по определенным правилам и у которых искусственно устранены неоднозначности.

Язык характеризуется смыслом, *семантикой* слов и выражений.

Морфология языка — правила и способы построения и изменения слов.

Синтаксис — правила построения выражений.

Морфология и синтаксис образуют *грамматику* языка.

По сфере применения языки могут быть разделены на следующие группы: специализированные, по определенной тематике (автоматика, вычислительная техника); отраслевые языки; межотраслевые языки; международные языки (УДК, ИСИРЕПАТ).

Простейшим языком является универсальная десятичная классификация (УДК), имеющая иерархическую структуру. УДК охватывает все отрасли знаний и дает возможность выполнять неограниченное деление подклассов без нарушения структуры классификации, строить комбинации подклассов с любой степенью дробности и выражать текстуальные отношения помощью знака «+».

В то же время для УДК характерны следующие недостатки:

- жесткая структура;
- отсутствие новых понятий и длительная процедура внесения изменений и дополнений;
- непоследовательность, а порой и неоднозначность многоаспектной классификации, что обусловлено участием человека в процессе индексации;
- неравномерная детализация в разных направлениях, что затрудняет автоматизацию поиска.

В качестве языка могут быть использованы семантические сети. Поиск осуществляется путем сравнения дерева запросов с деревом заглавий документа по критериям. Критерии должны «работать» и при неполном совпадении в формулировках запросов и документов.

Недостаток языка — сложность автоматизации перевода формализованных текстов документов и запросов с естественного языка на информационный. Для устранения недостатка пошли по пути заблаговременного ручного формирования словарей (тезаурусов) смысловых кодов понятий для последующего использования в индексации документов. В связи с этим наибольшее распространение получили дескрипторные языки в сочетании с УДК. Связи между дескрипторами интерпретируются как конъюнктивные.

Для создания поисковых систем патентной документации создан классификационно-фасеточный комбинированный язык ИСИРЕПАТ.

Поисковые системы: поиск по ключевым словам, гипертексту. К таким поисковым системам следует отнести Google, Rambler. Рассмотрим систему Rambler.

Rambler просматривает первые 650 результатов поиска по запросу и ищет в них интересующий пользователя сайт. Если этот сайт найден, результат поиска содержит его позицию и ссылку на наиболее релевантную страницу сайта. Если же сайт найти не удалось, вы-

даются первые 15 сайтов. В качестве языка запросов выступает язык HTML. Регистр написания поисковых слов и операторов в общем случае значения не имеют.

Слова ищутся во всех формах. Действует оператор AND. Например, запрос «поисковая система» будет обрабатываться как поисковая AND-система. Rambler будет искать те документы, в которых встречаются оба слова.

Слово запроса должно присутствовать в документе. Можно указать запрос в кавычках. По запросу «поисковая система» будут найдены страницы, где имеются оба слова: вначале «поисковая», затем «система».

Существуют стоп-слова — самые частотные слова русского и английского языков, например, предлоги, частицы и артикли. Присутствие этих слов может замедлить поиск и негативно повлиять на полноту результатов. Есть возможность обозначить необходимость этих слов в запросе, взяв запрос в двойные кавычки или воспользовавшись поиском точной фразы в расширенном поиске.

Два запроса, соединенные оператором AND (логическое И) образуют сложный запрос, которому удовлетворяют только те документы, которые одновременно удовлетворяют обоим этим запросам. Так, по запросу поисковая AND-система найдутся лишь те документы, которые содержат оба слова. Сложному запросу, состоящему из двух запросов, соединенных оператором OR (логическое ИЛИ) удовлетворяют все документы, которые содержат хотя бы одно слово.

Оператор NOT (логическое И — НЕ) образует запрос, которому отвечают документы, удовлетворяющие левой части запроса и не удовлетворяющие правой. Так, результатом поиска по запросу «поисковая NOT-система» будут все документы, в которых есть слово «поисковая» и нет слова «система».

Если оператор явно не указан, по умолчанию используется оператор AND: находятся только документы, содержащие все слова запроса.

В языке запросов Rambler операторы AND и NOT имеют более высокий приоритет, поэтому запрос из нескольких слов при обработке сначала группируется по операторам AND и NOT и лишь потом по операторам OR. Изменить порядок группировки можно использованием скобок.

Использование скобок позволяет строить вложенные запросы и передавать их операторам в качестве аргументов, а также перекрывать приоритеты операторов, принятые по умолчанию.

Запрос без скобок «поисковик Yandex/Rambler» эквивалентен запросу поисковик AND Yandex OR Rambler и означает поиск документов, содержащих либо слова «поисковик» и Yandex, либо слово Rambler.

Запрос со скобками поисковик (Yandex/Rambler) равносилен запросу поисковик AND (Yandex OR Rambler), что означает найти до-

кументы, содержащие слово «поисковик» и одно из слов Yandex или Rambler.

Язык запросов Rambler пока не поддерживает поиск строк с использованием метасимволов ('*', '?'), которые обычно используются в значении «любая подстрока» и «произвольный одиночный символ» соответственно. Эти операторы зарезервированы для подобного использования в будущем.

По каждому слову запроса поиск ведется с учетом морфологии, т. е. правил словоизменения соответствующего языка. Rambler понимает и различает слова русского и английского языков.

Поиск ведется по всем формам слова. Например, при поиске по слову «человек» будут также найдены документы, содержащие слова «человеку», «человеком», «человека» и даже «люди». Чтобы провести поиск только по одной определенной форме слова, нужно взять его в двойные кавычки или воспользоваться поиском точной фразы в расширенном поиске.

Если запрос состоит из нескольких слов и при этом некоторые из них вообще не удалось найти в Интернете, то выдаются результаты поиска по частичному запросу, из которого исключены слова, отсутствующие в Интернете. При этом на странице результатов поиска выдается соответствующая диагностика.

Автоматическое аннотирование и реферирование. Задачу удовлетворения запроса относительно легко решить, если читателю надо или установить факт наличия документа, или получить документ (копию). Однако часто для читателя представляет интерес краткое содержание документа в виде реферата.

Осуществление реферирования вручную — задача трудоемкая. К тому же на него тратится труд квалифицированных референтов. В связи с этим актуальна задача автоматизации реферирования. Для этого применяются следующие методы:

- статистические;
- использование данных по грамматической и лексической структуре текста;
- применение искусственных информационных языков.

Аннотирование и реферирование проводят в два этапа.

1) выделяют ключевые слова (дескрипторы);

2) строят на их основе предложения с применением грамматических, синтаксических, лексических правил и основ семантики.

Используются языки грамматико-семантической обработки текста.

С автоматическим аннотированием и реферированием перекликаются работы по автоматическому переводу текста [46]. В нем выделяют нулевой, черновой и беловой переводы.

Нулевой перевод включает в себя сопоставление слов одного и другого языка. Проводится анализ слов начального языка, перевод отдельных слов, формирование предложений конечного языка в со-

ответствии с его правилами. Этот этап наиболее прост и реализуется во многих программных продуктах. На рис. 2.22 приведен пример такого перевода в интернет-файле mail.ru. Данный этап соответствует примерно первому этапу автоматического аннотирования и реферирования.

Черновой перевод учитывает полный смысл, заложенный автором исходного текста. Первоначально создаются варианты перевода, выявляются дефектные варианты, отбирается наилучший вариант. Затем осуществляются перевыражение мысли с помощью языка, приведение новых фраз в соответствие с языковыми нормами (правильно — неправильно), поправка на УЗУС (общепринятое носителями данного языка употребление языковых единиц).

Трудности перевода заключаются в неоднозначности самого процесса. В тексте выделяют функцию (потенциал воздействия), содержание и структуру. Чем-то при переводе надо «пожертвовать». Функциональность языка никогда не приносится в жертву. Структурно-семантические отступления от оригинала вызваны двумя основными причинами: либо нет языковых средств для копирования (квазитрансформация), либо копирование приводит к непониманию на конечном языке (трансформация). На этом этапе проводится синтаксический, грамматический и лексический анализы текста с использованием грамматических, синтаксических и лексических правил.

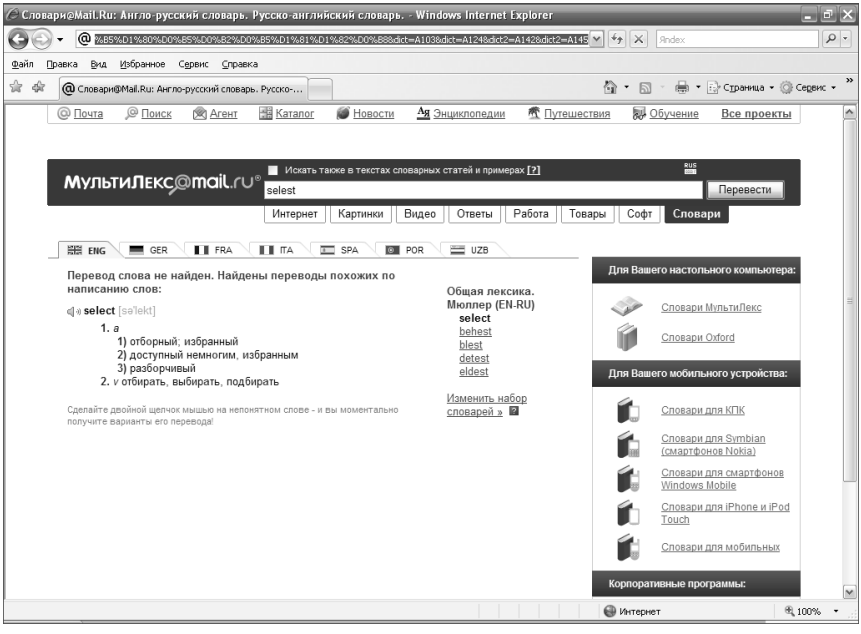


Рис. 2.22. Перевод с помощью Интернета

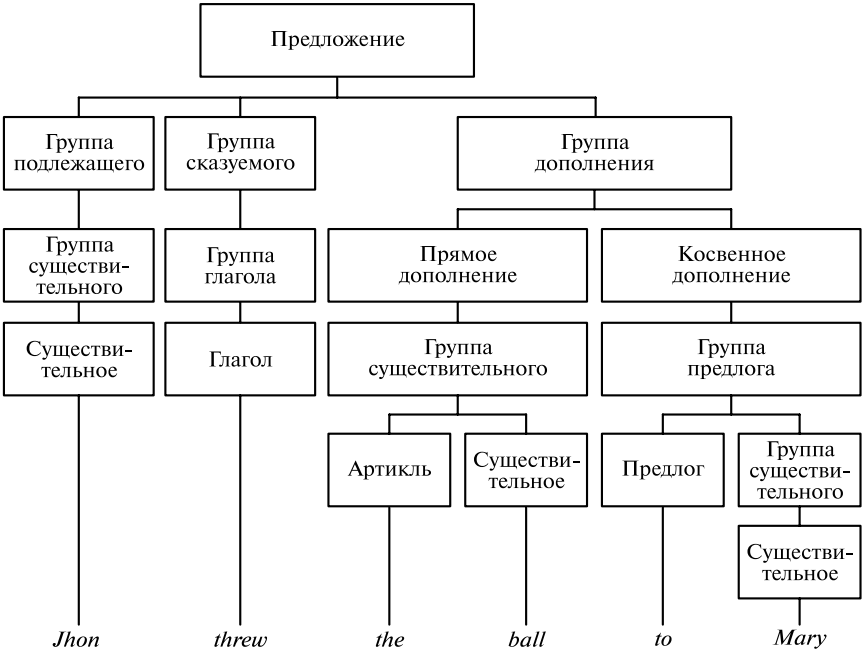


Рис. 2.23. Дерево грамматического разбора

Беловой перевод во многом повторяет работы предыдущего этапа, однако требует высокого уровня интеллекта и выполняется только человеком, возможно в режиме диалога с компьютером. При беловом переводе сохраняется узнаваемость стиля изложения материала автора исходного текста.

Структура грамматического разбора перекликается с семантическими сетями. На рис. 2.23 приведена структура предложения на английском языке *John threw the ball to Mary*. Правило (порядок слов) немного меняется при использовании страдательного залога The ball was throw Mary by John.

Существует ряд программных продуктов-переводчиков. Из них одним из успешных является пакет Promt.

2.6. Интеллектуальные системы управления

Классы современных интеллектуальных систем управления (ИСУ) характеризуются нахождением на «стыке» предметных областей.

Примером такой интеграции являются современные адаптивные автоматизированные системы управления.

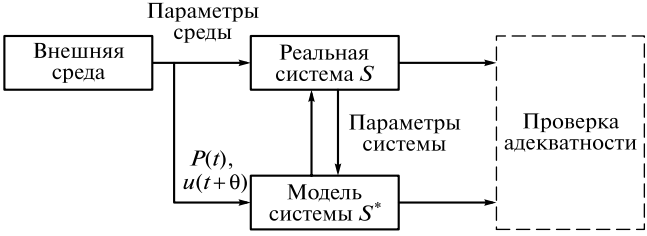


Рис. 2.24. Взаимосвязь предприятия и внешней среды:
P — план; u — управление; t — время; θ — запаздывание

Адаптивные системы отличаются от неадаптивных тем, что они в процессе функционирования получают дополнительную информацию о поведении среды или влиянии среды на параметры, структуру и цели системы и компенсируют это влияние.

В адаптивных автоматизированных системах управления цель качественно меняется (изменяется состав вектора цели) за счет оперативного перехода на выпуск новой продукции. Учет изменения состава вектора цели и тем более выработка самой новой цели относятся к творческим процедурам, и поэтому такие системы могут быть отнесены к интеллектуальным.

Если цель вырабатывается внутри системы, то это интеллектуальная система, если задана извне — интеллектная система. Далее обсудим вариант интеллектуальных систем при передаче на рынок новой пробной продукции и интеллектную систему перехода на выпуск новой продукции.

Первоначально покажем, что изменения целей организационной системы управления вызваны сильным влиянием внешней среды. Соотношение внешней рыночной среды и системы управления показано на рис. 2.24.

В общем виде модель процесса автоматизированного управления предприятием

$$S = S^*(y(t),\ P(t),\ R(t),\ b(t),\ C(t),\ A(t),\ u(t+\theta)); \qquad (2.3)$$

$$u(t+\theta) = u(P(t+\theta),\ R(t+\theta),\ K(t)), \qquad (2.4)$$

где S^* — теоретико-множественная модель процесса управления предприятием; $y(t)$ — выпуск продукции; $P(t)$ — план выпуска продукции; $R(t)$ — функция спроса на продукцию; $b(t)$ — ограничения на ресурсы; $C(t)$ — маржинальная прибыль; $A(t)$ — нормы расхода ресурсов; $u(t)$ — управляющее воздействие; $K(t)$ — критерий; θ — момент запаздывания (возникновения спроса).

В свою очередь,

$$K(t) = K(C(t)); \qquad (2.5)$$

$$b(t) = \{b_m(t),\ b_\psi(t)\}, \qquad (2.6)$$

где $b_m(t)$ — материальные ресурсы; $b_\psi(t)$ — остальные виды ресурсов.

В общем случае среда V описывается выражением

$$V = V^*(R, \text{Ц}, RS), \tag{2.7}$$

где R — спрос на продукцию; Ц — цены продукции и ресурсов; RS — ресурсы.

Наибольший интерес представляет случай, когда период стабильного спроса соизмерим с периодом стабильного выпуска продукции. Изменения параметров (спрос, цена, ресурсное обеспечение) должна скомпенсировать система управления.

В общем случае закон изменения, например, спроса может быть произвольным. В то же время для системы управления сложнее всего отработать скачкообразные изменения сигнала или параметра. В силу этого целесообразно описать изменения в виде системы скачков.

Появление нового заказа (изменение заказа, размещение нового заказа) $R_{\text{нов}}(t)$ можно отразить выражением

$$R_{\text{нов}}(t) = R^*_{\text{нов}} 1(t - \theta), \tag{2.8}$$

где $R_{\text{нов}}(t)$ — текущая величина спроса на новую продукцию; $R^*_{\text{нов}}$ — установившееся значение спроса; $1(t)$ — единичная функция.

Изменения вида (2.8) вызывают изменения в структурных связях системы в процессе ее эксплуатации, т. е. требуют построения адаптивной (приспосабливающейся) автоматизированной системы управления.

Компенсировать изменения параметров можно оперативным переходом на выпуск новой продукции, что связано не только с учетом динамических свойств, но и с учетом структурных изменений. Компенсация — нестационарный режим, требующий адаптации.

Под адаптацией понимается процесс изменения структуры, параметров и алгоритмов системы на основе дополнительной информации, получаемой в процессе управления, в целях достижения оптимального состояния или поведения системы при начальной неопределенности и изменяющихся условиях работы, определяемых во взаимодействии с внешней средой.

При этом возникают две задачи: определение цели вида (2.8) и реализация выработанной цели.

Обсудим процедуру *определения цели*. Хотя эта процедура носит творческий характер и исследуется методами системного анализа, составим один из вариантов формализованной модели при наличии на предприятии системы управления качеством.

Данные для определения цели получаются из двух групп источников: внешней и внутренней:

- группу внешних источников определяют потребители (среда). Маркетологи в процессе сбора числовой информации часто получают сообщения о недостатках продукции, реже — предложения по ее улучшению;

- группу внутренних источников определяют разработчики продукции, которые фиксируют недостатки продукции в процессе ее массовой эксплуатации, предлагают новые, прогрессивные конструкции и технологии. К внутренним источникам следует отнести и результаты фундаментальных исследований, на основе которых может быть создана принципиально новая продукция.

Перечисленные источники создают предпосылки для оперативного перехода на выпуск нового вида продукции.

Чтобы окончательно удостовериться в полезности такого перехода, нужно изготовить и выпустить на рынок малую партию новой продукции (активное воздействие на рынок). Процедуру потребления этой новой продукции маркетологи отслеживают обычными методами. Одним из способов математического описания процедуры может быть следующий.

В общем случае реакция рынка или мера отклика определяется вектором спроса $R = R(\text{Ц}, \text{З}, Q, AS, E)$, где векторы Ц — цена продукции; З — затраты на продвижение продукции и товародвижение; Q — качество продукции; AS — ассортимент продукции; E — переменные, не зависящие от производителя.

Спрос может измеряться как сбыт $S_j(t)$ продукции j на одну торговую точку, доход $Q_j(t)$, доля рынка $m_j(t)$, потенциальная прибыль $W_j(t)$, где t — время. Используют марковские и пуассоновские модели.

Пусть $P_{ij}(t)$ — вероятность переходов в марковской цепи; $P(t)$, $\mathrm{d}(t)$ — количество торговых точек, в которых продаются продукция фирмы и конкурирующая продукция; $T(t)$ суммарный сбыт конкурирующей продукции; $Q(t)$ — сбыт всей продукции; $u_{ij}(t)$ — случайная помеха; $\pi_j(t)$ — вероятность нахождения продукции на рынке; t — время.

Тогда можно составить следующую модель:

$$P_{ij}(t) = (K_2 P(t)/(\mathrm{d}(t) + P(t))) + u_{ij}(t);$$

$$\pi_j(t) = P_{ij}(t-1)\pi_j(t-1) + P_{ij}(t)\pi_j(t-1);$$

$$\pi_j(t) = m_j(t);\ m_j(t) = S_j(t)/T(t);$$

$$S_j(t) = S_j(t)/P(t);\ Q(t) = T(t)(Q(t) - P(t));$$

$$S(t) = K_2 T(t)P(t)/(Q(t) + (1 + K_2 - K_1)P(t))) + u^0(t);$$

$$S(t)/Q(t) - K_2 + (K_1 - K_2)m_j(t) + u^0(t),$$

где K_2, K_2 — параметры.

Если новая продукция получила одобрение рынка, возникает задача оперативного перехода на широкий выпуск нового вида продукции.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Как классифицируют экспертные системы?
2. Почему в экспертных системах отсутствуют лексический и лингвистический процессоры?
3. Дайте определение статической ЭС и ЭСРВ. В чем их различие?
4. Что такое знания?
5. Каково назначение искусственных нейронных сетей (ИНС)?
6. Что такое биологический нейрон и базовый процессорный элемент (БПЭ)?
7. Как классифицируют ИНС?
8. Дайте базовое математическое описание БПЭ и ИНС.
9. Сравните возможности экспертных систем и ИНС.
10. Дайте определение эволюционных вычислений.
11. В чем заключаются принципиальные отличия интеллектуальных агентов от объектно-ориентированных приложений?
12. Перечислите основные характеристики интеллектуальных агентов.
13. Дайте определение мультиагентной системы.
14. Перечислите основные направления исследований в области МАС.
15. Дайте характеристику основных видов архитектур МАС.
16. Почему нужна адаптация в автоматизированных системах управления?

РАЗДЕЛ II
ИНТЕЛЛЕКТУАЛЬНЫЕ ТЕХНОЛОГИИ

Глава 3
КОНЦЕПТУАЛИЗАЦИЯ КАК СПЕЦИФИЧЕСКИЙ ЭТАП ТЕХНОЛОГИИ ПРОЕКТИРОВАНИЯ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

3.1. Онтология как система терминов

Web-технологии. В разделе I уже был рассмотрен этап идентификации процедуры проектирования интеллектуальных систем. Этапы концептуализации и формализации дискретных и непрерывных системах существенно различаются.

Концептуализация в непрерывных системах фактически отсутствует и более характерна для дискретных систем. Продуктом этапа формализации для непрерывных систем являются функциональные зависимости, тогда как для дискретных систем — обобщенные логические зависимости.

Для дискретных систем существует тесная связь этапов концептуализации и формализации. В связи с этим названные этапы рассмотрим совместно. Обсудим прежде всего понятие «онтология».

Наиболее широко используемым определением является определение, данное в работах [34, 40].

Онтология — явная спецификация (список) концептуализации, на которой основано формальное представление знаний.

Концептуализация — выбор определенных объектов, понятий, отношений и других терминов для описания состояния мира.

Концептуализация, следовательно, описывает абстрактную и упрощенную точку зрения на предметную область.

Онтологии появились в искусственном интеллекте и первоначально использовались только при построении систем, основанных на больших базах знаний, разрабатываемых и развиваемых разными, зачастую независимыми группами исследователей. Впоследствии онтологии стали успешно использоваться в областях, выходящих далеко за границы систем, основанных на знаниях. Онтологии незаме-

нимы при построении систем, чьи компоненты должны быть универсальными, легко расширяться или работать в быстро развивающихся отрытых системах, в которых нет стандартов, или они не описывают все требуемые характеристики.

Наличие формального описания терминов и предметных областей стали неотъемлемым атрибутом современных сетевых технологий (CORBA, XML). Способы использования и содержание этих формальных описаний выходят за рамки спецификаций и вызывают сложности у программистов. Однако явное введение формальной семантики в современные технологии обеспечивает качественно новые возможности программного обеспечения любого типа — от автономного агента и сетевого сервиса до традиционных приложений, основанных на интерфейсе с пользователем.

Примерами могут служить синтаксический и семантический Web (рис. 3.1).

Все современные Web-технологии можно разделить на три уровня:

1) уровень представления данных;

2) уровень, определяющий синтаксические ограничения и ограничения на последовательность, в которой должны передаваться со-

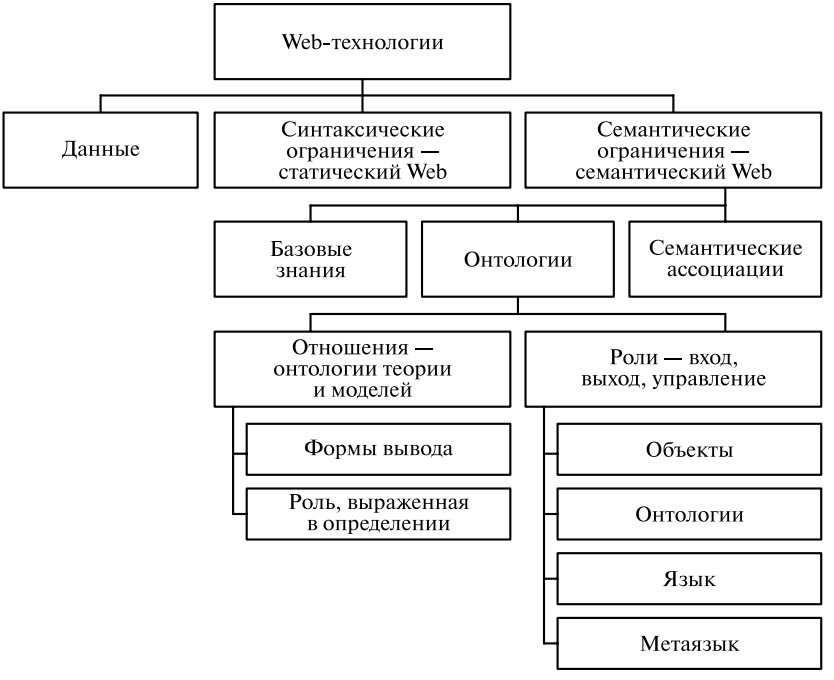


Рис. 3.1. Состав Web-технологий

общения или просматриваться документы, условно названный синтаксическим Web;

3) уровень, соответствующий семантическому Web.

К уровню представления данных можно отнести стандарты кодирования, например, Unicode, разметки (XML, HTML, XHTML), стандарты для представления ссылок на ресурсы (URI) и их компоненты (XPointer), стандарты на передаваемые сообщения (SOAP) и другие подобные стандарты. На этом уровне предполагается обработка данных по фиксированным сценариям при словаре терминов, понятном всем заинтересованным сторонам. В зависимости от способа обработки ресурса стандартный словарь может сокращаться или расширяться для различных участников взаимодействия. Например, для нормального отображения HTML-страницы Web-браузеру совсем не обязательно понимать предложения на английском языке, в отличие от приложения, проверяющего грамматику на HTML-странице, или выполняющего интерпретацию страницы.

Уровень синтаксического Web предполагает более сложные сценарии обработки ресурсов, которые иногда сами являются ресурсами, внешними относительно приложения. Однако, как и на предыдущем уровне, ресурсы должны формулироваться в стандартном словаре. К этому уровню относятся стандарты, определяющие порядок обмена сообщения с сервисом, и шаблоны этих сообщений (WSDL). Особое место занимает стандарт XML-Scheme, определяющий синтаксические ограничения на разметку XML-документов. Эти ограничения могут использоваться разными способами. В зависимости от способа использования этих ограничений их можно отнести либо к синтаксическому, либо к семантическому Web. Например, стандарт XML-Scheme можно отнести к синтаксическому Web при использовании схем для управления пользовательским интерфейсом и проверки синтаксической правильности данных и документов.

Самым верхним уровнем являются стандарты семантического Web. Данные, представленные в этих стандартах, являются универсальными и предназначены для представления семантических знаний. В этом верхнем уровне можно выделить два подуровня: подуровень семантических аннотаций и подуровень онтологий. В семантических знаниях, описывающих ресурс в семантическом Web, выделяют три компонента:

1) базовые знания (background knowledge) — знания, необходимые для понимания содержимого документа или описания, имеющиеся у всех заинтересованных сторон и поэтому не представленные явно;

2) онтология (ontology) — множество понятий предметной области с формальными определениями, описывающими общие знания об этой области;

3) семантические аннотации (annotation), описывающие формально содержание документа или любой другой ресурс на основе онто-

логий и базовых знаний, связанных некоторым образом с этим ресурсом.

К стандартам, позволяющим представлять семантические аннотации, относятся RDF, SHOE, XML-Scheme. В качестве примера использования семантических аннотаций можно привести применение ограничений, представленных в XML-Scheme при подборе в библиотеке компонентов ресурсов, способных выполнять определенные задачи, идентифицируемые по набору стандартных полей. Задачи могут относиться к различным заранее неизвестным типам ресурсов (ссылки на сетевые сервисы, агенты, базы знаний, удаленные интерфейсы).

Стандарт RDF-S и его расширения используются в семантическом Web для представления онтологий. Впервые онтологии на Web были явно реализованы в SHOE, являющимся расширением HTML. Из-за того, что SHOE разрабатывался до появления семантического Web, в него явно не введены стандарты, соответствующие описанным уровням семантического Web, а элементы разметки, предназначенные для определения онтологий, семантических аннотаций и представления данных, реализованы в рамках одного стандарта, неразрывно связанного с HTML.

Онтологии являются ключевым компонентом семантического Web, в задачи которого входит определение общих знаний об интерпретации семантических аннотаций, не связанных с индивидуальными свойствами конкретных ресурсов.

Семантический Web был определен консорциумом W3C как подход к абстрактному представлению данных на Web и основан на стандартах Resource Description Framework (RDF) и их расширениях. Использование семантического Web позволяет улучшить и упростить совместную работу людей и компьютеров за счет описания семантических значений представляемой информации. На практике это означает использование семантических аннотаций на базе RDF для обеспечения качественно новых функциональных возможностей при работе в следующих областях:

- для сопровождения XML-документов и других видов взаимодействий между компонентами системы;
- для описания возможностей, назначения, правил использования Web-сервисов, мобильных и интеллектуальных агентов и методов из внешних библиотек;
- для организации при взаимодействии с человеком интеллектуальных интерфейсов, в которых компьютер, если и не является экспертом в данной области, то, по меньшей мере, способен понимать обрабатываемую информацию.

Качественное расширение словаря, положенного в основу документов XML или любых других взаимодействий с внешним миром, является традиционной областью применения семантических аннотаций. В частности, децентрализованное расширение сценариев и содержания сообщений при полном отсутствии неявной и неописан-

ной формально информации является одним из требований к новому стандарту XMLP.

Применение семантических аннотаций при описании сервисов, агентов и методов имеет существенные особенности. Традиционно эти аннотации представляются в специальных терминах, стандартизованных в языках WSDL, DAML-S, UPML, KARL и многих других. Особое место в применении семантических аннотаций занимает организация интеллектуальных интерфейсов. Наиболее развитым направлением здесь является семантический поиск, основанный на использовании семантических аннотаций HTML-страниц и используемых вместе с лексическими онтологиями WordNet, EuroWordNet, обеспечивающими возможность учета отношений между словами в естественном языке.

Стандарт RDF позволяет представлять метаданные о любых сетевых ресурсах. В RDF ресурсами являются как абстрактные свойства, классы, объекты, множества, так и реальные объекты (файлы, наборы элементов разметки, Web-сервисы), т.е. объекты, которые можно создать средствами RDF и описать с помощью URI или XPointer. Эта способность RDF позволяет описывать связи между ресурсами и значения свойств отдельных ресурсов, например принадлежность ресурса к определенному классу. Семантический Web отделяет семантические аннотации от данных, непосредственно используемых при работе с сервисом или документом.

Онтология имеет двойственную природу. С одной стороны, она определяет концептуализацию на семантическом уровне как отображение во всех возможных ситуациях множества объектов из определяемой области в множество отношений. С другой стороны, частично формализует эту концептуализацию через онтологическую теорию и онтологические соглашения в формальные знания о связях значений отношений, классов и других определяемых терминов, не зависящих от конкретной ситуации.

При невозможности и отсутствии необходимости полного формального описания концептуализации идентичные онтологические теории и соглашения можно всегда сопоставить с различными возможными концептуализациями. Соотношение между этими двумя компонентами онтологии определяется предполагаемыми сценариями использования онтологий. В крайних случаях концептуализация может описываться полностью на неформальном языке, а онтологическая теория может определять только количество аргументов отношения или тип значения некоторых аргументов.

В то же время термины могут полностью определяться сложными логическими аксиомами и происходить от терминов ранее известных систем. Поэтому при построении онтологии необходимо определить: кто, как и когда ее должен и может использовать.

В соответствии с классификацией N. Guarino различают два этапа работы с онтологиями:

1) разработка системы, где онтологии либо участвуют в построении системы как базы знаний, либо используются программистом при концептуальном анализе, анализе различных подходов;

2) использование системы.

Принципы построения онтологий и подходы к их формированию. Рассматривая процесс построения онтологии с практической точки зрения, T. Gruber выделяет следующие пять принципов, определяющих содержание онтологий [40].

Принцип ясности требует от онтологии эффективности передачи подразумеваемых значений терминов. Все определения должны быть целенаправленными, но они не должны зависеть от внешних факторов, которые T. Gruber условно назвал социальным контекстом и контекстом вычислений задачи. Для обеспечения этого условия логические аксиомы представляются на формальных языках. Рекомендуется по возможности давать полные логические определения, описывающие необходимые и достаточные свойства терминов. Все онтологии должны пониматься людьми, и поэтому с требованием ясности непосредственно связана необходимость сопровождения всех определений документацией на естественном языке.

Принцип согласованности устанавливает согласованность аксиом онтологической теории и описаний всех терминов, использованных в онтологии. Требуется как традиционная непротиворечивость онтологической теории, так и непротиворечивость между утверждениями на естественном языке и знаниями, выводимыми из онтологической теории.

Принцип расширяемости онтологии указывает, что при разработке онтологии необходимо учитывать возможные будущие применения создаваемой онтологии как универсального способа описания предметной области. Онтологии должны разрабатываться с учетом возможности уточнения значений терминов и определения новых терминов в онтологии, что позволит применять их к новым задачам без внесения изменений в ранее определенные аксиомы теории.

Принцип минимизации вопросов кодирования устанавливает предпочтение решений, имеющих минимальный уклон в вопросы кодирования данных. Считается, что концептуализация должна определяться на уровне знаний независимо от конкретных способов кодирования на символьном уровне. Тогда знания, представленные в определенной онтологии, смогут использоваться системами с различными форматами представления данных. Косвенным результатом этого принципа является предпочтение вариантов, основанных на явном указании списка неопределяемых объектов, по сравнению с вариантами, ограничивающимися только описанием их числа.

Принцип минимизации онтологических соглашений определяет, что онтологии должны налагать минимальные ограничения на моделируемую предметную область, но достаточные для решения поставленных задач. Следовательно, нужно использовать наиболее «по-

верхностную» и общую теорию, которая содержит только термины, реально используемые в базах знаний.

Рассмотренные принципы являются полезными рекомендациями для построения универсальных онтологий, поддерживающих возможность расширения, но не определяют ни внутренней структуры онтологии, ни связи онтологических знаний с другими знаниями, используемыми системой.

Существует два подхода [67, 71] к формированию определений: подход, основанный на определении через отношения, и подход, основанный на роли информации.

Онтологии, построенные по первому подходу, предназначены для формального описания теорий и моделей представления предметной области. Будем называть их онтологиями теорий и моделей. Форматы, используемые для их представления, имеют три типа новых конструкций:

1) конструкции, предназначенные для реализации определенных форм вывода (наследования, проверки ограничений, вывода в дескриптивной логике);

2) конструкции, определяющие роль определенного выражения в определении. Можно выделить множество ролей выражения: определяющую аксиому; выражение, устанавливающее связь между терминами; достаточное условие;

3) конструкции для представления документации, комментариев и примеров.

Второй подход традиционно используется для определения элементов онтологии через их функциональность, что является особенно полезным при определении функциональных спецификаций задач, методов и сервисов. В этом случае определение строится вокруг таких типовых ролей информации, как входная, выходная, внутренняя и управляющая информации. Эти роли связываются с определенными онтологическими элементами. Однако такой подход к определению на основании метаролей определяемых элементов или других онтологических элементов относительно определяемого элемента намного шире. Полностью его потенциал еще не исследован.

Формат RDFS и типы онтологий. Его применимость к построению онтологии [67] частично была показана в формате RDFS(FA). В нем все определяемые классы и отношения в RDF-S распределены по четырем уровням:

1) уровень объектов соответствует объектам и отношениям, определяемым пользователем на основании онтологии;

2) на уровне онтологий определяются типовые элементы, используемые при построении описаний в рамках ограниченной предметной области;

3) уровень языка содержит примитивы языка, задействованные при определении онтологий и заданные на отношениях и классах онтологического уровня;

4) на уровне метаязыка определяются конструкции, используемые для описания конструкций языкового уровня.

В формате RDFS(FA) все стандартные примитивы RDF-S продублированы на всех четырех уровнях под разными именами. Явным описанием ролей конструкций разработчики попытались частично преодолеть проблемы RDF-S. В частности, класс rdf:Class в RDFS(FA) не содержит себя среди своих элементов. Однозначно фиксирована связь между классами rdf:Class и rdf:Resource. А семантика других стандартных свойств RDF-S задана более формально.

Онтологии тесно связаны с логикой рассуждений. В системах, основанных на знаниях, последние можно разделить на две группы:

1) знания о связях между объектами, структурирующих предметную область;

2) знания, описывающие модели рассуждений при решении задач.

Описания универсальных моделей рассуждений, используемых различными задачами, получили название «онтологии методов». Они должны решать три основные задачи: описывать входные данные и полученный результат; определять выполнимость процесса решения; обеспечивать возможность расширения и специализации метода.

Все форматы представления онтологии можно классифицировать по моделям представления знаний и свойствам онтологий: однозначности, формальности, назначению (рис. 3.2).

Лингвистические онтологии и тезаурусы являются онтологиями, устанавливающими соответствие между значением и набором терминов. Тезаурусы — это онтологии, состоящие из набора синонимов (синсетов) и отношений между ними. Для синсетов обычно описы-

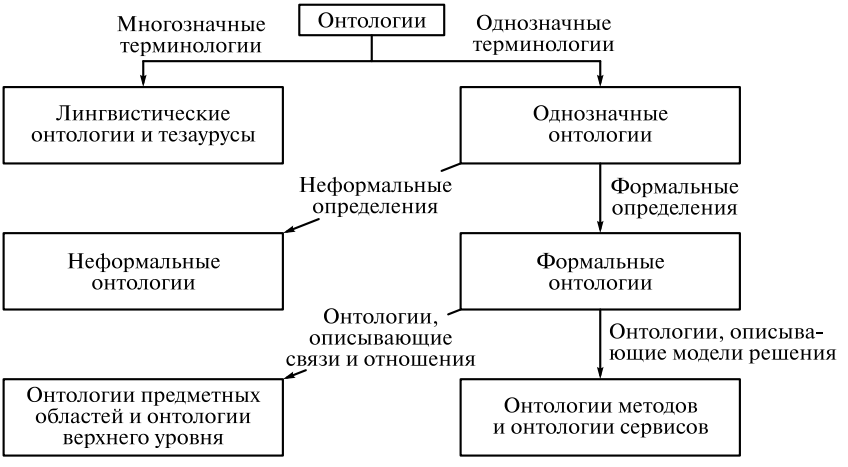


Рис. 3.2. Типы онтологий

ваются отношения «быть антонимом» и «быть гипернимом». (Гипернним — более общее понятие. Например, транспортное средство является гипернимом к автомобилю.)

Неформальные онтологии содержат только неформальные определения терминов на естественном языке. Примером неформальных онтологий являются схемы XML.

Очевидным применением онтологий методов является описание функциональности Web-сервисов. Онтологии сервисов описывают сервис на трех уровнях: функциональной спецификации (тип входных и выходных параметров); операционной спецификации (последовательность выполнения элементарных действий для преобразования входных параметров в результат); сетевых протоколов вызова сервиса.

Простейшие онтологии основаны на семантических сетях, построенных на двух отношениях: отношении part-of и отношении is-a. Примером такой онтологии является GeneOntology, определяющая десятки тысяч терминов в области генной инженерии.

Одно из наиболее перспективных применений онтологий — их использование для описания сетевых ресурсов, что позволяет автоматизировать их обработку и обеспечить возможность их динамического расширения.

В предположении, что онтология сформирована, обсудим основные положения общей теории дискретных интеллектуальных информационных систем — теорию представления знаний.

3.2. Теория представления знаний

Символьные методы. В простейшем случае знания и данные могут быть представлены неструктурированно («врассыпную»). В этом случае для описания процессов можно использовать символьные методы [61, 64].

Теория формальных символьных систем включает в себя методы построения формальных моделей для представления знаний о предметных областях и механизмы рассуждений (поиска), основанные на использовании символов в качестве средства для описания внешнего мира.

Методы основаны на гипотезе Ньюэлла и Саймона (1976 г.): физическая символьная система и поиск являются необходимой и достаточной характеристикой интеллекта. Выдвижение данной гипотезы и дальнейшие работы, направленные на обоснование условий ее достаточности и необходимости, привели к формированию трех важнейших методологических принципов научных исследований интеллектуальной деятельности [34]:

1) использование символов и символьных систем в качестве средства для описания мира;

2) разработка механизмов перебора, в том числе и эвристического, для исследования границ потенциальных умозаключений таких систем;

3) отвлеченность когнитивной архитектуры: правильно построенная символьная система может проявлять интеллект независимо от средств реализации.

Физическая реализация символьной системы не влияет на ее функционирование: любая среда, успешно реализующая правильные операции символьной обработки, достигает уровня интеллекта независимо от того, составлена она из логических цепей, нейронов и т. д.

Гипотеза о физической символьной системе определила основные вопросы исследований в области интеллектуальных систем:

- определение структур символов и операций, необходимых для интеллектуального решения задач;
- разработка стратегий эффективного и правильного поиска потенциальных решений, сгенерированных этими структурами.

В интеллектуальных системах символами (языковыми знаками) представляются все формы знаний, опыта, понятий и причинности.

Знак — материальный предмет (процесс, явление, событие), выступающий в качестве представления некоторого другого предмета, свойства или отношения и используемый для приобретения, хранения, переработки и передачи сообщений (семантической информации, знаний).

Языковые знаки подразделяются на знаки естественных языков и знаки искусственных языков. Языковые знаки не функционируют независимо друг от друга, а образуют систему, правила которой определяют закономерности их построения (правила грамматики или синтаксиса), осмысления (правила смысла или значений символов) и употребления [37].

Символы вместе со своей семантикой могут использоваться для построения формальных систем. Осуществить постановку задачи формально означает выразить на некотором формальном языке все знания о предметной области, необходимые для решения задачи. Формальный язык требует рассмотрения двух его неотъемлемых частей: синтаксиса и семантики.

Синтаксис языка описывает допустимые в языке предложения, состоящие из последовательностей символов, принадлежащих определенному множеству (алфавиту). Синтаксис языка позволяет отличить предложения, принадлежащие языку, от предложений, ему не принадлежащих.

Семантика языка определяет смысл этих предложений, сопоставляя символы языка с объектами реального мира, а предложения — с отношениями между объектами [39].

Представление знаний — выражение на некотором формальном языке свойств различных объектов и закономерностей, важных для

решения прикладных задач и организации работы пользователей с интеллектуальной системой. Любой современный метод представления знаний — это совокупность взаимосвязанных средств формального описания знаний и оперирования (манипулирования) этими описаниями.

Формальная система представляет собой совокупность абстрактных объектов, не связанных с внешним миром, в которых представлены правила оперирования множеством символов в чисто синтаксической трактовке без учета смыслового содержания.

Формальная система определена в следующих случаях:

- задан конечный алфавит — конечное множество символов;
- определена процедура построения формул или слов формальной системы;
- выделено некоторое множество формул — аксиом;
- задано конечное множество правил вывода, которые позволяют получить из некоторого конечного множества формул другое множество формул.

В алфавите формальной системы различают константы, переменные и операторы.

Одна и та же формальная система может служить моделью различных конкретных ситуаций и предметных областей. Эта универсальность как результат абстрагирования от конкретной реальности является главным достоинством формальных методов представления знаний.

Чтобы использовать формальную систему для решения конкретной задачи, необходимо провести процедуру интерпретации, обеспечивающую распространение исходных положений какой-либо универсальной формальной системы на реальный мир. Интерпретация придает смысл каждому символу формальной системы и устанавливает взаимнооднозначное соответствие между символами формальной системы и реальными объектами.

Логические исчисления. В основе всех формальных методов использования знаний лежат те или иные логические исчисления. Отличительной чертой логических методов являются единственность теоретического обоснования и возможность реализации системы формально точных определений и выводов.

Логическое исчисление — система, в которой существует некоторое число исходных объектов, в том числе аксиом, и некоторое число правил вывода новых объектов из исходных объектов.

При использовании логических моделей все знания о предметной области описываются в виде формул этого исчисления или правил вывода.

К классическим логическим исчислениям относят исчисления предикатов и исчисления высказываний.

Логика как система представления и использования знаний конструируется из языка предикатов, нескольких теорем, описывающих

отношения в терминах этого языка и закладываемых в базис логически полной системы, и правил вывода.

Широкое применение логики предикатов началось с 1965 г., когда Дж. Робинсон предложил способ машинного вывода, названный принципом резолюции.

Логика предикатов в соответствии со сложностью синтаксических правил языка имеет иерархическую организацию, которую образуют так называемые предикаты первого порядка, предикаты второго порядка и т. д. Логику предикатов первого порядка, о которой и пойдет речь, часто называют классическим исчислением предикатов.

Основные формализмы представления предикатами:

- терм, устанавливающий соответствие знаковых символов описываемому объекту;
- предикат для описания отношений сущностей в виде реляционной формулы, содержащей в себе термы.

Предикаты могут не иметь аргументов, и тогда они являются полными аналогами логических переменных. В общем случае предикаты могут иметь в качестве аргументов объектные константы и переменные, область значений которых может быть даже бесконечной. Истинностное значение предикатов в этом случае зависит от значения его аргументов.

В логике предикатов используются кванторы, которые позволяют делать высказывания в более лаконичной форме о множестве связанных определенными отношениями объектных переменных.

Вывод в логике предикатов представляет собой процедуру, которая из заданной группы выражений выводит выражение, отличное от начальных. Если посылка является истиной, то должно гарантироваться, что выражение, выведенное из них в соответствии с правилами вывода, также будет истиной. В логике предикатов используется правило, которое из двух выражений «A» и «A → B» выводит новое выражение «B». Правило вывода, выводящее подобным образом новое выражение, называется правилом дедуктивных выводов. В логике предикатов имеются и универсальные правила вывода, которые оперируют с формулами, содержащими свободные переменные.

Предикат, все термы которого являются термами-константами, называется высказыванием. В общем случае при употреблении термина «предикат» подразумевается, что в него входит терм-переменная.

Синтаксис *логики высказываний* прост и имеет прямые синтаксические и семантические аналоги в естественных языках, что облегчает понимание данной логики. Символами языка логики высказываний, составляющими ее алфавит, являются логические константы ИСТИНА и ЛОЖЬ, логические переменные, логические связи И, ИЛИ, НЕ, ЭКВИВАЛЕНТНО, ВЛЕЧЕТ и круглые скобки. Значениями логических переменных являются логические константы. Фор-

мулы, составленные в соответствии с правилами логики высказываний, называют правильно построенными формулами.

В логике высказываний, как в частном случае логики предикатов, интерпретация определяет семантику формул (предложений, высказываний) путем сопоставления переменных в формулах со свойствами объектов предметной области, а отношений этих свойств — формулами. Это позволяет по значению формул (после подстановки вместо переменных конкретных значений свойств) судить о наличии или отсутствии у предметной области тех или иных совокупных свойств или отношений. Если дана какая-то формула, то подстановка в формулу констант вместо ее переменных называется унификацией.

Семантика простейших формул логики высказываний близка к соответствующим высказываниям на естественном языке. В то же время выразительные возможности логики высказываний очень невысоки и требуется много формул логики высказываний для описания даже простых предметных областей.

Логика предикатов существенно более выразительна, чем логика высказываний, и позволяет представлять знания гораздо более компактно.

Исчисление предикатов включает в себя все формулы исчисления высказываний, а также формулы, содержащие предикатные символы с кванторами (общности и существования).

При рассмотрении возможностей практических применений логики предикатов и их модификаций необходимо учитывать такие свойства исчисления, как полнота и непротиворечивость. Для того чтобы установить непротиворечивость исчисления, в общем случае требуется осуществить вывод всех теорем. Только при отсутствии среди них любых двух теорем, одна из которых является отрицанием другой, можно сделать заключение о непротиворечивости исчисления. Очевидно, что для сложных систем такой «предварительный» вывод нереален, и противоречивость исчисления можно определить только в процессе функционирования системы, реализованной на базе такого исчисления.

Исчисление предикатов имеют серьезные ограничения в представлении знаний человека.

Почти одновременно с силлогистикой была создана другая система — *логика стоиков*, ставшая прообразом современного исчисления высказываний. Вместе с тем в этой системе явно представлено только две формы мысли: суждение и умозаключение, тогда как связь понятий в суждении осталась за кадром. Несмотря на этот пробел логика стоиков отражает гораздо более сложные структуры суждений и умозаключений, чем те, что представлены модусами силлогистики. В этом причина того, что после открытия Дж. Булем пропозициональных функций результаты стоиков оказались более востребованы современной математической логикой, нежели результаты Аристотеля.

Обращение в современной логике к внутренней структуре элементарных суждений связано с рассмотрением предметно-пропозициональных функций, которые наборам предметов $<x_1, ..., x_n>$ ставят в соответствие специфические предметы «истина» и «ложь» (и «промежуточные» истинностные значения в случае многозначной логики). Эти функции были названы предикатами по аналогии с предикатом Аристотеля («сказуемое», «то, что высказывается о чем-то»). Однако такой предикат, как и входящие в него переменные, не является понятием — образ ситуации непосредственно связан с множеством предметов, минуя обобщенный образ предмета. Таким образом, можно сказать, что в современной математической логике категория «понятие» фактически отсутствует.

Частным случаем предикатов является *дескриптивная логика*, применяемая в системах на естественном языке. Она является основной моделью представления онтологических знаний, с которыми работают современные сервисы.

Дескриптивная логика. Цель создания дескриптивной логики в конце 1970-х гг. — описание семантики конструкций простейших моделей описания знаний, таких как фреймы и семантические сети.

Дескриптивная логика пошла по пути ограничения выразительных способностей и оценки вычислительной сложности, добавляемой с появлением каждой новой конструкции. Благодаря такой постановке задачи удалось создать множество семейств дескриптивных логик с различными вычислительными сложностями алгоритмов построения вывода. Дескриптивная логика разделяет все знания на две группы:

- интенсиональное знание, определяющее общие логические взаимосвязи в предметной области, иногда называемое схемой предметной области;
- экстенсиональное знание — знание, связанное с текущим состоянием системы и описывающее взаимосвязи между конкретными объектами и объектами, представляемыми семантической сетью.

Интенсиональное знание определяет некоторый язык понятий и поэтому в дескриптивной логике оно относится к terminology box (TBox). На языке, заданном экстенсиональным знанием, описывается состояние предметной области. Интенсиональное знание представляется в assertions box (ABox). При этом модель представления знаний дескриптивной логики работает с четырьмя типами элементов предметной области [16, 31].

1. Объект представляет собой конкретную сущность предметной области или значение свойства другого объекта.

2. Понятие обозначает конкретный класс объекта.

3. Роли обозначают бинарные отношения между объектами классов.

4. Признаки обозначают функции, заданные на некотором подмножестве объектов предметной области.

Дескриптивная логика определяет собственный синтаксис для представления выражений. Синтаксис конструкций TBox задается следующей грамматикой:

$$C := \text{Atom} \mid \top \mid \bot \mid \neg C \mid C \sqcup C \mid C \sqcap C \mid \forall\ R.C \mid \exists\ R.C \mid \exists\ R \mid$$
$$\geq n\ R.C \mid \leq n\ R.C \mid p{:}C \mid p \uparrow p \mid p \downarrow p \mid p \uparrow \mid \{\text{object}\}$$
$$p := \text{Feature} \mid p \circ p$$
$$R := \text{Role} \mid R \sqcap R \mid \text{R}^{-} \mid R \in \text{R}_{+} \mid R \circ \text{R} \mid$$
$$E := C \mid C \doteq C \mid C \sqsubseteq C \mid R \sqsubseteq R,$$

где C — нетерминал, обозначающий понятие; R — нетерминал, обозначающий роль; p — нетерминал, обозначающий признак; E — выражения дескриптивной логики; Atom, Role, Feature и object — произвольные слова, обозначающие атомное понятие, роль, признак и объект соответственно. Семантика этих выражений, определенная через интерпретацию I, универсум Δ, понятия C и D, роли P и R, признаки p и q, приведена в табл. 3.1.

Таблица 3.1. **Семантика основных конструкций TBox дескриптивной логики**

Имя конструкции	Синтаксис	Семантика	Классы ДЛ
Самое верхнее понятие	$\top$	$\top^{\text{I}} = \Delta$	SA
Самое нижнее понятие	$\bot$	$\bot^{\text{I}} = \varnothing$	SA
Отрицание	$\neg C$	$(\neg C)^{\text{I}} = \Delta \setminus C^{\text{I}}$	SC
Дизъюнкция	$C \sqcup D$	$C^{\text{I}} \cup D^{\text{I}}$	SUC
Конъюнкция	$C \sqcap C$	$C^{\text{I}} \cap D^{\text{I}}$	S
Ограничение значения	$\forall\ R.C$	$\{x \mid \forall y.(x,y) \in R^{\text{I}} \Rightarrow y \in C^{\text{I}}\}$	S
Ограничение существования	$\exists\ R.C$	$\{x \mid \exists y.(x,y) \in R^{\text{I}}\ \&\ y \in C^{\text{I}}\}$	SEC
Неограниченная квантификация существования	$\exists\ R$ ($\exists\ R.\top$)	$\{x \mid \exists y.(x,y) \in R^{\text{I}}\}$	S
Определенное числовое ограничение	$\geq n\ R.C$	$\{x \mid \#\{y \mid y \in C^{\text{I}} \& (x,y) \in R^{\text{I}}\} \geq n\}$	Q
	$\leq n\ R.C$	$\{x \mid \#\{y \mid y \in C^{\text{I}} \& (x,y) \in R^{\text{I}}\} \leq n\}$	Q
Числовое ограничение	$\geq n\ R$	$\{x \mid \#\{y \mid (x,y) \in R^{\text{I}}\} \geq n\}$	N
	$\leq n\ R$	$\{x \mid \#\{y \mid (x,y) \in R^{\text{I}}\} \leq n\}$	N
Выбор	$p{:}C$	$\{a \in dom(p^{\text{I}}) \mid p^{\text{I}}(a) \in C^{\text{I}}\}$	F

Окончание табл. 3.1

Имя конструкции	Синтаксис	Семантика	Классы ДЛ
Согласование	$p \uparrow p$	$\{a \in dom(p^{\mathrm{I}}) \cap dom(p^{\mathrm{I}}) \mid p^{\mathrm{I}}(a) = q^{\mathrm{I}}(a)\}$	F
Рассогласование	$p \downarrow p$	$\{a \in dom(p^{\mathrm{I}}) \cap dom(p^{\mathrm{I}}) \mid p^{\mathrm{I}}(a) \neq q^{\mathrm{I}}(a)\}$	F
Неопределенность	$p \uparrow$	$\Delta \setminus dom(p^{\mathrm{I}})$	F
Композиция признаков	$p \circ q$	$p^{\mathrm{I}} \circ q^{\mathrm{I}}$	F
Конъюнкция ролей	$P \sqcap \mathrm{R}$	$P^{\mathrm{I}} \cap R^{\mathrm{I}}$	R
Инверсия роли	R^{-}	$\{(x,y) \mid (y,x) \in R^{\mathrm{I}}\}$	I
Транзитивное замыкание примитивных ролей	$R \in \boldsymbol{R}_{+}$	$R^{\mathrm{I}} = (R^{\mathrm{I}})^{+}$	S
Композиция ролей	$R \circ R$		R
Полное определение	$C \doteq C$	$C^{\mathrm{I}} = D^{\mathrm{I}}$	
Указание объекта	$\{o\}$	$\{o^{\mathrm{I}}\}$	O
Частичное определение	$C \sqsubseteq D$	$C^{\mathrm{I}} \subseteq D^{\mathrm{I}}$	
Иерархия ролей	$R \sqsubseteq S$	$R^{\mathrm{I}} \subseteq S^{\mathrm{I}}$	H

Простейшая дескриптивная логика определяет язык L, используя следующие конструкции:

- ограничения значения;
- неограниченную квантификацию существования;
- конъюнкцию.

Эта дескриптивная логика получила название «язык фреймов» — сокращенно FL. Очевидно, что данный язык не имеет достаточно средств для определения семантики терминов. В язык AL были добавлены следующие конструкции:

- самое верхнее понятие;
- отрицание атомов;
- самое нижнее понятие.

Дальнейшее расширение этих базовых классов обозначается добавлением буквы, показывающей соответствующие конструкции.

Следует отметить, что добавление отрицания произвольных понятий, обозначаемого $\neg C$, позволяет получать дизъюнкцию и ограничение существования из базовых конструкций. В силу этого в класс логик ALC помимо отрицания были включены все конструкции логики ALUE. Логика ALC также имеет полиномиальный алгоритм вывода.

Важным расширением логики ALC является добавление транзитивных замыканий примитивных ролей. Такое расширение получило название S. Язык OWL, определяющий онтологии в семантическом Web, основан на дескриптивной логике SHOIQ. Следует отметить, что существует и логика ALCS, расширяющая логику ALC конструкциями для работы с элементами множеств. В частности, логика ALCS определяет следующие новые роли:

- роли $\in R(a, b)$, обозначающие, что свойство R, примененное ко всем элементам a, будет иметь значение b;
- роли $\in R(a, b)$ обозначающие, что все множества, содержащие элемент a, будут иметь значение роли R, равное b;
- роль $\in(a, b)$, обозначающая принадлежность элемента b множеству a.

Имеется множество других расширений дескриптивной логики, позволяющих представлять модальные, темпоральные и нечеткие конструкции, работать с n-местными предикатами, управлять семантикой рекурсивных (циклических) определений, и многие другие. Ограничение доступных выразительных возможностей ставит авторов баз знаний перед выбором: необходимо либо ограничивать описываемую семантику, либо расширять концептуализацию предметной области.

Например, определим хаб, как повторитель, имеющий не менее трех портов и передающий данные на все порты, кроме порта, с которого они поступили. Для записи этого определения в дескриптивной логике необходимо анализировать рассогласование признаков. Если мы используем логику SHIQ, то напрямую можно представить только следующее определение:

$$\mathtt{Hub} \sqsubseteq \mathtt{Repeater} \sqcap \geq \mathtt{hasPort}\ 3 \sqcap \forall\ \mathtt{transports.Frames}$$

Выражение определяет хаб как некоторый подкласс повторителей, имеющих не менее трех портов и позволяющих передавать только кадры. Для того чтобы определить отсутствие передачи обратно к отправителю, требуется рассматривать двойки «повторитель, кадр» и «порт, кадр», которые назовем FrameInRepeater, FrameInPort. При этом FrameInRepeater будет обладать ролями sendTo и receiveFrom, связывающими его с объектами типа FrameInPort:

$$\begin{aligned}
&\mathtt{FrameInRepeater} \doteq \exists\ \mathtt{forFrame.Frame} \sqcap \leq 1\ \mathtt{forFrame} \sqcap \\
&\quad \exists\ \mathtt{forRepeater.Repeater} \sqcap \leq 1\ \mathtt{forRepeater} \sqcap \\
&\quad \exists\ \mathtt{receiveFrom.FrameInPort} \sqcap \leq 1\ \mathtt{receiveFrom} \sqcap \\
&\quad \forall\ \mathtt{sendTo.FrameInPort} \\
&\mathtt{FrameInPort} \doteq \exists\ \mathtt{forFrame.Frame} \sqcap \leq 1\ \mathtt{forFrame} \sqcap \\
&\quad \exists\ \mathtt{forPort.Port} \sqcap \leq 1\ \mathtt{forPort} \sqcap
\end{aligned}$$

Тогда факт отсутствия передачи данных обратно к отправителю можно записать в форме

$\neg$(FrameInRepeater $\sqcap \exists$ receiveFrom.$\forall$ sendTo
(FrameInRepeater $\sqcap \exists$ forRepeater.Hub))/

Эта формула утверждает, что отсутствуют такие пары «повторитель, фрейм», которые получены из тех же портов, через которые хаб их отправил. Пара «порт, кадр» является всегда уникальной, и все порты в такой паре могут использоваться или для отправки фрейма, или для его получения. Таким образом, нехватку выразительных возможностей в большинстве случаев можно компенсировать усложнением концептуальной модели предметной области.

ABox представляет факты вида: $C(a)$, $R(a, b)$. Из-за их простоты дескриптивная логика не определяет единый синтаксис для представления таких утверждений. Например, допустима запись $a{:}C$ или $a \in C$.

Алгоритм построения вывода в дескриптивной логике основан на наборе правил преобразования выражений ABox и TBox. Перед применением правил все формулы преобразуются в исходную нормальную форму, которая зависит от используемого класса логик. Например, для логики ALC исходной считается форма следующего вида: $\{x{:}C\}$, где C — некоторое выражение, определяющее понятие. Цель вывода состоит в определении возможности существования такого x, которое будет принадлежать понятию C. При использовании ABox в процессе вывода могут быть определены конкретные значения для x. Например, правило преобразования для дизъюнкции принято записывать следующим образом:

$S \rightarrow \sqcup \{x{:}D\} \cup S$
если 1. $x{:}C1 \sqcup C2 \in S$
2. $x{:}C1 \notin S$ и $x{:}C2 \notin S$
3. $D{=}C1$ или $D{=}C2$

Применение этого правила к $S = \{x{:}C1 \sqcup C2\}$ даст два множества: $S1 = \{x{:}C1\}$ и $S2 = \{x{:}C2\}$. Далее вывод строится для множеств $S1$ и $S2$ независимо. Вывод закончится в двух случаях: из-за отсутствия применимых правил или из-за обнаружения противоречия во всех множествах Si, полученных из множества S. В первом случае будет получен положительный ответ, во втором — отрицательный. Возможные варианты противоречий зависят от семейства логик. В логике ALC они могут возникать только в одной из двух форм: $x{:}\perp$, $x{:}A$, $x{:}\neg A$.

Понятие «смысл» в традиционных интеллектуальных символьных системах развито весьма слабо. Способности человека создавать, использовать, интерпретировать осмысленные символы есть следствие его интеграции в изменяющуюся социальную среду. Современные же средства и методы создания интеллектуальных систем весьма далеки от способности кодировать и использовать эквивалентные по «смыслу» системы.

Прямым следствием бедной семантики является то, что методология поиска в традиционных интеллектуальных системах рассма-

тривает лишь предварительно интерпретированные состояния и контексты. Это означает, что создатель интеллектуальной системы связывает с используемыми символами семантический смысл. Такие интеллектуальные системы могут строить лишь некую вычисляемую функцию в этой интерпретации. Вследствие этого бóльшая часть интеллектуальных систем сильно ограничена в возможности построения новых смысловых ассоциаций по мере изучения изменяющегося окружающего мира. Из-за таких ограничений наиболее значительные успехи связаны с разработкой интеллектуальных приложений, в которых можно абстрагироваться от чересчур широкого контекста и описать основные компоненты решения задачи с помощью заранее интерпретируемых символьных систем.

Можно констатировать, что используемые в настоящее время методы позволяют реализовать прикладные системы, относящиеся к категории «Автоматические и автоматизированные системы со статической предметной областью». Для адаптивных систем указанные выше ограничения могут стать существенным тормозом в их дальнейшем развитии.

Расширить рамки практического применения символьных систем смогут принципиально новые подходы к методам представления и использования знаний.

При создании систем искусственного интеллекта (СИИ) одной из основных задач является обеспечение оптимального соответствия методов функционирования системы формам человеческой мысли, т.е. логике естественного интеллекта (ЕИ). Именно при этом условии результаты функционирования СИИ могут быть понятны человеку и находиться под его контролем.

Проведенные в течение длительного периода времени фундаментальные и прикладные исследования в области логических систем нового типа (модальная, ситуативная логика) позволили сформулировать основные положения по построению функциональной модели понятия [7, 72].

Такая модель названа концептоидом и представляет собой частный случай «почти экстенсионально свободной функции» (ПЭС-функции). Свойства этого рода функций и их связь с логикой были впервые изложены в работе [8].

3.3. Логика концептоидов как формальная модель

Концептоид — это особого рода функция, свойства которой отражают существенные свойства понятия как формы мысли, и в которой обобщаются и выделяются предметы некоторого класса по определенным общим и в совокупности специфическим для них признакам.

Одной из существенных особенностей является возможность свести сетевое представление базы знаний (БЗ) к системе функциональных (концептоидных) уравнений, решаемых методом итераций. Сходимость итерационного процесса означает непротиворечивость требований (постулатов, аксиом) к свойствам объектов, описываемых в БЗ. До сих пор непротиворечивость никак не устанавливалась и вопрос о надежности работы СИИ решался чисто статистически на основе опыта наблюдений над результатами функционирования.

Элементарными формулами логики концептоидов являются равенства и неравенства термов. Их множества можно приводить к системам функциональных уравнений, решаемых методом итераций. Это эффективный способ получения явных определений понятий на основе заданных связей между ними (например, системой аксиом). Такого способа, которым можно исследовать конкретную систему понятий на непротиворечивость, нет в пропозициональных логических системах.

Логика концептоидов — весьма гибкая система, способна погрузить в себя, видимо, все другие логические системы. Имеется опыт погружения классического и конструктивного исчислений предикатов, модального исчисления S4 Льюиса, паранепротиворечивой логики.

Реализация на основе предложенной логики систем искусственного интеллекта нового поколения может существенно расширить область применения интеллектуальных методов для решения сложных задач в неопределенных и динамически меняющихся средах. Она создает универсальное средство для реализации как новых СИИ, так и разработанных ранее на базе других логик путем реконструкции любой логической системы на базе ПЭС-функций.

Для иллюстрации универсальности полезно показать связь ПЭС-функций с логикой. Такую связь можно рассматривать в двух ракурсах:

- дать общее описание свойств ПЭС-функций и отношений между ними средствами формального логического исчисления (аксиоматической теории), исследовав такие дедуктивные свойства этого исчисления, как непротиворечивость, полнота, независимость аксиом;
- интерпретировать известные логические исчисления как некоторые системы ПЭС-функций, представляя логические объекты (предикаты, связки, кванторы и пр.) в виде ПЭС-функций и операций над ними.

В качестве реализации первого подхода рассмотрим формальное исчисление B1, являющееся одной из возможных аксиоматизаций общей теории ПЭС-функций.

Алфавит: a, b, ..., z, ..., a_n, b_n, ..., z_n, ... ($n \geq 0$) — переменные.

Служебные знаки: $\boldsymbol{B}$ — функтор; = — предикатор (имя отношения тождественного равенства).

Термы:

любая переменная есть терм;

если α, β и γ — термы, то $\boldsymbol{B}\alpha\beta\gamma$ — терм;

выражение есть терм тогда и только тогда, когда оно построено согласно пп. (i) — (ii).

Формулы:

если α и β — термы, то $\alpha = \beta$ — формула. Другого вида формул нет.

Определение: $\boldsymbol{I}xyuw =_{Df} \boldsymbol{BB}xvyw\boldsymbol{B}xwy$.

Аксиомы:

A1: $\boldsymbol{B}xxy = x$; A2: $\boldsymbol{B}xyx = y$; A3: $\boldsymbol{B}xyy = x$;

A4: $\boldsymbol{BI}xya_1b_1\boldsymbol{I}xya_2b_2\boldsymbol{I}xya_3b_3 = \boldsymbol{I}xy\boldsymbol{B}a_1a_2a_3\boldsymbol{B}b_1b_2b_3$.

Правила вывода:

ПР (правило равенства): $\dfrac{\Gamma = \Delta,\ \Gamma = \Sigma}{\Delta = \Sigma}$;

ПП (правило подстановки): $\dfrac{\Gamma = \Delta}{\Gamma[\alpha/\Sigma] = \Delta[\alpha/\Sigma]}$;

ПЗ (правило замены): $\dfrac{\Gamma = \Delta}{\Sigma[\alpha/\Gamma] = \Sigma[\alpha/\Delta]}$.

(Здесь Γ, Δ и Σ – термы, α — переменная, а выражение вида $\Gamma[\alpha/\Sigma]$ (и аналогичные) означает результат подстановки терма Σ на все места вхождений переменной α в терм Γ. Если в Γ нет ни одного вхождения α, то $\Gamma[\alpha/\Sigma]$ есть Γ.) Понятия вывода формулы из посылок и доказательства формулы определяются, как обычно для исчислений гильбертова типа, к которому B1 относится.

Для исчисления B1 доказаны его непротиворечивость (имеется недоказуемая формула $x = y$), независимость аксиом и неполнота. Смысл неполноты состоит в том, что объем множества U всех возможных значений переменных может оказаться меньше, чем число аргументов ПЭС-функции. Тогда не все отношения эквивалентности на множестве аргументов могут быть реализованы, и именно те отношения, что определяют различие функций, могут оказаться в числе нереализуемых. Вместе с тем исчисление B1 обладает полнотой относительно интерпретации. Это означает следующее. Если в табличном задании ПЭС-функций F и G даны в виде

$x_1x_2 \ldots x_n$	F	G
...	...	...

столбцы F и G полностью совпадают, то в B1 доказуема формула $F = G$. Если же в какой-то строке (где представлено больше двух клас-

сов эквивалентности на множестве переменных) имеется несовпадение, то эта формула недоказуема и фактически имеем $F \neq G$. Однако формулу $F = G$ можно постулировать, т. е. добавить к числу аксиом, не вызывая противоречия. Вот такая логическая тонкость имеет место, и ее нужно учитывать во всех приложениях теории ПЭС-функций.

В работе [7] доказано, что исчисление B1 обладает свойством *разрешимости*. Это означает наличие чисто формальной процедуры, позволяющей установить доказуемость или недоказуемость той или иной формулы, не обращаясь непосредственно к интерпретации, а только на основе графического сравнения символьных выражений, получаемых в ходе дедуктивных преобразований. Смысловая ориентация такой разрешающей процедуры на интерпретацию остается «за кадром», но она важна для общего понимания теории и оценки возможностей ее применения. Разрешающая процедура основана на приводимости любого терма к так называемой нормальной **I**-форме, которая находится во взаимно-однозначном соответствии с табличным заданием ПЭС-функции, обозначаемой данным термом.

Заметим, что исчисление B1 не использует типичные для современных логических исчислений пропозициональные связки, кванторы и подобные им операторы. Весь этот арсенал находится в *метатеории*, описывающей дедуктивные свойства данной «предметной» теории. В самой же теории единственным оператором, позволяющим образовать из термов формулу (предикат), является знак =. Даже для выражения неравенства функций ($F \neq G$) служит *косвенный вывод* (приведение к противоречию), а именно: $F = G \Rightarrow x = y$. Однако существует возможность представления типичных для логики операций в виде ПЭС-функций, исполняющих специальную роль. Здесь мы уже обращаемся ко второму аспекту связи ПЭС-функций с логикой.

Добавим в алфавит исчисления B1 две переменные специального назначения: τ и θ. Полный список переменных будем рассматривать как множество истинностных значений пропозициональных функций в некоторой многозначной логике с переменным числом значений, причем τ и θ интерпретируются как «выделенные» значения, т. е. **Истина** и **Ложь** соответственно. Введем также в язык бесконечный список синтаксических переменных: A, ..., Z, A_0, ..., Z_0, ..., A_n, ..., Z_n, ... Эти переменные будем называть *пропозициональными*, а область их значений составляют все термы, которые можно построить в алфавите B1 (включая новые переменные τ и θ). Те же переменные, которые обозначаются строчными буквами, будем называть *предметными*. При этих соглашениях дадим следующие определения пропозициональных операций (логических связок).

Df ⌉: $\rceil F =_{Df} B\theta\tau F$ (Сильное отрицание — утверждение «F есть **Ложь**»);

Df \: $\backslash F =_{Df} B\tau\theta F$ (Слабое отрицание — утверждение «F не есть **Истина**»);

Df &: $F\&G = {}_{Df}\mathrm{I}F\theta\theta\mathrm{I}G\theta\theta\tau$ (Конъюнкция);
Df ∨: $F\vee G = {}_{Df}\mathrm{I}F\theta\mathrm{I}G\theta\theta\tau\tau$ (Дизъюнкция);
Df ⊃: $F\supset G = {}_{Df}\mathrm{I}F\theta\tau\mathrm{I}G\theta\theta\tau$ (Импликация).

Какое-либо выражение Ф, построенное из пропозициональных переменных с применением приведенных операций, представляет «высказывание» (или «суждение»). Высказывание считается «установимым», если можно доказать равенство Ф = τ, либо формальными средствами исчисления B1, либо табличным методом. Перебор возможных отношений эквивалентности между пропозициональными переменными в принципе ничем не отличается от аналогичных сравнений предметных переменных. В такого рода доказательствах удобнее всего непосредственно обращаться к смыслу отношений «виртуального равенства» и «виртуального различия» ($\simeq$ и $\not\simeq$), варьируя возможные случаи и их распределения между пропозициональными переменными. Если указанными методами можно доказать равенство Ф = θ, то высказывание Ф считается «опровержимым». Терминами «неустановимость» и «неопровержимость» будем обозначать *недоказуемость* указанных равенств, используя (неформально, поскольку в языке B1 нет символа ≠) выражения Ф ≠ τ и Ф ≠ θ. В качестве метатеоремы можно доказать следующую связь между установимостями высказываний, известную в логике как *Modus Ponens* и используемую как основное правило логического вывода:

$$\textbf{MP:}\quad \frac{\neg\neg F,\quad F\supset G}{\neg\neg G}\ .$$

С точки зрения интерпретации это правило означает, что если $F \neq \theta$ и $(F\supset G) = \tau$, то $G \neq \theta$.

Можно доказать, что

$$\neg F = \backslash(\backslash F\supset F) = F\supset\backslash(F\supset F).$$

и сильное отрицание можно выразить через слабое. Обратное представление несправедливо. Также представляют интерес следующие доказуемые равенства:

$$\neg\neg F = \backslash\neg F = (\backslash F\supset F)$$
$$\backslash\backslash F = \neg\backslash F.$$

Доказательство этих равенств представлено в табл. 3.2.

Табличным методом можно доказать, что имеет место установимость следующих суждений группы 1 (Гр. 1) и неустановимость суждений группы 2 (Гр. 2):

Гр. 1) $(F\supset G)\supset(\backslash F\vee G)$; $(F\supset\backslash F)\supset\backslash F$; $(\backslash F\supset F)\supset F$; $F\vee\backslash F$;
$(F\supset G)\vee(F\supset\backslash G)$; $\backslash\backslash F\supset F$; $F\vee\backslash(F\&\backslash F)$;

Таблица 3.2. **Доказательство равенств**

$\theta\tau F$	$\neg F$	$\setminus F$	$\neg\neg F$	$\neg\setminus F$	$\setminus\neg F$	$\setminus\setminus F$	$F\supset F$	$\setminus(F\supset F)$	$\setminus F\supset F$	$\setminus(\setminus F\supset F)$	$F\supset\setminus (F\supset F)$
0 0 0	0	0	0	0	0	0	0	0	0	0	0
0 0 1	0	0	0	0	0	0	0	0	0	0	0
0 1 0	1	1	0	0	0	0	1	0	0	1	1
0 1 1	0	0	1	1	1	1	1	0	1	0	0
0 1 2	0	1	1	0	1	0	1	0	1	0	0

Гр. 2) $\setminus(F\&\setminus F)$; $\setminus F\supset(F\supset G)$; $\setminus F\supset(F\supset\setminus G)$; $(F\&F)\supset G$; $F\supset(\setminus F\supset G)$; $F\supset(\setminus F\supset\setminus G)$; $(F\supset G)\supset((F\supset\setminus G)\supset\setminus F)$; $F\supset\setminus\setminus F$; $(\setminus F\supset\setminus G)F\supset((\setminus G\supset F)\supset G)$.

Приведенное распределение установимых и неустановимых высказываний является типичным для *паранепротиворечивой* пропозициональной логики, формализуемой в сигнатуре $<\setminus, \supset, \&, \vee>$. Если же при построении высказываний не пользоваться слабым отрицанием ($\setminus$) и ограничиться сигнатурой $<\neg, \supset, \&, \vee>$, то мы будем иметь дело с *классической* пропозициональной логикой. Это доказывается установимостью следующих трех высказываний, которые Я. Лукасевич предложил в качестве системы аксиом классического исчисления высказываний:

CL1: $F\supset(G\supset F)$;

CL2: $(F\supset(G\supset H))\supset((F\supset G)\supset(P\supset H))$;

CL3: $(\neg G\supset\neg F)\supset((\neg G\supset F)\supset G)$.

Те же самые выводы относительно использования двух видов операции отрицания делает А. С. Карпенко [23], когда рассматривает паранепротиворечивую логику как структуру внутри обычной многозначной логики, где истинностные значения являются константами, а не переменными.

Другой вариант представления многозначной пропозициональной логики через ПЭС-функции можно получить, введя вместо слабого отрицания похожую, но бинарную операцию:

Df $\setminus^2$**:** $F\setminus G =_{Df} \mathrm{B}F\theta G$ (Разность).

Очевидно, что прежнее слабое отрицание можно выразить через разность в виде $(F\supset F)\setminus F$. В то же время особый интерес представляет возможность выразить формально отношение виртуального равенства $F \simeq G$, используемое в интерпретации:

Df $\simeq$**:** $F\simeq G =_{Df} \neg(F\setminus G)\ \&\neg(G\setminus F)$ $(=\mathrm{I}FG\tau\theta)$ (Виртуальное равенство);

Df $\not\simeq$**:** $F\not\simeq G =_{Df} \neg(F\simeq G)$ $(=\mathrm{I}FG\theta\tau)$ (Виртуальное различие).

Выражения $F \simeq G$ и $F \not\simeq G$ являются *предикатами*, если значениями синтаксических переменных F и G являются термы исчисления B1, не содержащие переменных τ и θ. Но поскольку мы рассматриваем *все* предметные переменные как истинностные значения функций некоторой многозначной логики, то тем самым фактически стирается грань между «исчислением высказываний» и «исчислением предикатов». Остается только указать способ представления любого предиката в той системе логических операций, которая представима через ПЭС-функции. Эту задачу решает «логика концептоидов», которую рассмотрим ниже.

Рассмотренная связь логики с теорией ПЭС-функций позволяет строить «в едином ключе» функциональные модели как суждений, так и понятий. Кроме того, моделируемыми оказываются и те формы мысли, которым традиционная логика не уделяла специального внимания. Например, *единичные* «идеальные предметы» типа отдельных геометрических фигур или отдельных чисел не являются понятиями, поскольку, принадлежа к классу объектов, они не определяют класс полностью.

Другой важнейшей формой мысли является вопрос. Математика нашла для вопроса такую форму выражения, как уравнение или система уравнений, где переменная играет роль неизвестного компонента. Но составлять и решать логические уравнения мы пока не умеем, хотя уже давно есть работы в области «логики вопросов» (или «интеррогативной логики» [28, 63]). Применение ПЭС-функций позволяет устранить эти пробелы.

Для названных ранее τ и θ переменными специального назначения не оговаривалась специфика их применения. Как аргументы конкретной ПЭС-функции они ничем не отличаются от прочих переменных. Только при рассмотрении множества ПЭС-функций как системы объектов, построенных из некоторых первичных элементов, эта специфика себя как-то проявляет. Для более точной ориентации в этом вопросе нужно рассмотреть некоторые общие свойства переменных, определяющих специфику их участия в отношении, называемом функцией, причем это касается не только ПЭС-функций.

Определение 1. Переменная x называется *нейтральным аргументом* функции F, заданной на множестве переменных V, если при каждом распределении значений переменных, входящих в $V\backslash\{x\}$, либо значение функции фиксировано при изменении значений переменной x, либо функция принимает значение самой переменной x. Если всегда имеет место лишь первый случай, то x называется *фиктивным аргументом* функции F. Аргумент функции, не являющийся фиктивным, называется *явным*.

Функция называется *m-местной*, если она имеет ровно m явных аргументов.

Согласно этому определению ПЭС-функция Bxyz действительно является трехместной, причем переменная y — ее нейтральный яв-

ный аргумент. Функция I*xyvw* — четырехместная, с нейтральными аргументами *v* и *w*. У функции A*xyz* нейтральным явным аргументом является *x*, а функции C*xyz*, D*xyz* и E*xyz* не имеют нейтральных явных аргументов. Кроме того, каждую отдельную переменную можно рассматривать как одноместную ПЭС-функцию, тогда как двуместных ПЭС-функций вообще не существует.

Для дальнейших построений существенное значение имеет следующая метатеорема:

1. $\Phi = \Psi \Rightarrow \boldsymbol{I}\Phi\Psi\alpha\beta = \alpha$, где Φ и Ψ — любые ПЭС-функции, α и β — любые переменные.

2. $\boldsymbol{I}\Phi\Psi\alpha\beta = \alpha \Rightarrow \Phi = \Psi$, где Φ и Ψ — любые ПЭС-функции, но хотя бы одна из переменных α или β является нейтральной (в частности, фиктивной) для функций Φ и Ψ.

Общее понятие ПЭС-функции допускает рассматривать ПЭС-функции второго порядка, например:

$$I'XYVW = \begin{cases} \overset{V}{(X = Y)} \\ W \\ (X \neq Y) \end{cases} = \begin{cases} \forall\varepsilon(\overset{V}{X \simeq Y}) \\ W \\ \exists\varepsilon(X = Y), \end{cases}$$

где *X*, *Y*, *V*, *W* — синтаксические переменные, значениями которых могут быть любые ПЭС-функции (первого порядка), а также отдельные переменные; ε — отношение эквивалентности на множестве переменных (также первого порядка), образуемом явными аргументами ПЭС-функций *X*, *Y*, *V*, *W*. Аналогично, ПЭС-функцией второго порядка является B'*XYZ* = I'*XZYX*. Нетрудно доказать следующее обобщение аксиомы A4 исчисления B1:

$$\mathrm{BI}'XYA_1B_1\mathrm{I}'XYA_2B_2\mathrm{I}'XYA_3B_3 = \mathrm{I}'XY\boldsymbol{B}A_1A_2A_3\mathrm{B}B_1B_2B_3.$$

Кроме того, очевидно, что, какова бы ни была конкретная ПЭС-функция Φ, применение операции **B**'τθΦ дает ПЭС-функцию с *нейтральными* аргументами τ и θ.

Определение 2. Любую ПЭС-функцию, заданную на списке переменных: τ, θ, *a*, *b*, ..., *z*, ..., a_n, b_n, ..., z_n, ...— будем называть *концептоидом*, если она имеет τ в качестве нейтрального аргумента. Если при этом τ — явный аргумент, то ПЭС-функция называется τ-*концептоидом*; если τ — фиктивный аргумент, то — θ-*концептоидом*.

Если функция, являющаяся τ-концептоидом, всегда принимает значения либо переменной τ, либо переменной θ, то данный концептоид будем называть *суждением*.

Если функция, являющаяся θ-концептоидом, всегда принимает значения либо некоторой переменной *x*, либо переменной θ, то данный концептоид будем называть *понятием*. При этом переменную *x* будем называть *номиналом* данного понятия.

Вводимые данным определением термины будем применять также и к термам формального языка, описывающего ПЭС-функции. Следует учесть условность такого использования терминов, поскольку обычно «понятиями» и «суждениями» называют собственно мысли, а не их материализуемые модели, языковые или функциональные.

Используем для обозначения в символьных выражениях общего характера, когда речь идет о любых концептоидах, греческие литеры: A, ..., Ω, A_0, ..., Ω_0, ..., A_n, ..., Ω_n, ... Если же речь идет именно о θ-*концептоидах*, то будем применять латинские литеры: A, ..., Z, A_0, ..., Z_0, ..., A_n, ..., Z_n, ... При этом будем избегать литер, имеющих одинаковый вид в обоих алфавитах, когда нужно отразить различие между τ-концептоидом и θ-концептоидом.

Поскольку в логике концептоидов необходимо по определению ввести знаки = и ≠ в качестве функторов в составе термов, то будет целесообразно заменить знак = в его прежнем смысле (предикатор в исчислении B1) на знак ≡, чтобы не путать формальные выражения с интерпретацией.

Строить логику концептоидов будем в сигнатуре <¬, \, ~, ∇, ⌊⌋>. Стилизованные кавычки ⌊⌋ нужно понимать как знак одной логической операции.

Интерпретация сигнатуры:

Df ¬: $\neg\Phi \equiv_{Df} B\theta\tau\Phi$	(Отрицание);
Df \: $F\backslash G \equiv_{Df} BF\theta G$	(Разность);
Df ~: $\sim\Phi \equiv_{Df} B'\tau\theta\,\Phi$	(Опровержение);
Df ∇: $\nabla\Phi \equiv_{Df} B\Phi\tau\theta$	(Деноминация);
Df ⌊⌋: $\Phi\lfloor\Psi\rfloor \equiv_{Df} \Psi[\tau/\Phi]$	(Концептуализация).

Как и выше для исчисления B1, метавыражение вида $\Psi[\tau/\Phi]$ означает результат замены аргумента τ функции Ψ функцией Φ.

Правила построения термов формального языка логики концептоидов зададим таким образом, чтобы они гарантировали нейтральность аргумента τ соответствующих ПЭС-функций.

Термы (*концептоиды*)*:*

любая предметная переменная (кроме τ и θ) есть θ-концептоид;

если Φ и Ψ есть θ-концептоиды, то Φ\Ψ есть θ-концептоид;

если Φ есть θ-концептоид, то ¬Φ, ∇Φ и ~Φ есть τ-концептоиды;

если Φ есть τ-концептоид, то ¬Φ, ∇Φ и ~Φ есть τ-концептоиды;

если Φ есть θ-концептоид, а Ψ есть τ-концептоид, то Φ⌊Ψ⌋ есть θ-концептоид;

если Φ и Ψ есть τ-концептоиды, то Φ⌊Ψ⌋ есть τ-концептоид;

каждый терм есть θ-концептоид или τ-концептоид только в силу пп. (i) — (vii).

Определения логических операций:

Df ⊃: $\Phi\supset\Psi \equiv_{Df} \neg(\Phi\lfloor\neg\Psi\rfloor)$ (≡ΠΨθΦθθτθ) (Импликация);

Df ∨: $\Phi\vee\Psi \equiv_{Df} \neg((\neg\Phi)\lfloor\neg\Psi\rfloor)$ (≡ΠΠΨθBθτΦθθτθ) (Дизъюнкция);

Df &: $\Phi\,\&\,\Psi \equiv_{Df} \neg\lfloor\neg\Phi\rfloor\neg\Psi\rfloor)$ (≡ΠΨθθΦθθτ) (Конъюнкция);

Df ↔: $\Phi \leftrightarrow \Psi \equiv_{Df} (\Phi \supset \Psi)\&(\Psi \supset \Phi)$ (≡IBθτΦBθτΨτθ) (Логическая эквивалентность);

Df ∩: $F \cap G \equiv_{Df} F\backslash(F\backslash G)$ (≡I*FGF*θ) (Пересечение);

Df ⊂: $F \subset G \equiv_{Df} \neg(F\backslash G)$ (≡I*FG*τI*F*θτθ) (Включение — «*F* есть *G*»);

Df ≃: $F \simeq G \equiv_{Dff} (F \subset G)\&(G \subset F)$ (≡I*FG*τθ) (Виртуальное равенство);

Df ≠: $F \neq G \equiv_{Df} (F \simeq G)$ (≡I*FG*θτ) (Виртуальное различие);

Df I: $IXYVW \equiv_{Df} (W\lfloor X \neq Y\rfloor)\lfloor\nabla(V\lfloor X \simeq Y\rfloor)\rceil$ (≡I*XYVW*);

Df B: $BXYZ \equiv_{Df} IXZYX$ (≡B*XYZ*);

Df θ: $\theta \equiv_{Df} F\backslash F$ (≡θ) (0-местная предметная операция);

Df τ: $\tau \equiv_{Df} \lceil\theta$ (≡τ) (0-местная пропозициональная операция);

Df □: $\Box\Phi \equiv_{Df} \lceil\sim\Phi$ (≡BθτB'τθΦ) (Необходимость);

Df ◊: $\Diamond\Phi \equiv_{Df} \sim\lceil\Phi$ (≡B'τθBθτΦ) (Возможность);

Df =: $F{=}G \equiv_{Df} \Box(F \simeq G)$ (≡I'*FG*τθ) (Строгое равенство);

Df ≠: $F{\neq}G \equiv_{Df} \Diamond(F \neq G)$ (≡I'*FG*θτ) (Строгое различие).

Характеристики концептоидов приведены в табл. 3.3.

Таблица 3.3. **Сравнительные характеристики символьных систем**

Логическая система	Достоинства	Недостатки
Логика концептоидов	Описание предметной области и процедур логического вывода в терминах понятий. Формализация процедур проверки непротиворечивости баз знаний. Возможность реконструкции любой логической системы на базе ПЭС-функций. Реализация аппаратных средств СИИ на базе решателя с нетрадиционной процессорной архитектурой. Может использоваться для реализации как систем искусственного интеллекта, так и искусственного сознания	Часть теоретических вопросов находится на стадии прикладных научно-технических исследований. Практическое применение ограничено программными прототипами нескольких интеллектуальных систем. Технология проектирования систем искусственного интеллекта на базе логики концептоидов находится на стадии прикладных научно-технических исследований. Технология создания аппаратных средств реализации систем искусственного сознания находится на уровне фундаментальных научно-технических исследований

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Назовите методы математического описания процессов в интеллектуальной системе.
2. Что такое онтология? Какова сфера ее применения?
3. Что такое концептуализация?
4. Назовите уровни Web-технологий.
5. Назовите и дайте характеристики подходов к формированию определений.
6. Что такое дескриптивная логика?
7. Что представляет собой теория формальных символьных систем?
8. Укажите важнейшие методологические принципы научных исследований интеллектуальной деятельности.
9. Что является знаком в интеллектуальной системе?
10. В чем отличие синтаксиса языка от семантики?
11. Дайте определение формальной системы.
12. Определите логическое исчисление.
13. Дайте сравнительную характеристику логики высказываний и логики предикатов.
14. Укажите основные формализмы представления предикатами.
15. Что представляет собой вывод в логике предикатов?
16. Укажите основные типы логических систем.
17. Дайте определение концептоида.
18. Какие существуют ограничения символьных систем?
19. Какова связь между ПЭС-функциями и логикой?
20. В чем суть логики концептоидов как формальной системы?
21. Как проявляется связь логики с теорией ПЭС-функций?
22. В чем отличие τ-концептоида от θ-концептоида?
23. Каковы основные компоненты модели представления знаний дескриптивной логики?
24. Охарактеризуйте семантику основных конструкций TBox дескриптивной логики.
25. Какие существуют системы, позволяющие работать с дескриптивной логикой?
26. Дайте сравнительную характеристику символьных систем.

Глава 4

ТЕХНОЛОГИЯ ПРОЕКТИРОВАНИЯ ЭКСПЕРТНЫХ СИСТЕМ

4.1. Математическое описание знаний

Системно описать блоки — базы данных (БД), базы правил (БП), машины логического вывода (МЛВ), блока объяснений, интерфейса пользователя (ИП) — можно, используя два варианта:

- применять универсальный математический аппарат для всех блоков;
- описывать блоки разными математическими методами, стыкующимися между собой.

В силу большого разнообразия природы блоков описание универсальным методом не представляется возможным.

Очевидно, что БП и блок объяснений должны описываться одним математическим аппаратом. Объяснения могут иметь разную форму: КАК получено решение (на основе какай системы правил), КАК звучит правило с определенным номером, в КАКИХ правилах используется тот или иной параметр, ПОЧЕМУ компьютер запрашивает данные для уточнения запроса.

Работа интерфейса предполагает введение запроса, получение ответа системы, получение объяснения и напоминает работу БД.

Таким образом, необходимы три вида математического описания: БД, БП, МЛВ.

К методам описания предъявляются следующие требования:

- однозначность описания;
- оперирование множествами и подмножествами;
- интеграция методов описания данных, описания знаний и методов логического вывода;
- простота получения решения;
- возможность описания непрерывных процессов;
- доходчивость, наглядность и простота понимания правил человеком.

В общем случае системная работа перечисленных блоков в части вывода результатов Y может быть описана в виде

$$Y = < X,\ P,\ B,\ R >, \qquad (4.1)$$

где X — множество базовых элементов (фактов); P — правила с условиями их применения; B — множество аксиом построения синтак-

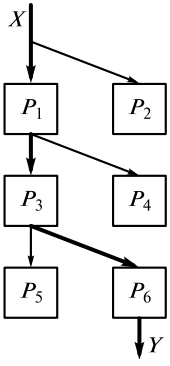


Рис. 4.1. Процесс вывода в блоке МЛВ (жирными линиями показан окончательный путь вывода)

сически правильных совокупностей; R — множество правил вывода. Результат Y в теории экспертных систем называют целью.

Совокупность $<X, P>$ образует аппарат описания знаний в блоках БД и базе правил, совокупность $<B, R>$ — закономерности вывода в блоке МЛВ.

Объяснение O в общем виде может быть описано

$$O = <P_i,\ i = 1,\ N>, \qquad (4.2)$$

где P_i — правила, участвовавшие при выводе данного результата. Они (рис. 4.1) запоминаются в процессе вывода (путь вывода помечен жирными линиями).

Для базы данных следует использовать реляционную алгебру, для базы правил и объяснений — методы описания знаний, для МЛВ — методы логического вывода.

Связь данных и правил можно представить в виде рис. 4.2.

Одноуровневая система знаний может быть представлена в виде табл. 4.1.

При иерархической (многотабличной) системе правил некоторые цели могут стать ключами подчиненных таблиц.

Сложнее обстоит дело с сочетанием $<X, P>$ и $<B, R>$, поскольку методы описания знаний и логического вывода первоначально развивались автономно. В связи с этим одной из задач экспертных систем является определение такого набора методов описания блоков, которые математически интегрировались бы в общую систему. Иными словами, требуется создание системной модели. Интегральное описание должно характеризоваться наглядностью описания; простотой и однозначностью математического описания; эффективностью методов; возможностью формирования объяснений; возможностью интеграции методов.

Для того чтобы оценить особенности согласования методов описания знаний и методов логического вывода, рассмотрим сами методы.

Таблица 4.1. **Одноуровневая система правил**

Данные			Цель
A	*B*	*C*	*G*
*A*1	*B*1	*C*1	*G*1
*A*2	*B*2	*C*2	*G*2

Примечание. *A*, *B*, *C*, *G* — имена; 1, 2 — значения данных.

Рис. 4.2. Представление знаний

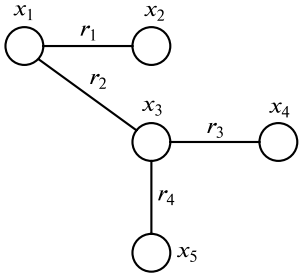


Рис. 4.3. Семантическая сеть

Начнем с методов описания знаний. С этой целью используются такие методы, как семантические сети, фреймы, продукции (правила).

Семантические сети. Иллюстрируем метод на примере выражения: «Студент пришел в аудиторию и слушал лекцию, которую читал преподаватель. Лекция была посвящена ЭС».

Введем обозначения: x_i — элементы (существительные); r_j — связки (глаголы).

Студент — x_1; аудитория — x_2; лекция — x_3; преподаватель — x_4; ЭС — x_5; пришел — r_1; слушал — r_2; читал — r_3; была посвящена — r_4.

На основе принятых обозначений можно дать графическую трактовку предложения (рис. 4.3). Данный способ хорош при оперировании с ЕЯ, однако плохо сочетается с теоретическим аппаратом описания МЛВ.

Фреймы. Имеются различные градации фреймов [17, 56, 57]. Воспользуемся одной из них. Выделим фреймы-описания (элементы) и ролевые фреймы (связки).

Фрейм может иметь иерархическую структуру: $F = [<N_1,V_1>,<N_2,V_2>, ...,<N_n,V_n>]$. Каждая такая пара $<N_m,V_n>$ называется слотом; N_n — название слота; V_n — значение слота. Возможны ссылки на другие фреймы. Приведем примеры.

Фрейм-описание

```
[<машины>,<станки токарные, 70>, <станки фрезерные,
20>].
```

Ролевой фрейм

```
[<отгрузка>,<станки токарные, 70>, <куда, Москва>,
<откуда, СПб>,
<вид транспорта, железная дорога>].
```

Имеются также понятия «фрейм-образец» (прототип, без введенных данных), называемый интенсиональным описанием, «фрейм-

экземпляр» с соответствующими конкретными данными и «экстенсиональное описание».

В другой градации [56] выделяют фреймы-структуры (объекты, понятия), фреймы-роли (вариант поведения объекта), фреймы-сценарии (взаимодействие связанных процессов) и фреймы-ситуации (влияние внешней среды).

Все процедуры фреймов делятся на две группы: «демоны» и присоединенные процедуры. «Демоны» действуют автоматически при удалении, запросе, изменении данных, присоединенные процедуры действуют лишь после выполнения условий.

В силу необычности описания и сложности языка программирования метод получил ограниченное применение. К тому же он плохо сочетается с методами МЛВ.

Продукции (правила). Здесь используется схема ЕСЛИ ..., ТО... Левую часть (условия) называют *антецедентом*, а правую (результат) — *консеквентом*. Описание достаточно наглядно, хорошо сочетается с методами логического вывода, в частности с предикатами первого порядка.

Сравнительные характеристики методов описания знаний представлены в табл. 4.2.

Таблица 4.2. **Сравнительные характеристики метода описания знаний**

Метод	Достоинства	Недостатки
Семантические сети	Возможность представления смысла фраз и ограничений. Наличие внутренней структуры. Определение операций над объектами (естественный язык)	Нет средств зависимости от времени. Произвольность структуры. Сложность сочетания с методами логики
Фреймы	Наличие внутренней структуры и связности	Жесткость структуры. Трудности построения модулей и модификации. Сложность сочетания с методами логики
Продукции	Модульность. Простота модификации правил. Отделение предметных правил от управления. Простота сочетания с методами логики	Отсутствие внутренней структуры. Зависимость шагов вывода от принятой стратегии вывода

Из приведенных структур методов описания знаний нетрудно заметить, что построить, позволяющую наглядно увидеть правила и обнаружить ошибки в их формировании, удобнее всего для продукций. К тому же продукции легко трансформируются при изменении правил. Перечисленные факторы определили более широкое использование продукций.

Описание знаний касалось блока базы правил. Перейдем к описанию процессов в МЛВ, т. е. к теории логического вывода.

4.2. Методы логического вывода

Для описания логического вывода применяют следующие методы: традиционная логика; булева алгебра (алгебра логики); исчисление предикатов.

Традиционная логика. Возникла еще во времена Аристотеля и была предназначена для уменьшения количества логических ошибок в различного рода диспутах и спорах. Отправной точкой этой логики служат следующие понятия:

- высказывание — любое суждение, которому можно присвоить значения истинно (true) или ложно (false);
- суждение — законченная мысль, которую можно выразить с помощью естественного языка;
- силлогизм — совокупность правил, выводов, умозаключений (от общего к частному) на основе множества суждений, каждое из которых должно быть записано в допустимой форме.

Для высказываний в традиционной логике существуют законы:

1) тождества ($A = A$);
2) противоречия $\left(A \neq \bar{A}\right)$;
3) исключенного третьего ($A = A$ или $A \neq \bar{A}$) (истина или ложь).

При использовании кванторов допускаются четыре формы суждений для классов S и P:

1) $\forall S$ является P;
2) (отрицание $\forall$)S является P;
3) $\exists$ элементы в S, являющиеся P;
4) (отрицание $\exists$) элементы в S, являющиеся P.

Квантор — логическая операция, которая по соответствующим отношениям строит высказывания, характеризующие подмножества истинности.

Выделяют два вида кванторов: квантор общности (все, математическое обозначение $\forall$) и квантор существования (существует, математическое обозначение $\exists$). Возможны и кванторы-отрицания (никакой, не существует).

Для высказываний как понятий более высокого уровня абстракции сформировалась алгебра из следующих правил:

1) если из S следует P и S — истина, то P — истина;

2) если из S следует P, S — ложь, то P — ложь;

3) если S и P не являются одновременно истиной и S истина, то P ложно;

4) если S и P не являются одновременно истиной и S не истина, то P истина.

Наличие понятия «квантор» является большим достоинством традиционной логики.

В то же время в XIX в. выяснилось, что традиционная логика не обладает однозначностью, что объясняется следующими положениями.

1. Нет формального понятия «дополнение».

2. Нет перехода от положительного к отрицательному значению класса, и наоборот.

3. При логических выводах о существовании элемента какого-либо класса возникают существенные сложности.

4. Не «работает» схема ЕСЛИ ..., ТО ...

Таким образом, традиционная логика не удовлетворяет ранее сформулированным требованиям к логическому описанию и не может быть использована в МЛВ.

Булева алгебра (алгебра логики). Оперирует высказываниями. Чаще всего используют одноместные $y = f(x)$ и двуместные $z = f(x, y)$ высказывания.

Одноместным высказыванием является логическая операция НЕ. Возможны 16 вариантов двуместных высказываний, из которых наиболее известны операции И ($\cap$) и ИЛИ ($\cup$).

В последнее время и, в частности, в предикатах первого порядка, часто используется операция, получившая название «импликация» (табл. 4.3).

Импликация читается так: из a следует b ($a \to b$). Из табл. 4.3 видно, что истинно выражение $(a \to b) = \bar{a} \cup b$, которое будет широко использоваться в предикатах первого порядка.

Существуют следующие правила для булевой алгебры:

1) $a \cup a = a$;

2) $a \cap a = a$;

3) $a \cup \bar{a} = 1$;

4) $a \cap \bar{a} = 0$;

Таблица 4.3. **Импликация**

a	b	$a \to b$	$\bar{a}$	$\bar{a} \cup b$
0	0	1	1	1
0	1	1	1	1
1	0	0	0	0
1	1	1	0	1

5) $a \cup 1 = 1$;
6) $a \cap 1 = a$;
7) $a \cup 0 = a$;
8) $a \cap 0 = 0$.

Законы Де Моргана:

$$\overline{a \cup b} = \bar{a} \cap \bar{b};$$

$$\overline{a \cap b} = \bar{a} \cup \bar{b}.$$

Доказательство равнозначности может иметь разновидности:

1) поэлементное сравнение (семантическое преобразование);

2) синтаксическое преобразование (построение правильно построенных форм в соответствии с правилами алгебры логики).

При синтаксическом преобразовании ведется построение так называемых правильно построенных (по определенным правилам) форм и возможны три варианта:

1) прямое преобразование левой части в правую;

2) нахождение равных промежуточных значений;

3) доказательство истинности или ложности.

Приведем пример для второго варианта:

$$a \to (b \to c) = (a \cap b) \to c.$$

Преобразуем левую часть:

$$a \to (\bar{b} \cup c) = \bar{a} \cup \bar{b} \cup c.$$

Преобразуем правую часть:

$$(\overline{a \cap b}) \cup c = \bar{a} \cup \bar{b} \cup c.$$

Обе части равны и исходное равенство справедливо.

Покажем доказательство либо истинности, либо ложности:

$$(x \cup a) \cap (y \cup \bar{a}) \to (x \cup y);$$

$$\overline{(x \cup a) \cap (y \cup \bar{a})} \cup (x \cup y) = (\overline{x \cup a}) \cup (\overline{y \cup \bar{a}}) \cup (x \cup y) =$$

$$= (\bar{x} \cap \bar{a}) \cup (\bar{y} \cap a) \cup (x \cup y) =$$

$$= x \cup (\bar{x} \cap \bar{a}) \cup y \cup (\bar{y} \cap a) = [(x \cup \bar{x}) \cap (x \cup \bar{a})] \cup$$

$$\cup [(y \cup \bar{y}) \cap (y \cap a)] =$$

$$= (x \cup \bar{a}) \cup (y \cup a) = (x \cup y) \cup (\bar{a} \cup a) = 1.$$

Результирующее выражение истинно и исходное выражение справедливо. Подобные преобразования используются в методе предикатов первого порядка.

Булева алгебра отличается однозначностью. Однако она не может оперировать множествами и подмножествами, а использует операции только с отдельными элементами. Иными словами, в ней отсутствует понятие «квантор».

К моменту поиска аппарата для МЛВ появилась необходимость описывать не только дискретные, но и непрерывные процессы, для чего необходимо использовать непрерывные функции.

Возникает вопрос: справедливы ли законы, выведенные для дискретного случая, для непрерывных функций?

Алгебра логики не дает ответа на этот вопрос. Названные причины привели к тому, что булева алгебра не используется в МЛВ.

Исчисление предикатов. ***Предикат*** (сказуемое) — отношение между множествами элементов соответствующих аргументов. Например: $Q(a_1, a_2, ..., a_n)$, где a_n — аргументы; Q — предикат.

Предикаты первого порядка «наследуют» кванторы от традиционной логики и строгую логику алгебры логики. Предикат является *предикатом первого порядка*, если квантор не входит в сам предикат. Предикаты первого порядка охватывают большую часть отношений. Доказано, что в ряде частных случаев предикаты более высокого порядка могут быть сведены к предикатам первого порядка. Хотя предикаты первого порядка не позволяют описывать возможность и необходимость, убеждения, намерения, цели, метазнания (знания о знаниях), они охватывают широкий спектр возможностей и получили широкое распространение в построении ЭС.

Выделяют три вида аргументов предиката: абстрактные $Q(X, Y)$ или отец (X, Y), конкретные $Q(a, b)$ или отец (Иван, Петр), анонимные $Q(\_, b)$, где _ ставится в случае, когда конкретная величина аргумента не имеет значения.

Процедура перехода от абстрактных аргументов к конкретным называется унификацией.

Предикаты хорошо согласуются с аппаратом правил. Пусть имеется правило

ЕСЛИ X отец Y и Y отец Z, ТО X дед Z.

В предикатном представлении это правило имеет вид

$$\text{Отец}(X, Y) \cap \text{отец}(Y, Z) \rightarrow \text{дед}(X, Z),$$

т. е. фактически записана операция импликации. Используя ее свойства из табл. 4.3, получим

$$\overline{\text{Отец}(X,Y)} \cup \overline{\text{отец}(Y,Z)} \cup \text{дед}(X,Z).$$

В силу этого свойства очень часто четвертым методом описания знаний называют [50] логический метод, подразумевая метод предикатов первого порядка. Обобщим и получим:

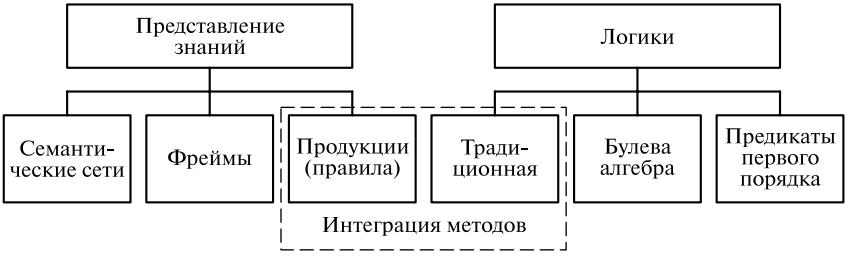


Рис. 4.4. Интеграция методов описания процессов в ЭС

$$\bigcap_{i=1}^{n} A_i \to G, \tag{4.3}$$

где A_i — предпосылка (аксиома, данные, факты); G — цель; выражение (4.3) — формула, предложение, теорема.

Таким образом, логика предикатов первого порядка хорошо согласуется с методом продукций. Возможность интеграции методов описания знаний и логики вывода показана на рис. 4.4.

Выражение (4.3) может быть представлено в виде истинности (фразы Хорна)

$$\bigcap_{i=1}^{n} \bar{A}_i \cup G. \tag{4.4}$$

Анализ формулы (4.4) — это доказательство теоремы. В процессе доказательства возникают три проблемы: выводимости, извлечения результата, оценки достоверности результата.

Для больших наборов правил (тысяча и более) необходимо иметь математический (компьютерный) путь решения этих проблем.

В решении п р о б л е м ы в ы в о д и м о с т и используются следующие понятия.

1. Если формула (4.4) справедлива на всех интерпретациях $\boldsymbol{U}$, то она является общезначимой.

2. Если формула (4.4) ложна на всех интерпретациях, то она невыполнима (противоречива).

3. Формула (4.4) необщезначима, если она не является общезначимой.

4. Формула (4.4) непротиворечива, если она не является противоречивой.

Два предиката F и G эквивалентны ($F \equiv G$), если они совпадают на множестве истинности (ложности). Для выводимости формула (4.4) должна быть общезначима. Доказательство этого положения простое. Если A_i — истинны, используется формула (4.3), если A_i — ложны, — формула (4.4).

В решении проблемы выводимости первоначально использовали формулу (4.4). Однако в прикладных математических операциях выяснилось, что проще доказать ложность отрицания цели.

В этом случае вместо выражения (4.4) следует использовать выражение вида

$$\bigcap_{i=1}^{n} A_i \cap \bar{G}. \tag{4.5}$$

Оно получается из выражения (4.4) взятием отрицания. В дальнейшем будем использовать (4.5).

Одной из разновидностей предиката является квантор. Заметим, что квантор общности не накладывает ограничений на свойства предикатов, в то время как квантор существования имеет серьезные ограничения по области действия: логическая зависимость справедлива только на подмножестве. В связи с этим с кванторами надо «обращаться» аккуратно. Например, справедливо выражение

$$(\forall X)(F(X) \cap G(X)) = (\forall X)F(X) \cap (\forall X)G(X). \tag{4.6}$$

Если в выражении (4.6) заменить конъюнкцию $\cap$ на дизъюнкцию $\cup$, то справедливость теряется.

Аналогично справедливо выражение

$$(\exists X)(F(X) \cup G(X)) = (\exists X)F(X) \cup (\forall X)G(X). \tag{4.7}$$

При замене в выражении (4.7) дизъюнкции $\cup$ на конъюнкцию $\cap$ справедливость теряется.

Для кванторов справедливы законы де Моргана:

$$\overline{(\exists X)F(X)} = (\forall X)\bar{F}(X); \tag{4.8}$$

$$\overline{(\forall X)F(X)} = (\exists X)\bar{F}(X). \tag{4.9}$$

Легко показать, что множества интерпретаций, где предикат истинен, для обеих частей выражения (4.8) совпадают.

Выражение (4.9) доказывается на основе (4.8) заменой F на $\bar{F}$ и взятием отрицаний от левой и правой частей: $\overline{\overline{(\exists X)\bar{F}(X)}} = \overline{(\forall X)F(X)}$.

Для того чтобы оперировать квантором существования, используют процедуру перехода (сколемизацию) от квантора существования к квантору общности. Иллюстрируем на примере:

$$(\forall x)(\exists y)\ P(x, y),$$

$x = \{1, 2\}$ элементы множества x;

$y = \{1, 2\}$ элементы множества y,

отсюда множество интерпретаций $U = \{1,1;\ 1,2;\ 2,1;\ 2,2\}$. Пусть предикат имеет значения $P(1, 1) = 1$; $P(1, 2) = 0$; $P(2, 1) = 0$; $P(2, 2) = 1$. Введем функцию

$$y = f(x) = \begin{cases} 1, \text{ если } x = 1 \\ 2, \text{ если } x = 2 \end{cases}$$

и получим

$$(\forall x)(\forall y)P(x, f(x)),$$

где $f(x)$ — функция Сколема, в рамках которой квантор существования ведет себя как квантор общности.

Очень часто квантор общности опускают при написании предикатов. Приведем более общий пример сколемизации:

$$(\forall x)(\exists y)(\forall z)(\forall u)(\exists w)P(x, y, z, u, w) =$$

$$= (\forall x)(\forall y)(\forall z)(\forall u)(\forall w)P(x, f(x), z, u, g(x, z, u)).$$

Заметим, что в предикатах используется предваренная (префиксная) нормальная форма: кванторы находятся слева от системы предикатов (матрицы).

Все дальнейшие теоретические выкладки проводятся для так называемой фразовой формы, представляющей собой конъюнктивную нормальную форму, где знаки конъюнкций связывают дизъюнкты. Такая форма носит название стандартной предикатной формы. Иногда в ней знаки конъюнкции опускаются и выписываются только дизъюнкты.

Переход от исходной предикатной формы к стандартной предикатной форме осуществляется поэтапно:

1) исключение импликаций по формуле эквивалентности;

2) замена отрицаний с помощью формул де Моргана;

3) замена квантора существования квантором общности с использованием сколемизации;

4) перенесение дизъюнкций внутрь скобок дизъюнктов и приведение формулы к клаузальной (от clause — предложение) форме;

5) вынесение кванторов общности перед предикатами и их опусканием.

Рассмотрим пример:

$$(\forall x)\Big\{P(x)\to\Big\{(\forall y)\Big[P(y)\to P\big[f(x,y)\big]\Big]\cap\Big\{(\forall y)\Big[\overline{\bar{Q}(x,y)\cup P(y)}\Big]\Big\}\Big\}\Big\}.$$

1. Заменим импликации:

$$(\forall x)\Big\{\bar{P}(x)\cup\Big\{(\forall y)\Big[\bar{P}(y)\cup P\big[f(x,y)\big]\Big]\cup\Big\{(\forall y)\Big[\overline{\bar{Q}(x,y)\cup P(y)}\Big]\Big\}\Big\}\Big\}.$$

2. Заменим отрицание и y на w:

$$(\forall x)\Big\{\bar{P}(x)\cup\Big\{(\forall y)\Big[\bar{P}(y)\cup P\big[f(x,y)\big]\Big]\cap\Big\{(\exists w)\Big[Q(x,w)\cap\bar{P}(w)\Big]\Big\}\Big\}\Big\}.$$

3. Введем функцию Сколема $w = g(x)$ и получим

$$(\forall x)\left\{\bar{P}(x)\cup\left\{(\forall y)\left[\bar{P}(y)\cup P\left[f(x,y)\right]\right]\cap\left\{\left[Q(x,g(x))\cap\bar{P}(g(x))\right]\right\}\right\}\right\}.$$

4. Построим префиксную форму и опустим значки кванторов общности:

$$\left\{\bar{P}(x)\cup\left\{\left[\bar{P}(y)\cup P\left[f(x,y)\right]\right]\cap\left\{\left[Q(x,g(x))\cap\bar{P}(g(x))\right]\right\}\right\}\right\}.$$

Реализация логического вывода в экспертных системах. Доказательство общезначимости (противоречивости) при решении проблемы выводимости в любом методе базируется на двоичном семантическом дереве (рис. 4.5). При построении дерева используются следующее правила:

- каждая дуга/узел помечена литералом — предикатом или его отрицанием;
- из одного узла выходят две дуги, помеченные литералами;
- никакая ветвь не содержит двух противоположных литералов;
- никакая ветвь не содержит более одного вхождения одного литерала.

Если множество формул конечно, то соответствующее им семантическое дерево конечно. Каждому его узлу соответствует частичная интерпретация, сопоставляющая истинностные значения некоторых элементов.

Конечное семантическое дерево полно, если каждый его лист, т. е. конечная висячая вершина, соответствует некоторой всюду определенной интерпретации.

Формула является выполнимой, если, по крайней мере, для одного из листьев семантического дерева получено значение «истина». Следовательно, алгоритм выполнимости формул и выводимости результата является переборным и решает задачу для конечного семантического дерева.

Если в предикатах используются кванторы и функции, то семантическое дерево может быть бесконечным. Тогда требуется доказать: если некоторые свойства справедливы на части дерева, то они справедливы и на всем дереве.

Для решения проблемы выводимости были сформулированы методы представления, к которым относятся методы Куайна, редукции, Эрбрана. Однако они алгоритмически сложны и поэтому применя-

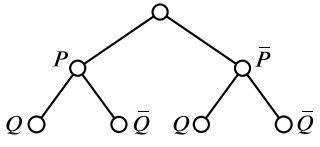


Рис. 4.5. Семантическое дерево

ются редко. Наибольшее распространение получил метод резолюции Робинсона.

В методе резолюций используется формула (4.5): при выводимости цели результат должен быть пустым дизъюнктом. Покажем сказанное на примере.

Пусть имеется система правил (дизъюнктов):

$$P(a) \cup \bar{Q}(a,b); \tag{4.10}$$

$$Q(X,Y) \cup R(X,Y); \tag{4.11}$$

$$S(b); \tag{4.12}$$

$$\bar{R}(a,b). \tag{4.13}$$

Необходимо определить, достижима ли цель $P(a)$.

Возьмем отрицание цели

$$\bar{P}(a). \tag{4.14}$$

Резольвируем (4.14) и (4.10), получаем

$$\bar{Q}(a,b). \tag{4.15}$$

Резольвируем (4.15) и (4.11), получаем

$$R(a,\ b). \tag{4.16}$$

Резольвируем (4.16) и (4.13), получаем

$$\#. \tag{4.17}$$

Иногда строят дерево резолюций, показанное на рис. 4.6.

Метод резолюций Робинсона позволяет работать и с функциями. Он решает проблему выводимости цели из данной системы правил.

В общем случае процедура резольвирования носит комбинаторный (многовариантный) характер, поэтому появилось много разно-

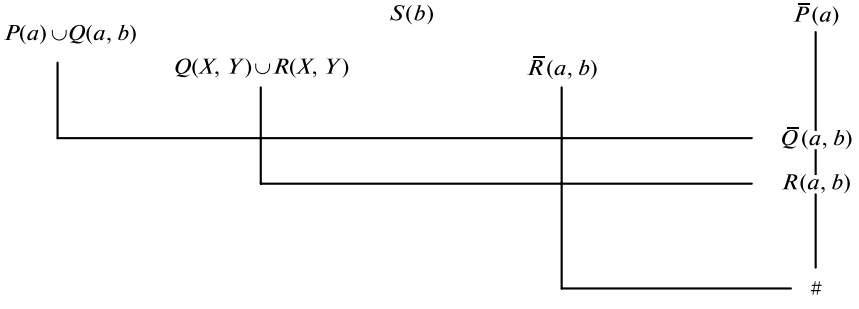


Рис. 4.6. Дерево резолюции

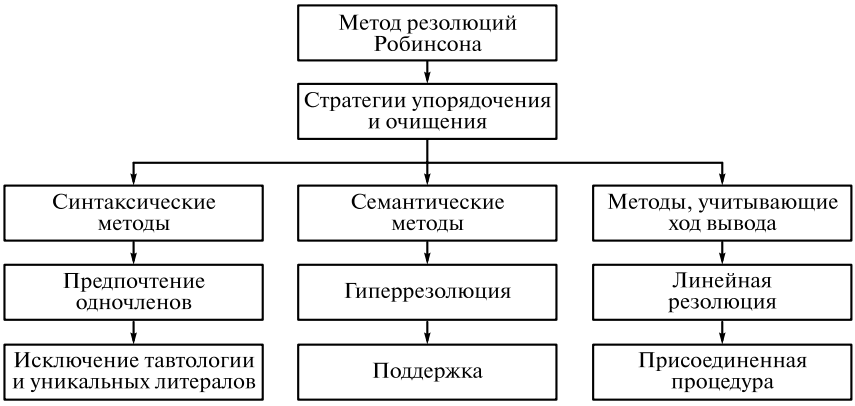


Рис. 4.7. Классификация методов представления

видностей этого метода (рис. 4.7), которые называются стратегиями упорядочения и очищения.

Рассмотрим некоторые из них.

Метод *предпочтения одночленов* рекомендует начинать с дизъюнктов с наименьшим числом литералов.

Метод *исключения тавтологий и уникальных литералов*: если есть литерал и нет его отрицания или наоборот, то резолюции ни в каком случае не произойдут и такие литералы могут не рассматриваться. Если какой-либо дизъюнкт общезначим, его можно не рассматривать. Обычно рассматриваются два резольвирующих дизъюнкта.

Персональный компьютер может рассмотреть несколько резольвирующих дизъюнктов и осуществить *гиперрезолюцию*. Если в дереве резолюций последовательно записать дизъюнкты и резолюции осуществить в соответствии с этой записью дизъюнктов, то получится *линейная резолюция*.

Метод *присоединенной процедуры*: иногда на предварительном этапе анализа возможно оценивать значение истинности некоторых литералов, чтобы не включать их или их отрицания в базовое множество. Оценивание ведется для основных частных случаев. Например, если символ E — равенство чисел, то достаточно оценить частные случаи, например предикаты $E(7, 3)$. Если имеются такие предикаты или их отрицания и указан диапазон изменения аргументов, то для оценки следует составить таблицы возможных основных частных случаев, объем которых достаточно велик. С каждым предикатным символом ассоциируются программы, получившие название присоединенных процедур. Литерал оценивается путем запуска соответствующей процедуры.

Возникает два варианта:

1) если значение истинности есть истина, то все выражение, содержащее данный литерал, можно исключить;

2) если значение истинности есть ложь, то можно исключить вхождение данного литерала в данный дизъюнкт и, например, выражение $P(x) \cup Q(a) \cup E(7, 3)$ может быть заменено на $P(x) \cup Q(a)$.

Таким образом, рассмотрен метод решения проблемы выводимости результата из системы правил. При определении выводимости с помощью метода резолюций Робинсона к системе правил добавляют отрицание целей.

При решении проблемы извлечения результата к системе правил добавляют дизъюнкт, состоящий из отрицания цели и цели. Покажем это на примере.

Пусть имеется система правил:

1) Петр находится там же, где и Иван;

2) Иван находится в школе.

Цель: определить, где находится Петр.

Введем предикат $B(X, Y)$, где X — имя; Y — место нахождения. Тогда система правил имеет вид:

1) B(Иван, Y) $\rightarrow$ B(Петр, Y);

2) B(Иван, школа).

Цель: B(Петр, Y).

Цель выводима, как это следует из рис. 4.6. Получим результат вывода (рис. 4.8).

Таким образом, найден математический аппарат для теоретического решения проблемы извлечения результата.

Вместе с тем такой математический аппарат не всегда удобен для прикладных целей. В связи с этим в прикладных целях для поиска решений используют другие методы: поиск в глубину и в ширину — в языке Пролог, прямая и обратная аргументация — в оболочке экспертной системы GURU.

Проблему оценки достоверности результата решают исходя из того, что и посылки, и правила могут быть заданы с некоторой степенью достоверности.

Уже в первых ЭС EMYCIN, предназначенных для медицинской диагностики, диагноз выдавался с некоторой достоверностью.

Так, в пакете GURU имеются следующие оценки:

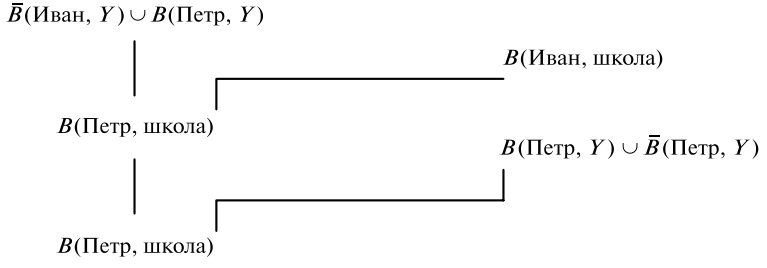


Рис. 4.8. Извлечение результата методом резолюций Робинсона

- вид стратегии оценки посылки данного правила (точный, допустимый, эталонный — для всех правил сразу);
- порядок, в котором рассматриваются конкурирующие правила (приоритет, стоимость, в порядке набора правил, случайный порядок);
- точность системы при выводе значений переменной (минимальный, тщательный);
- степень ответа на команду КАК от 0 (нет ответа) до 6 (наиболее полный ответ);
- пределы ответа на команду ПОЧЕМУ от 0 (нет ответа) до 5 (наиболее тщательный ответ).

Описанные подходы — прерогатива модальной логики. В понятии «модальный» выделяют следующие составляющие:

- аподиктический (apodectikos — достоверный, неопровержимый), например, целое больше части.
- ассерторический (assertirius — утвердительный) — утверждает факт, но не выражает его непреклонности, логической необходимости: яблоко разделено на части.
- проблематичный (problematicos — возможный) — возможный, но не доказанный, сомнительный: высказывается возможность или вероятность высказывания или его отрицания, но не гарантируется его непреложность.

Появилось понятие «нечеткие знания», которое определяется неполнотой логики (немонотонная логика, логика умолчания), ненадежностью и нечеткостью данных.

На математический аппарат для определения достоверности могут претендовать теория вероятностей, теория фактора уверенности, теория нечетких переменных.

При определении достоверности, как правило, возникает процедура из последовательных событий. В связи с этим теория вероятности для данных целей подходит плохо. Так, если каждое событие в последовательной цепи имеет вероятность $P = 0{,}5$, то результат для четырех событий $R = (0{,}5)^4 = 0{,}0625$, а такими цифрами теория вероятности не оперирует. В связи с этим были предложены два других метода: фактор уверенности и нечеткие переменные.

Фактор уверенности cf — это целая величина от 0 до 100, задаваемая экспертным путем. Алгоритмы обработки величины также задаются экспертным путем и называются алгебрами.

Возможные алгебры отражены в табл. 4.4 при $cf1 = 40$, $cf2 = 60$.

Нетрудно видеть, что два первых правила дают малоинформативные результаты. В частности, вторая алгебра — суть алгоритма теории вероятностей. В связи с этим боле предпочтительны третий и четвертый алгоритмы.

Более подробно характеристики фактора уверенности рассмотрены при обсуждении интеллектуального пакета GURU.

Нечеткая переменная представляет собой набор значений, каждый из которых может быть со своим фактором уверенности.

Таблица 4.4. **Возможные алгебры обработки фактора уверенности**

Алгебра	Алгоритм	Результат
1. Минимум	$\min(cf1, cf2)$	40
2. Произведение	$(cf1 * cf2)/100$	24
3. Среднее	$(cf1 + cf2)/(2)$	50
4. Баланс	$\frac{cf1 \cdot cf2}{100} \cdot \left(2 - \frac{\max(cf1, cf2)}{100}\right)$	34

Пусть $M = \{1, 2, 3, 4\}$ — набор из четырех чисел, нечеткая переменная $F = \{1\ cf\ 100,\ 2\ cf\ 60,\ 3\ cf\ 10,\ 4\ cf\ 0\}$. Представим это число графически (рис. 4.9). Говорят, что эта переменная F для малых чисел.

Другой записью этой нечеткой переменной может быть

$$F = 1/1 + 0{,}6/2 + 0{,}1/3 + 0/4.$$

В общем виде выражение для нечеткой переменной записывается так:

$$F = \sum_{i=1}^{n} \mu_F(U_i)/U_i.$$

Для нечетких переменных установлена алгебра в виде следующих правил.

1. Отрицание (для одной переменной)

$$\bar{F} = \sum_{i=1}^{n} \{1 - \mu_F(U_i)/U_i\}.$$

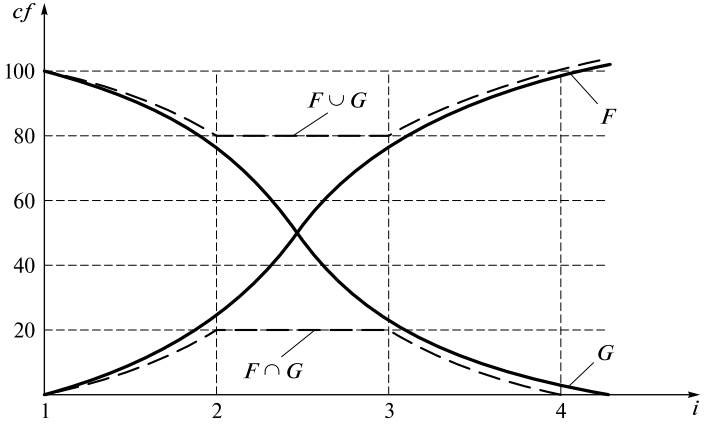


Рис. 4.9. Геометрическая иллюстрация нечеткой переменной

Приведем правила для двух переменных. Для этого введем еще одну переменную

$$G = 0/1 + 0{,}1/2 + 0{,}6/3 + 1/4.$$

В нечетких переменных вместо операций сложения и умножения используют операции max и min.

2. Объединение

$$F \cup G = \sum_{i=1}^{n} \max\left\{\mu_F(U_i)/U_i ; \mu_G(U_i)/U_i\right\},$$

$$F \cup G = 1/1 + 0{,}6/2 + 0{,}6/3 + 1/4.$$

3. Пересечение

$$F \cap G = \sum_{i=1}^{n} \min\left\{\mu_F(U_i)/U_i ; \mu_G(U_i)/U_i\right\},$$

$$F \cap G = 0/1 + 0{,}1/2 + 0{,}1/3 + 0/4.$$

Две предыдущие операции показаны на рис. 4.9.

4. Декартово произведение

$$F1 = F * G = \sum_{i=1}^{n}\sum_{j=1}^{n} \min\left\{\mu_F(U_i)/U_i ; \mu_G(V_j)/U_j\right\},$$

$$F * G = \begin{pmatrix} 0 & 0{,}1 & 0{,}6 & 1 \\ 0 & 0{,}1 & 0{,}6 & 0{,}6 \\ 0 & 0{,}1 & 0{,}1 & 0{,}1 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

5. Свертка

$$F1 * H = \sum_{i=1}^{n}\sum_{r=1}^{n} \max\left[\min\left\{\mu_F(U_i, V_j); \mu_H(V_j, W_k)\right\}\right].$$

Для примера используем матрицу $F1$ и матрицу H:

$$H = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0{,}4 & 0{,}4 \\ 0 & 0 & 0{,}5 & 0{,}9 \\ 0 & 0 & 0{,}5 & 1 \end{pmatrix};$$

$$F1 \times H = \begin{pmatrix} 0 & 0 & 0{,}5 & 1 \\ 0 & 0 & 0{,}5 & 0{,}6 \\ 0 & 0 & 0{,}1 & 0{,}1 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Обсудим общие возможности логического представления знаний.

Сравнительные характеристики методов логического вывода с позиций названных требований приведены в табл. 4.5.

Таблица 4.5. **Характеристики методов логического вывода**

Метод	Достоинства	Недостатки
Традиционная логика	Наличие большого практического опыта. Оперирование множествами и подмножествами (кванторы)	Неоднозначность описания. Несогласованность с методами описания знаний. Трудность получения решения в сложной структуре выводов. Невозможность описания непрерывных процессов
Булева алгебра	Однозначность описания. Высокая степень формализации	Невозможность оперирования множествами и подмножествами. Невозможность описания непрерывных процессов
Предикаты первого порядка	Однозначность описания. Высокая степень формализации. Оперирование множествами и подмножествами. Получение решения в сложной структуре выводов. Описание непрерывных процессов. Простота получения решения	Несогласованность с методами описания знаний (метод продукции)

4.3. Язык программирования ПРОЛОГ

Язык ПРОЛОГ резко отличается от процедурных языков по своей философии. В силу этого опишем особенности языка.

1. Язык ПРОЛОГ является логическим и декларативным. Это означает, что он фиксирует отношения (ЧТО должно быть сделано), тогда как процедурный язык указывает на технологию получения результата.

Покажем это различие. Пусть имеется зависимость

ЕСЛИ R и S, то P,

где P, R, S — предикаты.

Процедурный язык трактует задачу так, чтобы решить задачу P, надо сначала решить задачу R, а затем — S. Декларативная трактовка: P истинно, если R и S истинны.

В ПРОЛОГе не существует понятия «присваивание». Некоторым его аналогом служит понятие «унификация». Это значит, что в процессе выполнения программы язык какое-то время оперирует абстрактными, т. е. алгебраическими, переменными.

В процедурных языках переменная действует на протяжении всей программы, тогда как в ПРОЛОГе — только в переделах правила. Следовательно, разные переменные в разных правилах могут иметь одинаковые обозначения. С другой стороны, если в $(n + 1)$-м правиле необходимы данные из предиката n-го правила, то последний следует повторить в $(n + 1)$-м правиле.

В то же время ПРОЛОГ не является «чисто» декларативным языком, поскольку характер выполнения программы зависит от расположения правил.

2. Составными элементами программ на языке ПРОЛОГ являются предикаты, которые делятся на стандартные, не требующие предварительного объявления, и нестандартные, требующие предварительного объявления.

Назначение и место ряда стандартных предикатов, которых насчитывается свыше 50, показано на рис. 4.10.

3. Необычна форма записи правил:

C:- A, B

что означает

C, ЕСЛИ A и B.

Нетрудно видеть, что операция ЕСЛИ обозначается (:-), И — запятой (,), ИЛИ — точкой с запятой (;). Операция ИЛИ используется редко. Чаще при наличии правила с этой операцией оно заменяется на два правила.

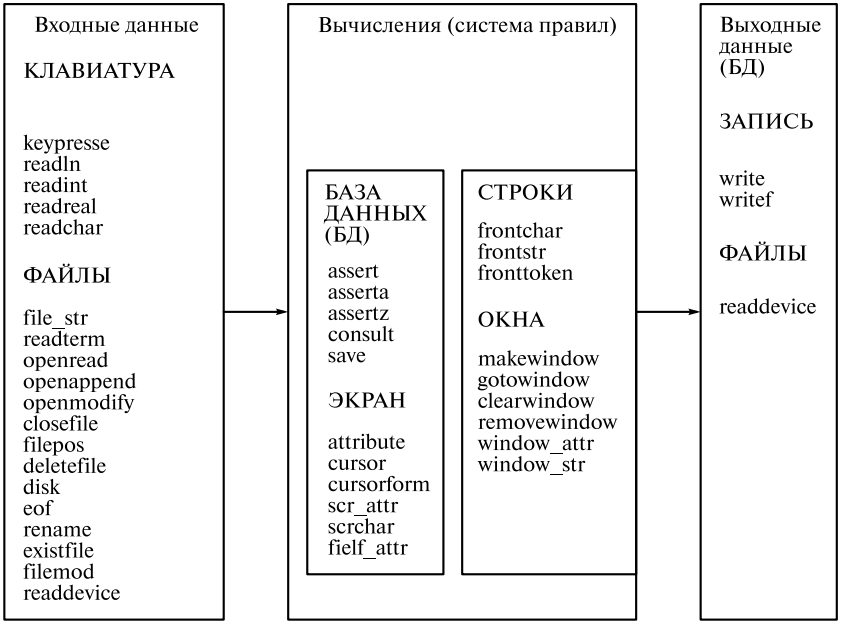


Рис. 4.10. Место стандартных предикатов языка ПРОЛОГ

4. Выявление (поиск) результата осуществляется по семантическому дереву в глубину.

В этом алгоритме первоначально просматривается левая ветвь семантического дерева. Если в каком-то ее узле результат не получен, происходит возврат на один уровень выше — откат. Он может вызываться и искусственно. Дальше просматривается следующая левая ветвь и т.д. Алгоритм прост в реализации, однако является переборным. При этом требуется для объяснений запоминание «пройденных» вершин, для чего требуется достаточная память (стек).

Если заранее известно, что результат должен быть только один, то перебор можно сократить после получения результата применением оператора отсечения (символ !). При его применении остальные ветви дерева и узлы «забываются» (выбрасываются). Применение оператора отсечения, давая удобства программисту, фактически нарушает идею логического программирования, вклиниваясь в логическую семантику.

5. Недостаточно использования возможностей только декларативных языков, поскольку требуются алгоритмические приемы (циклы). В языке ПРОЛОГ они могут осуществляться с использованием откатов или рекурсий.

При использовании первого способа циклы могут организовываться следующим образом:

- естественным откатом, если условие не выполнено;
- искусственным откатом при помощи стандартного предиката fail;
- применением нестандартного предиката repeat (повторение) с фиксацией условия остановки цикла.

При использовании откатов приходится возвращаться на число уровней более двух, а этот режим неустойчив.

6. При использовании *рекурсий* требуется еще больше памяти. Приведем пример рекурсии:

```
write_list ([])
write_list ([H|T]): – write (H),
    write_list (T).
```

Одновременно показана процедура рекурсии: некий аргумент (например, список) делится на две части: «голову» (один элемент) и «хвост» (остальное). Для «головы» проводится операция write во втором правиле и результат записывается в стек. Дальше остается только «хвост», для которого процедура повторяется.

Необходимо правило остановки рекурсии — пустой список, т. е. первое правило. После этого рекурсия прекращается, и из стека на экран считываются результаты операции в обратной последовательности.

7. Процесс рассмотрения компьютером набора правил и вывода результата — консультация. Она возможна как в диалоге, так и в автоматическом режиме. Путь поиска называют трассой.

8. Объяснения включают в себя две группы вопросов:

- при вопросе КАК компьютер должен выдать на экран трассу в виде использованной при выводе системы правил;
- вопрос ПОЧЕМУ имеет различные трактовки, которые детально опишем при рассмотрении конкретных программ.

9. В силу декларативного характера языка ПРОЛОГ в нем сложно выполнять условные переходы и циклы. Цикл означает возврат по дереву на уровень выше. Можно обеспечить возврат максимум на два уровня.

10. Программа на языке ПРОЛОГ имеет довольно жесткую структуру с секциями (разделами): domains (поля), database (внешняя база данных), predicates (предикаты), goal (цель), clauses (предложения, правила). Данные (факты) фиксируются в виде базы данных (внутренней или внешней по отношению к программе). Цель может задаваться внутри программы или через интерфейс языка ПРОЛОГ (внешние цели), где программа должна быть специально запущена.

Секция domains не обязательна, но она значительно сокращает время компиляции. В секции задаются типы данных и используется четыре формата:

- первый содержит элементы стандартного типа: integer (целые числа), real (действительные числа), character (знаковый), symbol (символьный), string (строковый);
- второй формат используется при применении списков, например,

```
name = integer*,
```

где символ (*) означает список, в данном случае — целых чисел;
- третий формат служит для описания сложных объектов;
- четвертый формат используется тогда, когда к файлу обращаются по символьному (логическому) имени.

Секция database присутствует при использовании внешней базы данных. Задается имя БД с указанием переменных, объявленных в секции domains. Сама база данных создается в другом файле. Вызов файла, в котором не должно быть повторений, осуществляется стандартным предикатом consult программы. База данных в процессе выполнения программы может пополняться (предикаты assert, asserta, assertz) или уничтожаться (retract, retractall).

В *секции predicates* объявляются нестандартные предикаты и их аргументы.

В *секции goal* задается внутренняя цель. Задание заканчивается точкой. Если цель внешняя, то секция отсутствует. В этом случае цель определяется через четырехоконный интерфейс пользователя. В левом верхнем окне пишется программа, в верхнем правом — указывается цель и получается результат, а в правом нижнем указываются предикаты, выполняемые в данное время программой. Возможен полноэкранный вариант интерфейса пользователя. Тогда на экране присутствует программа, а цель и результат смещаются в ее конец.

Возможно и создание (в секции clause) собственного интерфейса пользователя, для чего можно использовать предикаты для формирования окон (см. рис. 4.10).

В *секции clauses* записывают правила, которые заканчиваются точкой. Различают два варианта: факты и правила. Факты образуют внутреннюю базу данных.

11. После составления программы осуществляется ее компиляция. Если компиляция неудачна, необходимо найти ошибки. Простые ошибки могут быть исправлены немедленно.

Сложность использования языка ПРОЛОГ (Prolog) связана напрямую с большим числом предикатов и заключается в слабой диагностике ошибок. Для проведения диагностики вводится стандартный предикат trace, который записывается в начале программы. Если trace стоит без аргументов, проверяются все предикаты. Если есть предположение об ошибках в каких-то предикатах, в trace указываются их имена.

12. Возможны два варианта реализации на языке ПРОЛОГ:

- на правилах — прямое применение ПРОЛОГа при поиске, в котором сложно запрограммировать процедуру объяснений;
- на логике — при использовании какой-либо оболочки ЭС, при этом правила вводятся в базу правил и при необходимости могут быть трансформированы легко.

Примеры программ на языке ПРОЛОГ приведены в приложении 2.

ПРОЛОГ ограничен по размерам примеров, вызывает затруднения в реализации векторно-матричных операций, характеризуется высокой трудоемкостью построения программ.

По мнению оппонентов язык ПРОЛОГ имеет следующие недостатки:

- слабые средства защиты от незначительных ошибок в написании команд;
- язык не только декларативный, но и процедурный, поскольку порядок расположения правил существенен и требует понимания деталей процедуры отката;
- стандартные предикаты имеют побочные эффекты, что затрудняет применение языка для параллельной обработки алгоритмов;
- слабо развита модульность, и нет модулей в обычном понимании этого слова;
- база данных весьма условна.

По некоторым позициям критики можно аргументированно возразить, а по другим — подискутировать. Однако ряд позиций касаются ограничений технических характеристик и с ними нельзя не согласиться.

Основные технические ограничения языка ПРОЛОГ таковы:

- в предикате нельзя использовать более 50 параметров;
- максимальное число модулей (по директиве include) — 10;
- максимальное количество имен в секции domains — 250, альтернатив — 250;
- максимальное число предикатов — 300;
- максимальное количество переменных в предложении — 100;
- максимальное количество предложений для каждого предиката — 500.

Сюда следует добавить ограничения по области памяти, среди которых наиболее уязвимы по объему памяти области STACK (рекурсивные алгоритмы) и CODE (код, полученный в результате компиляции). По умолчанию области CODE отводится 16 Кбит, что недостаточно для объемных программ. Эту область можно увеличить лишь до 160 Кбит.

Перечисленные ограничения послужили толчком для использования ПРОЛОГ-идей (оболочек) на базе процедурных языков программирования. К таким оболочкам относятся GURU и G2.

4.4. Оболочки экспертных систем

Напомним, что экспертная система без правил образует оболочку. Оболочка GURU реализована на базовом языке С, а языком пользователя является фактически PASCAL, дополненный операторами языка SQL.

GURU имеет по сравнению с ПРОЛОГом более слабые ограничения. Многие характеристики (количество правил в наборе, переменных в наборе правил, таблиц в базе данных) помечены словом «неограниченные». Такие характеристики, как количество записей в таблице (65 535), символов в записи (65 535), фактически тоже являются неограниченными. Вариантов стратегии поиска правил (50) и алгебр фактора уверенности (16) более чем достаточно для практических нужд. Ограничен лишь объем набираемого текстового файла (16 Кбит).

GURU относят как к интеллектуальным (статическая экспертная система), так и к интегральным пакетам. Дело в том, что GURU содержит следующие составляющие: экспертные системы, база данных, электронная таблица, массивы, формы экранные, графическая составляющая, естественный язык. Декларируется, что все составляющие могут взаимодействовать между собой (свойство синергии).

Связь с пакетом может осуществляться следующими способами:

- на естественном языке с помощью внутреннего словаря (примерно на 500 слов);
- введением команд через командную строку;
- составлением программ;
- использованием меню с глубиной до семи уровней.

Оболочку GURU можно разделить на две части: собственно оболочка, с помощью которой строится экспертная система, и среда, фиксирующая настройки e.*. Например, e.lstr = 70 устанавливает предельную длину строки в 70 символов. Насчитывается свыше 50 настроек. В среде имеется около 20 утилит (#).

Программы разных составляющих имеют разные расширения. Далее будем использовать расширения *.itb — для базы данных, *.ipf — для связи составляющих, *.rss — для экспертной системы, *.rcs — для скомпилированной программы экспертной системы. Программа экспертной системы является единственной, которая в GURU компилируется.

Экспертная система может работать автономно или во взаимодействии с другими составляющими (например, базой данных).

В *автономном режиме* программа имеет следующие разделы: GOAL, INITIAL объявления переменных, вывод результатов (DO), правила (RULE и номер), вывод обозначений переменных (VAR) и END. В каждом правиле выделяют разделы PRIORITY, COST, IF, THEN, REASON. PRIORITY устанавливает (по наибольшему значению в интервале от 0 до 100) порядок выполнения правил, COST

аналогичен PRIORITY с установкой по наименьшему значению, REASON формулирует правило, выдаваемое пользователю в процедуре объяснения.

Работа с вопросами не всегда пригодна. Иногда целесообразно использовать внутреннюю БД. В GURU 2.0 БД представляется в виде набора таблиц, связи в ней не устанавливаются. Они могут быть установлены в процессе работы с помощью языка SQL.

В GURU 2.0 декларируется, что программные элементы могут *взаимодействовать друг с другом* или работать друг на друга, в частности БД на ЭС, и обратно. На самом деле это не так, таблицу БД приходится преобразовывать в двумерный массив ar1(A, i). Для преобразования необходимо знать число строк и столбцов таблицы. Число строк может быть получено с помощью специальной утилиты. Такой утилиты для числа столбцов нет, поэтому в последнем столбце ставится специальная метка, характеризующая количество столбцов.

При использовании БД оставим вариант работы в режиме диалога. БД может быть создана заранее с помощью меню БД или программным способом. Объяснения задаются программным способом.

Примеры программ в рамках GURU приведены в приложении 2.

К недостаткам статической ЭС GURU следует отнести следующие:

- сложность использования даже приведенного примера для практических целей;
- затруднения при наличии значительного количества данных, для размещения которых требуется несколько связанных таблиц;
- сложное взаимодействие базы данных с ЭС;
- название полей латинскими буквами, что не всегда удобно для пользователя;
- отсутствие прямого учета времени, что необходимо в динамических системах управления.

Многие из этих недостатков отсутствуют в оболочках динамических ЭСРВ, например G2. Оболочка G2 труднодоступна по разным причинам; G2 является ЭС реального времени. Наиболее известны следующие области ее применения:

- мониторинг (слежение за чем-либо) в реальном масштабе времени;
- в системах управления высшего уровня;
- диагностика;
- составление расписаний;
- системы-советчики операторов;
- системы проектирования.

Наибольшее распространение оболочки получили в нефтегазовой промышленности.

При построении оболочки были использованы следующие принципы:

- объектно-ориентированное проектирование;
- объектно-ориентированное программирование;
- широкое использование стандартов;
- использование различных платформ;
- ориентация на классы задач, а не отдельные задачи;
- применение архитектуры клиент-сервер.

В состав оболочки входят следующие блоки: база знаний (БЗ); машина логического вывода; планировщик; подсистема моделирования; среда разработчика; интерфейс пользователя; интерфейс с внешним окружением.

Имеется две части *базы знаний:* набор правил для одного приложения и библиотека для БЗ (для двух и более приложений). Данные могут поступать из БД, контрольно-измерительной аппаратуры, имитационной модели, а также из других правил. В построении знаний используют три вида иерархии: классов, модулей, рабочих пространств. Выделяют две разновидности связей: простые связи и отношения.

В *машине логического вывода* правила могут запускаться данными; переменными, которые формируются в другом правиле; возбуждением правил через *n* секунд; переменными, входящими в условия, которым только что присвоены значения; при запуске приложения; при перемещении пользователем графического изображения на экране в виде мнемонической схемы; после изменения отношения между объектами.

В программе G2 наряду с консеквентом (правой частью правила) then используются более быстрые консеквенты when, whenever.

Планировщик — блок, служащий для управления другими блоками.

Подсистема моделирования может работать с тремя группами уравнений: дифференциальными, разностными, алгебраическими. Модель может работать от реальных датчиков, измерителей и т.д.

В *среду разработчика* входят редактор на естественном языке; интерфейс пользователя для работы с пиктограммами, учитывающими передвижение объектов, изменение цвета связей; изображения (циферблаты, измерители, диаграммы, таблицы); элементы управления; сообщения, управление доступом пользователя к элементам схемы; создание дополнительных элементов меню; средства инспекции и отладки. (Инспекция — поиск элементов схемы на основе типов или принадлежности к классам.)

К *интерфейсу пользователя* относятся редактор пиктограмм, графика изменения масштабов, графика элементов, меню перерисовки экранов.

К *интерфейсу с внешним окружением* относятся интерфейс с реальной системой управления, интерфейс с системой датчиков, интерфейс с БД (как внутренний, так и внешний).

G2 по своим характеристикам более подходит для создания АСУТП.

Следует отметить, что экспертную систему на ПРОЛОГ-идеях можно сформировать и на алгоритмическом языке. Далее демонстрируется экспертная система «Прием на работу», реализованная на СУБД InterBase в среде Delphi (язык Object Pascal).

4.5. Пример прикладного проектирования экспертной системы

В качестве примера рассмотрим процесс разработки ЭСРВ «Прием на работу».

Для выполнения проектных работ необходимы специалисты в соответствии со штатным расписанием. Отбор претендентов осуществляется по анкетным данным. В помощь руководителю необходимо спроектировать соответствующую ЭСРВ (базу знаний). Далее описаны основные этапы проектирования.

Идентификация. В качестве цели проектирования примем построение ЭСРВ в помощь конечному пользователю.

Цель работы при построении ЭСРВ — автоматизация интеллектуальных аспектов работы руководителя: выработка решений-советов и после их согласования с руководством — принятие решений, передаваемых на объект управления системы с прогнозом последствий их реализации.

Из-за относительной простоты проектируемой системы разработчиком является специалист-исследователь, совмещающий в себе функции эксперта, конечного пользователя (КП) и программиста. Ресурсные ограничения в данном случае не рассматриваются. Правила работы ЭСРВ формируются на основе представления исследователем-экспертом порядка ручной работы КП по управлению кадрами.

КП может быть несколько, поэтому в качестве вычислительного средства выбрана СУБД InterBase в среде Delphi, что позволяет легко реализовать сетевой режим клиент-сервер.

Первичные исследования процесса позволили сформировать структуру системы (рис. 4.11) и ее общее описание (рис. 4.12).

Концептуализация. В качестве терминов примем терминологию, использованную на рис. 4.12. Из рис. 4.11 видно, что в ЭСРВ отсутствует реальный объект управления и, следовательно, система датчиков и табло.

Схема процесса управления в такой системе отображена на рис. 4.13.

База знаний представлена базой данных (табл. 4.6, 4.7) и базой правил (табл. 4.8).

Считается, что данные и правила достоверны и учет недостоверности результатов не проводится.

Предполагается, что в процессе работы КП с БЗ правила и результаты их работы могут быть скорректированы. Правила могут

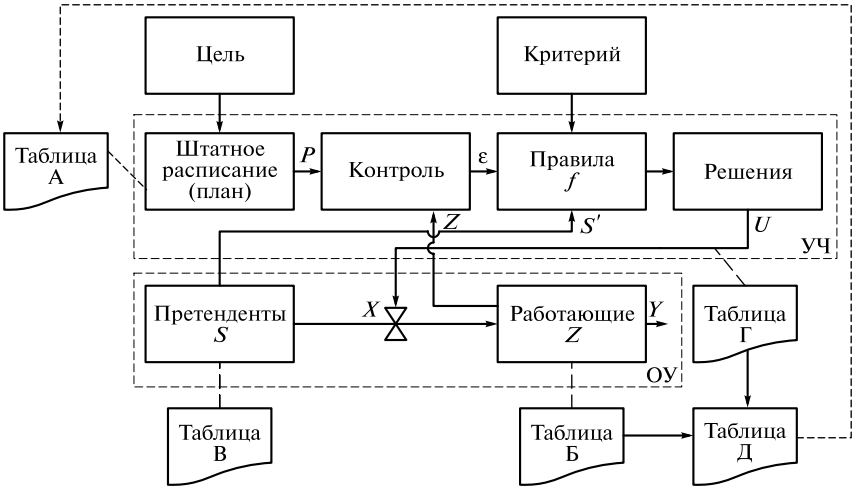


Рис. 4.11. Схема системы управления приемом на работу:

P — план; *X* — вход; *Y* — выход; ε — отклонение от плана; *Z*, *S* — состояния работающего; *U* — управление

корректироваться и инженером по знаниям в процессе получения новых знаний о предметной области.

Формализация. Работа ЭСРВ проводится в дискретном времени, состоящем по умолчанию из трех интервалов времени, состав-

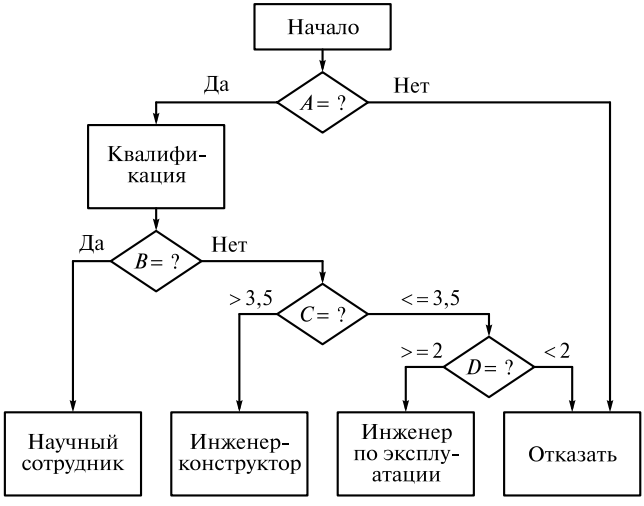


Рис. 4.12. Алгоритм приема на работу в научном учреждении:

A — ученая степень (degree); *B* — открытие (discov); *C* — средний балл (grade); *D* — стаж, лет (exper)

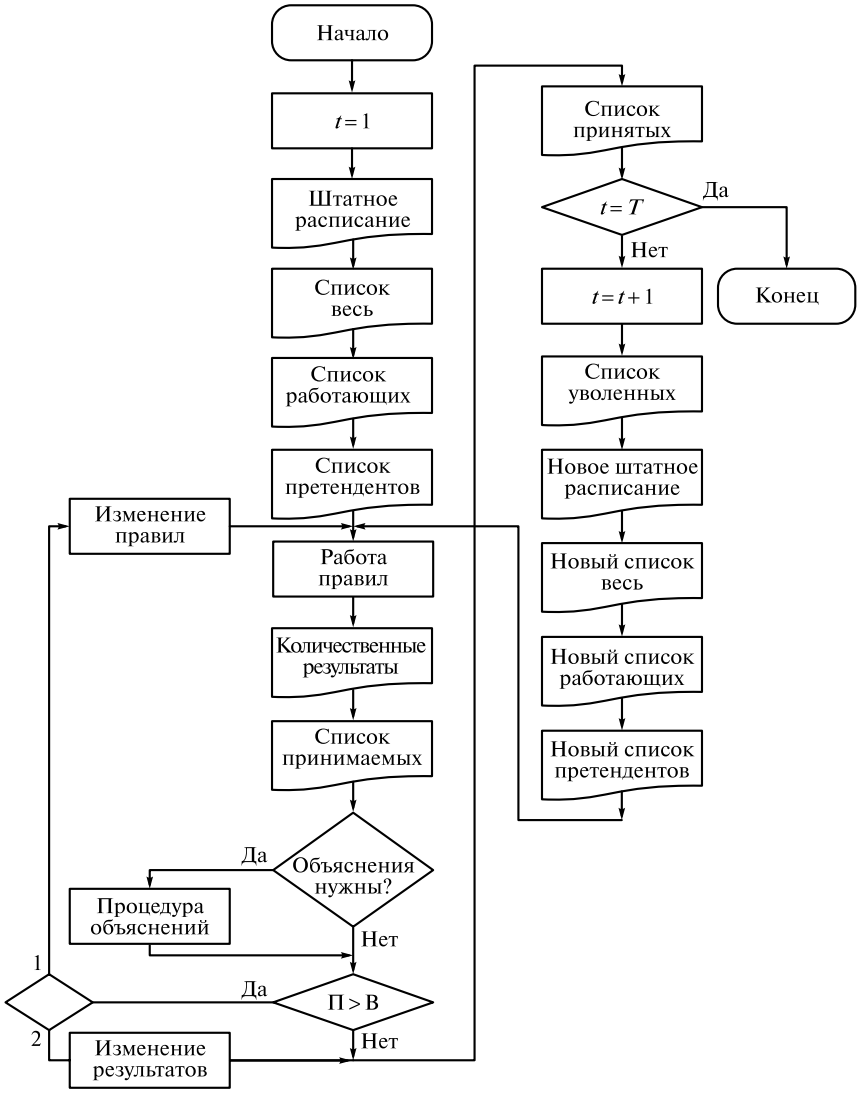


Рис. 4.13. Процедура работы с пользовательской БЗ:
П — принимаемые; В — вакансии

ляющих сессию функционирования системы. Каждый интервал времени называют также циклом.

После окончания сессии система должна вернуться в исходное состояние.

Формально описание ЭСРВ представлено данными (см. табл. 4.6, 4.7), системой продукций (см. табл. 4.8); математическим описанием

дискретной (машина логического вывода) и непрерывной составляющими модели управляющей части; описанием объекта управления в форме разностного уравнения; описанием действия среды, проявляющемся в увольнении людей и появлении претендентов в начале каждого цикла.

Для математического описания и описания объекта управления и действия среды введем дополнительные обозначения: дискретные моменты времени $t \in [1, T]$, $[t] = [(t) - (t - 1)] = \text{const}$ — интервал времени. Считаем, что на интервале $[t]$ скорости x, y, x', y', s, s' постоянны и меняются только на границе интервала времени. Пусть $[t]$ — один рабочий день.

Обозначим $x[t]$, $y[t]$ — число принятых и уволенных за день. Очевидно, что x, y, s — векторы. Например, $x = \{x_1, x_2, x_3\}^{\mathrm{T}}$, T — признак транспонирования; $\{x_v, v \in [1, 3]\}$, $v = 1$ — научный сотрудник, $v = 2$ — инженер-конструктор, $v = 3$ — инженер по эксплуатации. S' на рис. 4.11 характеризует информацию о соответствующих координатах: $x = x'$, $y = y'$, $s = s'$. План p может задаваться двояко: ежедневный; с накоплением (например, с начала месяца).

«Оцифруем» словесно выраженные решения r, суммарное количество которых R ($r \in (1, R)$) равно числу претендентов: $i = 0$ — «отказать»; $i = 1$ — «научный сотрудник»; $i = 2$ — «инженер-конструктор»; $i = 3$ — «инженер по эксплуатации».

Тогда циклическая процедура представляет собой систему правил следующего вида:

$$i_r[t] = \begin{cases} 0, A = \text{нет}; \\ 0, A = \text{да}, B = \text{нет}, C < 3{,}5, D < 2 \\ 1, A = \text{да}, B = \text{да} \\ 2, A = \text{да}, B = \text{нет}, C \geq 3{,}5 \\ 3, A = \text{да}, B = \text{нет}, C < 3{,}5, D \geq 2, \end{cases}$$

где $r \in [1, R]$; $i \in [1, 3]$.

Таблица 4.6. **Штатное расписание (начальное)**

Время	Должность	План	Факт	Вакансии
1	Научный сотрудник	10	5	
1	Инженер-конструктор	9	7	
1	Инженер по эксплуатации	12	9	

Таблица 4.7. **Список штатного персонала (начальный)**

Время	Фамилия	Ученая степень	Откры-тия	Средний балл	Стаж	Статус	Должность
1	Водянов	Да	Да	4,8	7	Работает	Научный сотрудник
1	Каримов	Да	Да	3,3	5	Работает	Научный сотрудник
1	Крамской	Да	Да	4,2	8	Работает	Научный сотрудник
1	Крикунов	Да	Да	3,5	9	Работает	Научный сотрудник
1	Трубецков	Да	Да	4,1	15	Работает	Научный сотрудник
1	Крымов	Да	Нет	4,0	4	Работает	Инженер-конструктор
1	Мамедов	Да	Нет	3,9	6	Работает	Инженер-конструктор
1	Орлов	Да	Нет	3,7	3	Работает	Инженер-конструктор
1	Синцов	Да	Нет	4,5	1	Работает	Инженер-конструктор
1	Петрович	Да	Нет	4,2	7	Работает	Инженер-конструктор
1	Тараканов	Да	Нет	4,1	3	Работает	Инженер-конструктор
1	Травкин	Да	Нет	5,0	9	Работает	Инженер-конструктор
1	Хохлов	Да	Нет	4,0	4	Работает	Инженер-конструктор
1	Черкас	Да	Нет	4,8	2	Работает	Инженер-конструктор
1	Касымов	Да	Нет	3,3	3	Работает	Инженер по эксплуатации

1	Контуров	Да	Нет	3,0	4	Работает	Инженер по эксплуатации
1	Купцов	Да	Нет	3,3	5	Работает	Инженер по эксплуатации
1	Ребров	Да	Нет	3,4	7	Работает	Инженер по эксплуатации
1	Ремезов	Да	Нет	3,4	5	Работает	Инженер по эксплуатации
1	Соколов	Да	Нет	3,3	3	Работает	Инженер по эксплуатации
1	Тиунов	Да	Нет	3,0	6	Работает	Инженер по эксплуатации
1	Троекуров	Да	Нет	3,0	4	Работает	Инженер по эксплуатации
1	Щавель	Да	Нет	3,2	9	Работает	Инженер по эксплуатации
1	Карпов	Да	Да	3,2	1	Претендует	
1	Крылов	Да	Да	3,1	2	Претендует	
1	Синцов	Да	Нет	4,5	1	Претендует	
1	Симонов	Да	Нет	3,9	2	Претендует	
1	Иванов	Да	Нет	3,2	3	Претендует	
1	Козлов	Да	Нет	3,4	4	Претендует	
1	Петров	Нет	Да	4,0	1	Претендует	
1	Гуров	Нет	Нет	4,0	3	Претендует	
1	Цветков	Да	Нет	3,0	1	Претендует	

Таблица 4.8. **Таблица правил**

Номер правила	Ученая степень	Открытия	Знак, балл	Средний балл	Знак, стаж	Стаж	Должность
1	Нет						Отказать
2	Да	Да					Научный сотрудник
3	Да	Нет	> =	3,5			Инженер-консруктор
4	Да	Нет	<	3,5	> =		Инженер по эксплуатаци
5	Да	Нет	<	3,5			Отказать

Непрерывная составляющая управляющей части системы состоит из разностных уравнений:

$$w_{ri}[t] = w_{r-1.i}[t] + 1(i_r[t]), \text{ если } i_r[t] \neq 0,\ w_{0i}[0] = 0,$$

при решении u:

$$u_i[t] = w_{ri}[t], \text{ при } r = R$$

и

$$x[t] = u[t],\ u(t) \leq \varepsilon(t);$$

$$\varepsilon = p - z', \tag{4.18}$$

где p — план приема по штатному расписанию; $z' = z$ — информация о состоянии объекта управления (количестве работающих).

Описание объекта управления имеет вид разностного уравнения

$$z(t) = z(t-1) + [t](x[t] - y[t]). \tag{4.19}$$

Наличие претендентов в цикле $[t]$ характеризуется списком штатного персонала (табл. 4.7 — статус «претендует»). Уволенные определяются в режиме диалога в конце каждого цикла $[t]$, кроме $[t] = 3$, заменой статуса с «работает» на «уволен».

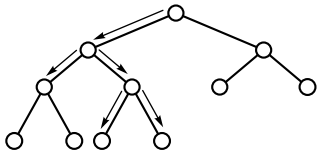
Заметим, что операции (4.18) и (4.19) удобно реализовать в рамках операций агрегирования базы данных. Процесс работы пользователя с реализованной на компьютере ЭСРВ представлен на рис. 4.13.

Реализация системы проведена на основе СУБД InterBase в среде Delphi в полном режиме клиент-сервер и рассмотрена ниже.

Выполнение (реализация). При реализации прежде всего возникает вопрос выбора метода поиска. Известны следующие методы.

Метод поиска в глубину осуществляется так же, как в языке ПРОЛОГ (рис. 4.14). На начальных этапах развития экспертных систем

Рис. 4.14. Поиск в глубину



основной целью являлась проверка работоспособности предлагаемых алгоритмов. В силу этого рассматривались примеры с небольшим количеством данных, которые чаще всего вводились в компьютер в режиме диалога. В этих обстоятельствах был предложен достаточно простой метод поиска в виде поиска в глубину.

Алгоритм прост в реализации, однако является переборным и поэтому работает медленно. В то же время при значительном количестве правил быстродействие становится важным параметром. Выявились затруднения и с процессом объяснений. Данное обстоятельство вынудило искать другие методы поиска. Таким методом является обратный поиск (рис. 4.15), как в GURU.

Этот поиск имеет две разновидности: прямой поиск (от данных к цели) и обратный поиск (по предъявленной цели определяют данные). Во второй разновидности выделяют прямую (последовательный перебор правил) и обратную (направленный перебор правил) аргументацию.

Алгоритм обратной аргументации имеет вид.

Ш а г 1. Находится (первое) правило, где после слова ТО имеется цель.

Ш а г 2. Анализируются посылки после слова ЕСЛИ. Если все конкретные значения переменных имеются, то правило срабатывает. Переход к шагу 5.

Ш а г 3. Если у какой-либо переменной конкретного значения нет, то эта переменная становится временной целью и для нее рассматривается предыдущая процедура до тех пор, пока не получится конкретное значение. Переход к шагу 2.

Ш а г 4. Если исходная цель в данном правиле недостижима, то переходят к следующему правилу, где имеется заданная цель (переход к шагу 1).

Шаг 5. Завершение алгоритма.

Здесь можно изменить порядок рассмотрения правил путем задания приоритета (целое число от 0 до 100) или цены (целое число от 100 до 0).

Такой поиск позволяет сопровождать описание данных словесным представлением правил, что резко упрощает реализацию процедуры объяснения, и расчетами достоверности результатов.

Рис. 4.15. Поиск в базе данных

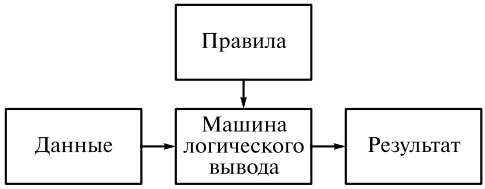


Рис. 4.16. Поиск с помощью баз данных

В этом методе снова предполагалась работа с одиночными объектами. В то же время для практических целей потребовалась работа со значительным количеством объектов, что могло быть обеспечено лишь использованием баз данных с достаточно сложной структурой (рис. 4.16).

В этом методе используются возможности баз данных. Здесь поиск определяется запросом, первоначально формируемым вручную, а затем вызываемым через интерфейс пользователя. Поиск в экспертной системе (см. рис. 4.16) проводится путем сравнения данных в базе правил и базе данных, при этом результат может быть определен с помощью алгоритмов системы управления баз данных и записан в базу данных. В базе данных может быть размещена и словесная формулировка правил. Полученный результат может быть использован в разностном уравнении непрерывной составляющей ЭСРВ.

Правила предоставляются в виде таблиц базы данных. Пример реализации данного метода показан на рис. 4.17. База правил может быть одноуровневой (рис. 4.17, *а*) или многоуровневой (рис. 4.17, *б*).

Приведенный метод позволяет работать практически с любым количеством объектов и правил. Использование отработанных СУБД дает возможность увеличить надежность работы программного продукта.

В силу большого числа исходных данных остановимся на варианте, приведенном на рис. 4.17, *а*. Он является относительно новым и не ограничен размерностью решаемой задачи. В силу простоты правил выберем одноуровневую структуру базы правил.

Отладка и тестирование. Результат работы на этом этапе представлен в виде программы. В силу значительного числа комментариев ее чтение не представляет труда. Обратим внимание лишь на реализацию поиска в соответствии с рис. 4.16.

```
BEGIN
/* Pravilo4*/
FOR SELECT
STEPEN,
OTKRITIE,
STAG,
```

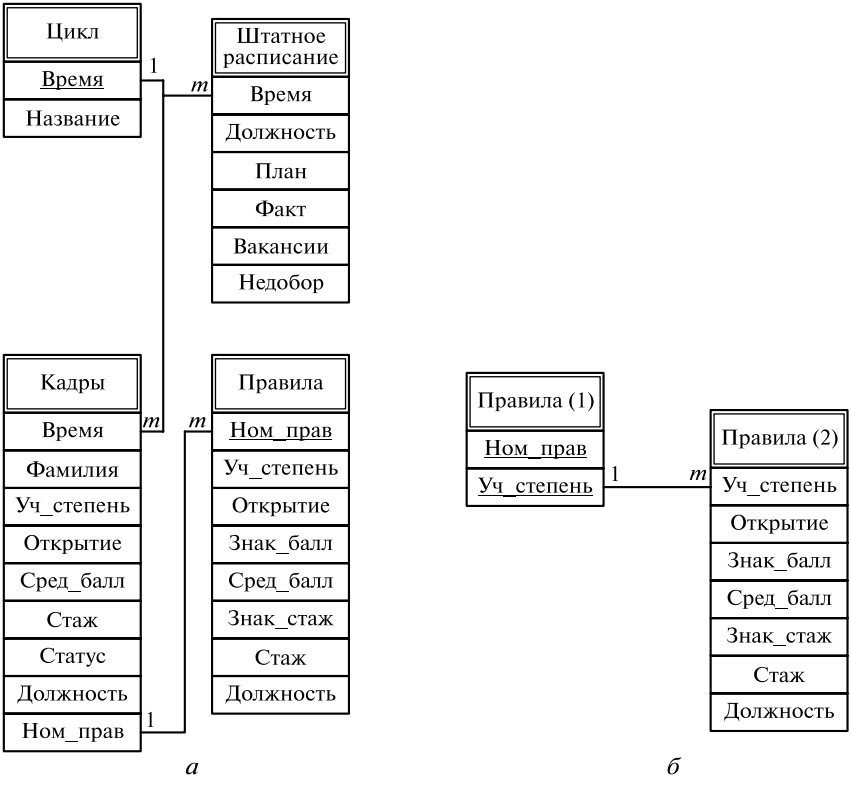


Рис. 4.17. Схема связей ЭСРВ «Прием на работу» при одноуровневой (*а*) и многоуровневой (*б*) структуре правил

```
AVG_BAL,
DOLGNOST
    FROM RULE
      WHERE NBR_RULE = 4
INTO :DSTEPEN, :DOTKRITIE, :DSTAG, :DAVG_BAL,
:DDOLGNOST
    DO
      BEGIN
       UPDATE KADRY
       SET NBR_RULE = 4,
           DOLGNOST = :DDOLGNOST
            WHERE STATUS = 'прете'
              AND STEPEN = :DSTEPEN
              AND OTKRITIE = :DOTKRITIE
              AND STAG>= :DSTAG
```

```
        AND AVG_BAL<:DAVG_BAL
        AND VREM = :IN_VREM;
    END
```

Нетрудно заметить, что логический вывод осуществляется командой select, в которой сравниваются поля базы правил и базы знаний.

Опытная эксплуатация. Эксплуатация осуществляется в соответствии со следующей инструкцией.

Работа с системой носит циклический характер, определяемый интервалами (циклами) времени (от 1 до *T*). В головном меню имеется три элемента: **Главная/Показать/Обработать.** Каждый из этих элементов имеет подчиненные элементы, при «кликании» мышью по которым начинаются определенные действия в работе базы знаний.

В дальнейшем выражение вида **Обработать/Начать** означают обращение к элементу головного меню **Обработать** и подчиненного ему элемента **Начать**.

Замечание. Если при выполнении работы вы допустили ошибку в любом месте выполнения данного программного продукта, обратитесь в Delphi к элементу меню Run/Programm Reset или элементу меню приложения **Главная/Выход**.

При новом запуске программа будет выполнена с самого начала с теми же исходными данными.

1. Запустить exe-файл или программу Delphi. Появляется меню (рис. 4.18).

2. Устанавливается общее количество интервалов времени — количество циклов (рис. 4.19, 4.20). Текущим становится первый интервал времени (цикл). Доступен элемент меню (ЭМ) **Показать/Обработать.**

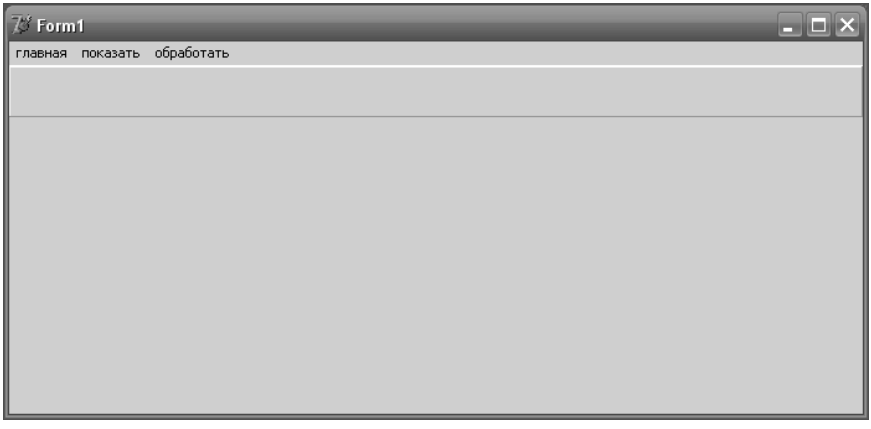
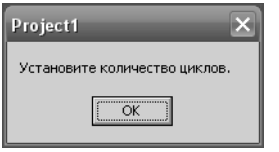


Рис. 4.18. Меню системы приема на работу

Рис. 4.19. Указание на установку количества циклов



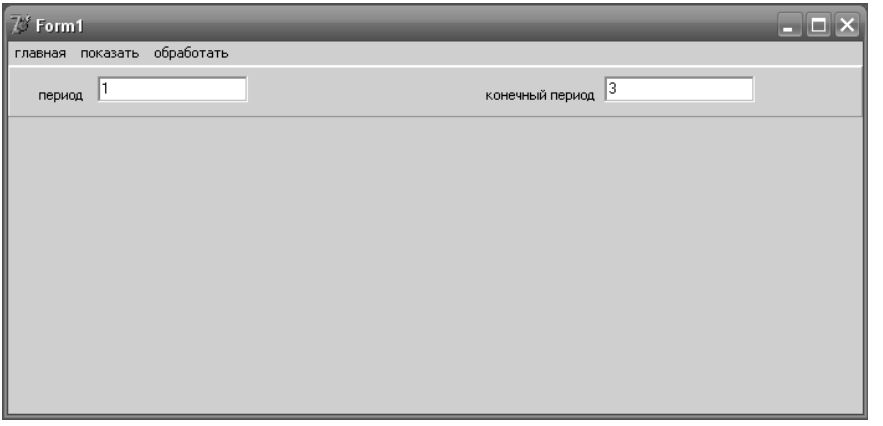


Рис. 4.20. Установка количества циклов

3. После установки закрывается доступ к ЭМ **Показать. Обработать** и становится доступным ЭМ **Показать/Просмотр/Штатное расписание** (рис. 4.21), с помощью которого вызывается соответствующая таблица. После закрытия этой таблицы «открываются» ЭМ **Показать/Кадры**, **Показать/Работающие**, **Показать/Претенденты**, выдающие на экран монитора соответствующие таблицы

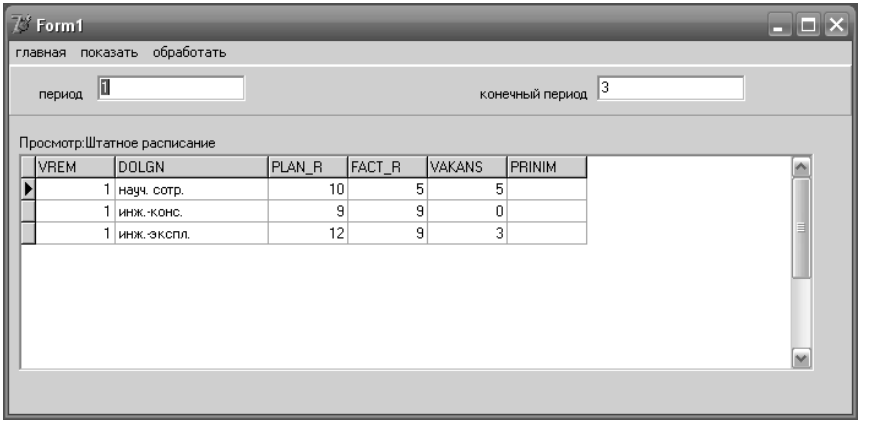


Рис. 4.21. Просмотр штатного расписания

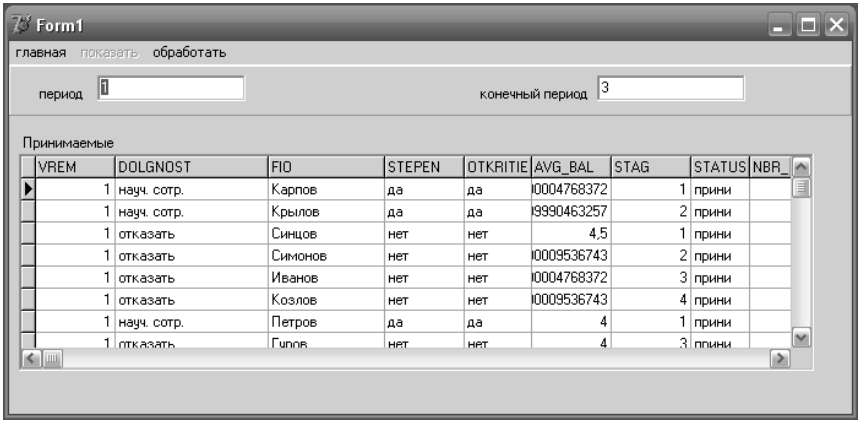


Рис. 4.22. Таблица **Принимаемые**

для последовательного просмотра пользователем. При просмотре таблицы **Претенденты** на экране появляется кнопка «Далее». Если к ней не обращаться, то можно вернуться к новому просмотру всех перечисленных таблиц.

4. Нажатие кнопки «Далее» прекращает доступ к перечисленным ранее ЭМ. Открывается доступ к ЭМ **Обработать/Запуск правил**. Результатом работы правил является выработка решений-советов для ЛПР о принятии претендентов на работу. Статус **Претенденты** заменяется на **Принимаемые** (рис. 4.22). Одновременно подсчитывается количество принимаемых по каждой должности. ЭМ **Обработать/Запуск правил** становится недоступным.

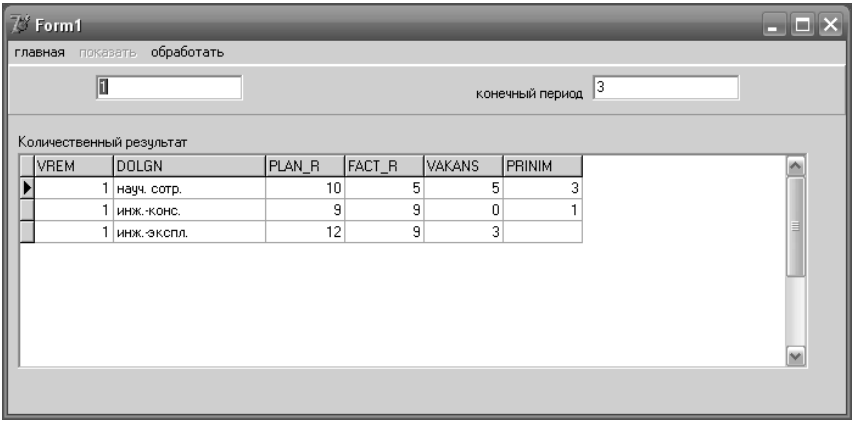


Рис. 4.23. Результат работы правил

5. Список принимаемых в виде таблицы возможно получить, нажав ЭМ **Показать/Принимаемые**, при этом будут указаны и должности «отказать».

6. После просмотра таблицы (закрытия формы) закрывается доступ к ЭМ **Показать/Принимаемые** и доступным становится ЭМ **Обработать/Количественный результат** (рис. 4.23). На экран вызывается таблица **Штатное расписание**, в которой указано количество вакансий и принимаемых. Одновременно открывается доступ к ЭМ **Обработать/Объяснения**.

7. Если какие-либо результаты работы компьютера вызывают у пользователя сомнения, то следует нажать ЭМ **Обработать/Объяснения**. Получается словесное описание правила, с помощью которого получен результат для какой-либо записи. Появляется кнопка «Далее».

8. При нажатии кнопки «Далее» может быть два исхода.

9. Если принимаемых больше вакансий, то переход к следующему этапу работы программного продукта невозможен. При этом выдается дополнительное сообщение, предлагающее пользователю изменить либо правила, либо результат. Становятся доступными ЭМ **Обработка/Корректировка правил**, **Обработка/Корректировка результата**.

10. После изменения правил (**Обработка/Корректировка правил**) происходит возврат к ЭМ **Обработка/Запуск правил** и пп. 4 — 9 могут повториться.

11. После изменения результатов (**Обработка/Корректировка результата** заменой какой-либо должности на «отказать») доступными остаются ЭМ **Обработка/Количественный результат**, **Обработка/Изменение результатов**.

12. Если число принимаемых не превышает количества вакансий, то, в зависимости от поставленной цели, возможно (хотя и не обязательно) обращение к ЭМ **Обработка/Изменение результатов**. Затем при нажатии кнопки «Далее» закрывается доступ к ЭМ пп. 6 — 9 и становится доступным ЭМ **Показать/Принятые**. Записи с должностями «отказать» уничтожаются, поскольку считается, что вторично лица, получившие отказ, не попадают в число претендентов. Статус «принимаемые» меняется на «принятые». Одновременно корректируется штатное расписание. На экране появляется список принятых.

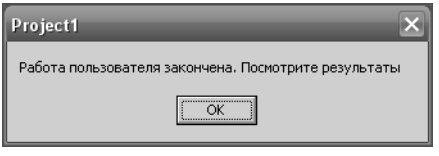


Рис. 4.24. Завершение работы

13. Если он не вызывает сомнений, он закрывается, становится доступным ЭМ **Обработка/Принять всех**. Следует его нажать. При этом статус «принятые» сменится на «работающие». На этом заканчивается очередной цикл работы в данном сеансе. Делается доступным ЭМ **Обработка/Продолжить**.

14. Нажатие его может вызвать два результата.

15. Если номер текущего цикла равен количеству циклов, то работа программы заканчивается, пользователю выдается сообщение об этом (рис. 4.24).

16. Если этот цикл не является последним, то во всех таблицах интервал времени (цикл) автоматически меняется на следующий и появляется ЭМ **Показать/Уволенные**.

17. При его нажатии первоначально вызывается таблица Кадры (работающие), в которой вручную (по указанию преподавателя) устанавливается статус «уволен» для уволившихся специалистов. Список-таблицу уволенных (до 20 % состава) можно просмотреть. После его закрытия в таблицу Кадры добавляется новый список претендентов и проводится переход к ЭМ **Показать/Штатное расписание**. Затем процедуры, начиная с п. 3, повторяются.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Перечислите методы описания знаний, логического вывода и дайте их характеристику.
2. Какие требования предъявляют к методам описания знаний и логического вывода?
3. Что такое алгебра, исчисление, предикат, предикат первого порядка?
4. Что такое предпосылка, цель, предложение? Какие существуют синонимы этих понятий?
5. Что такое интерпретация?
6. Что такое общезначимость, противоречивость формулы? Дайте понятие фразы Хорна. Почему для доказательства выводимости чаще используют противоречивость формулы?
7. Что такое сколемизация?
8. Что такое семантическое дерево?
9. Опишите метод резолюций Робинсона, дайте определение понятия «резольвирование».
10. Как теоретически получить результат вывода?
11. Что такое фактор уверенности?
12. Назовите алгебры обработки факторов уверенности.
13. Что такое нечеткая переменная?
14. Назовите правила обработки нечетких переменных.
15. Перечислите методы представления знаний. Опишите суть методов.
16. Каково соотношение теории предикатов первого порядка, традиционной логики и булевой алгебры?
17. В чем различие процедурных и декларативных языков?

18. Каково назначение языка ПРОЛОГ? В каких сферах его применяют?
19. Каковы достоинства и ограничения языка ПРОЛОГ?
20. Что такое унификация и консультация?
21. Каково назначение пакета GURU? В каких сферах его применяют?
22. Назовите достоинства и недостатки пакета GURU.
23. Какими способами можно осуществить связь с пакетом GURU?
24. Какие функции выполняет ЭСРВ «Прием на работу»?
25. Какой метод поиска используется в ЭСРВ «Прием на работу»?
26. Как представляются правила в ЭСРВ «Прием на работу»?
27. Какими программными средствами реализована ЭСРВ «Прием на работу»?

Глава 5

ТЕХНОЛОГИЯ СОЗДАНИЯ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ И МНОГОАГЕНТНЫХ СИСТЕМ

5.1. Технология создания искусственных нейронных сетей (ИНС)

Вначале рассмотрим распознавание образов, а потом сосредоточим свое внимание на области управления.

В распознавании образов различают линейное (рис. 5.1) и нелинейное (рис. 5.2) разделение объектов (образов), в частности, в двумерной системе координат x_1, x_2.

Рассмотрим линейный случай в одномерной системе координат.

Имеются два вида элементов: «крестики» (×) и «нолики» (0). Необходимо их разделить на области I — III, а затем области I и III объединить. Обсудим первую часть задачи.

Очевидно, что задача может быть решена, если удастся сформировать некоторую ступенчатую кривую *a* (рис. 5.3) с соответствующими параметрами. Необходимость многослойной схемы наиболее четко видна из этой задачи распознавания образов.

Для этого формируется сеть (рис. 5.4), на которой x_i — соответствующие элементы, *1* — дуга со значением 1. Элемент x_0 обладает порогом w_0, элемент x_1 копирует элемент x_0, в элементе x_2 вес $w_{12} < 0$, а $w_{13} > 0$. В элементах x_2 и x_3 используется сигмоидальная функция, элемент x_4 суммирует сигналы x_2 и x_3 с соответствующими весами. Покажем, что зависимость y близка к ранее отображенной функции.

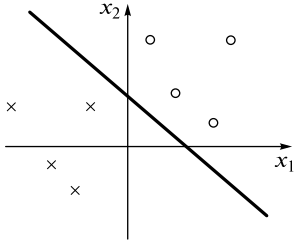


Рис. 5.1. Линейное разделение образов

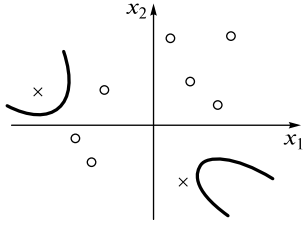


Рис. 5.2. Нелинейное разделение образов

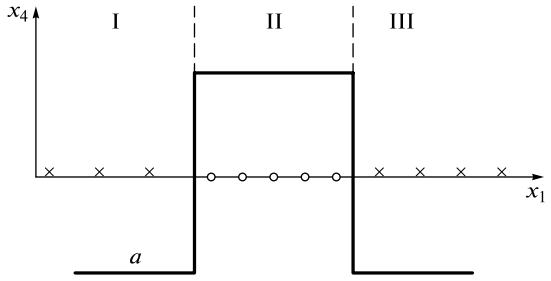


Рис. 5.3. Иллюстрация применения ИНС для распознавания образов

Запишем математические зависимости:

$$x_2 = f\left(w_{12}x_1 + w_{02}x_0\right);$$

$$x_3 = f\left(w_{13}x_1 + w_{02}x_0\right);$$

$$x_4 = w_{24}x_2 + w_{34}x_3 + w_{04}.$$

Используем трансформацию сигмоидальной функции в зависимости от весов, смещения и получим результат путем суммирования кривых x_2 и x_3 (рис. 5.5).

Следовательно, за счет весов с соответствующими величинами смещений формируется кривая x_4, которая при подборе соответствующих числовых значений может быть близкой к кривой на рис. 5.3. Таким образом, для решения данной задачи потребовалась многослойная сеть.

Нелинейное разделение (см. рис. 5.2) достигается с помощью специфической сети — радиальной базисной функции. Доказано, что при наборе образов **X** во входном пространстве размерности N можно найти такое нелинейное отображение $\varphi(\mathbf{X})$ размерности M, что образы $\varphi(\mathbf{X})$ будут разделены линейно.

Оказалось, что многослойные, многомерные сети могут решать разнообразные задачи. В дальнейшем будем рассматривать как одномерные и многомерные, так и однослойные и многослойные сети.

Далее перейдем к использованию ИНС в области управления. Заметим, что здесь возможны два варианта:

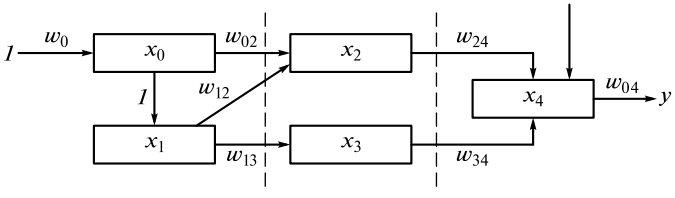


Рис. 5.4. Пример многослойной ИН

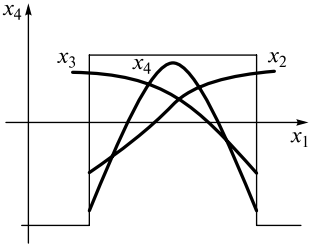


Рис. 5.5. Процесс распознавания образов

1) ИНС в процессе эксплуатации имеет разомкнутую структуру (например, ADALINE, MADALINE) и замыкается лишь в процессе обучения;

2) ИНС замкнута и в процессе эксплуатации (например, сеть Хопфилда), и в процессе обучения.

В настоящее время в большинстве случаев используют первый вариант, поскольку во втором варианте возникают сложные проблемы устойчивости и качества ИНС в процессе эксплуатации.

Вместе с тем сеть Хопфилда специфична и обладает свойством устойчивости. Покажем это.

Поскольку сеть Хопфилда (рис. 5.6) является замкнутой, возникает вопрос о ее устойчивости. На рис. 5.6 обозначение z^{-1} — элемент запаздывания на один такт. Далее дискретные моменты времени будем обозначать (t), а интервалы времени — $[t]$ или Δt.

Введем производные для сигналов s_i, тогда сеть Хопфилда может быть записана следующими выражениями:

$$q_i(t+1) = f(s'(t)); \tag{5.1}$$

$$s_i'(t) = \sum_{i=1,\, i\neq j}^{n} w_{ij} q_i(t) + r_i w_{0i}. \tag{5.2}$$

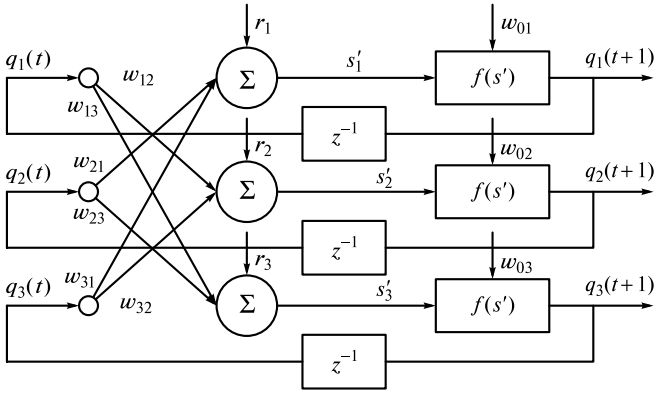


Рис. 5.6. Сеть Хопфилда

Подобные выражения могут быть записаны и в векторно-матричной форме. Часто выражения (5.1) и (5.2) записывают в немного иной форме:

$$q_i(t+1) = f(s'(t));$$

$$Ts'_i(t) = -\sum_{j=1,\, i\neq j}^{m} w_{ij} q_i(t) + r_i w_{0i},$$

где T — некоторая постоянная времени, характеризующая длину.

Понятие «устойчивость» связано с функцией Ляпунова. Для проверки устойчивости сети Хопфилда сформируем функцию Ляпунова L, а затем возьмем ее производную $\dot{L}$. Доказано, что если $\dot{L}$ имеет знак, противоположный L, то система устойчива.

Формирование функции L в значительной степени неформально. В качестве неформальной основы используют правила Хеба.

Правила Хеба формируются следующим образом.

1. Если два связанных нейрона возбуждаются вместе, то сила связи между ними (веса w) возрастает.

2. Если два связанных нейрона возбуждаются порознь, то сила связи между ними (веса w) уменьшается.

Представим себе, что время непрерывно.

На основании приведенных суждений предложена следующая функция Ляпунова:

$$L = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij} q_i q_j - \sum_{i=1}^{n} r_i q_i;$$

$$\dot{L}_i = \frac{dL}{dt} = \frac{\partial L}{\partial q_i}\cdot\frac{\partial q_i}{dt} = \frac{\partial L_i}{\partial q_i}\cdot q(t);$$

$$q_i(t+1) = f(s'(t)),$$

$$Ts'_i(t) = \sum_{j=1,\, i\neq j}^{m} w_{ij} q_i(t) + r_i w_{0i};$$

$$\frac{\partial L_i}{\partial q_i} = \sum_{i=1}^{m} w_{ij} q_i(t) - r_i w_{0i};$$

$$Tq(t) = Tf'_s S' \Rightarrow T\,\dot{q}(t) = f'_s\left(\sum_{j=1}^{m} w_{ij} q_i(t) + r_i w_{0i}\right) = -f'_s\left(\frac{\partial L_i}{\partial q_i}\right).$$

Постоянная величина смещения w_{01} в динамических выражениях не учитывается:

$$\dot{L} = \frac{\partial L_i}{\partial q_i} \cdot \frac{1}{T}\left(-f_s'\left(\frac{\partial L_i}{\partial q_i} \right) \right) < 0.$$

Обычно f_s' — величина положительная, и тогда выражение отрицательно. L и $\dot{L}$ имеют разные знаки и, следовательно, сеть Хопфилда в процессе эксплуатации устойчива.

С позиций предметной области управления возможны следующие варианты использования ИНС:

1) проектирование с выбором функции активации f в соответствии с поставленной целью;

2) задание алгоритма обучения сети с помощью подбора соответствующих весов;

3) запоминание (в процессе работы): при зафиксированных весах и функции активации входной вектор r запоминается в виде выхода q.

В последнем случае говорят, что ИНС имеет ассоциативную память. В двух последних случаях ИНС рассматривается как замкнутая. Из-за большого количества нелинейности и многослойности может быть много состояний равновесия.

Каждое из таких состояний (аттрактор) фактически соответствует незашумленному входному сигналу r. Наличие аттракторов позволяет выделять сигналы на фоне помех. Пусть на вход подается сигнал r, искаженный помехами. Когда сеть входит в соответствующий аттрактор, то на выходе сигнал q соответствует входному сигналу r без искажений.

ИНС требуется обучать. В процессе управления фактически возникают две процедуры: обучение и управление (в процессе эксплуатации системы).

Обучение происходит итеративно, поэтому единицей измерения времени служит итерация. Предпочтительно, чтобы вначале шло обучение, а затем — управление, хотя часто процедуры приходится совмещать. Теория таких совмещенных процессов достаточно сложна.

Выделяют теорию обучения однослойных и многослойных сетей. Первоначально развивалась теория для однослойных сетей. Для теории расчетов обоих видов сетей характерно большое количество неформальных моментов.

Методы обучения делятся на группы:

- по эталону r^* (с учителем);
- без учителя, с помощью оптимизации;
- смешанные (разные коэффициенты настраиваются по разным правилам).

В настоящее время чаще всего используют методы первой группы.

Схема режима обучения с учителем в общем виде может быть представлена рис. 5.7.

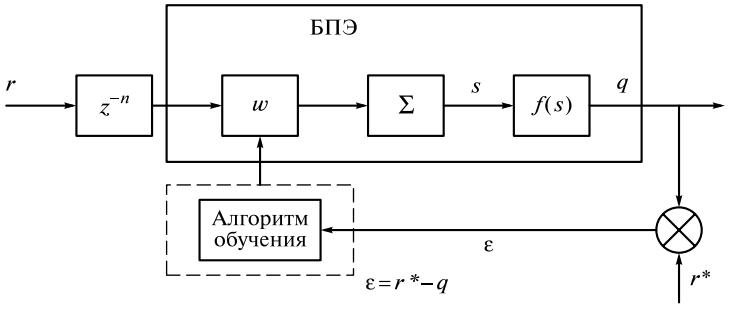


Рис. 5.7. Общая схема обучения ИНС:

z^{-n} — запаздывание на n интервалов времени; w — вес; $f(s)$ — сигмоидальная функция; ε — отклонение от эталона; s — сумма; q — выход

В ряде случаев при обучении вводят звено запаздывания z^{-n}, где n — количество дискретных интервалов времени.

В процессе обучения имеется много неформальных моментов:

- при реализации алгоритма обучения вводят понятия «интегрирующее звено» и «инерционное звено с постоянной времени T». Выбор величины T — это выбор между двумя крайностями. При большой величине T обучение происходит медленно, при малой величине T система может оказаться неустойчивой;
- величину T берут достаточно малой и для повышения устойчивости часто вводят еще одно корректирующее звено. В общем случае считается, что сеть многослойная с числом слоев k. По умолчанию изменению подвергаются соответствующие веса w, а изменение наклона логистической функции активации проводится в процессе проектирования;
- для оценки процедуры обучения (настройки ИНС) вводятся некоторые функции: $V(r)$ или $V(w)$. Выбор функции — процедура неформальная. Производится оптимизация этой функции, чаще всего максимизация:

$$V^*(r) = \max_r V(r).$$

Вводится понятие «сила» — $\dfrac{\partial V}{\partial r} = F_r$.

Поскольку речь идет о максимизации, считается, что со временем функция V возрастает и

$$\frac{\mathrm{d}V}{\mathrm{d}t} = \frac{\mathrm{d}V}{\mathrm{d}r} \cdot \frac{\mathrm{d}r}{\mathrm{d}t} > 0.$$

Последнее выражение справедливо, если

$$\operatorname{sign} F_r = \operatorname{sign}\left(\frac{\mathrm{d}r}{\mathrm{d}t}\right).$$

Рассмотрим методы обучения одно- и многослойных ИНС.

Выделяют следующие конкретные методы:

- правило Хеба;
- дельта-метод (δ-метод);
- обучение с конкуренцией (используется δ-метод, но между БПЭ и слоем устанавливается состязание по принципу: «Победитель получает все»).

Возможны два варианта рассмотрения обучения:

1) количество слоев и весов на каждом уровне задано. В этом случае используют δ-метод;

2) количество слоев и весов требуется найти. Здесь можно использовать генетические алгоритмы, как это показано ранее.

Предположим, что количество слоев и элементов в каждом слое задано. Обучение идет в два этапа: обучается последний слой, обучение переносится на предпоследний слой и т. д.

Начнем с последнего слоя: δ-метод предполагает наличие учителя (в данном примере q_j^*). Предполагается, что обучение ведется за счет изменения весов (кроме пороговых весов).

Пусть

$$V = \sum_{j=1}^{m} \left(q_j - q_j^*\right)^2 w_{ij} \to \min.$$

Найдем значение силы

$$F_w = \frac{\partial V}{\partial q_i} \cdot \frac{\partial q_i}{\partial s_i} \cdot \frac{\partial s_i}{\partial w_{ij}};$$

$$\delta = q_j - q_j^*.$$

В качестве вспомогательного элемента используем

$$T_w \frac{\mathrm{d}q}{\mathrm{d}t} = \delta f_s' q_j. \tag{5.3}$$

Практика показывает, что величина $\varphi = f_s' q_j$ значительно влияет на устойчивость обучения. В связи с этим, наряду с элементом (5.3), используют еще один корректирующий элемент, выбор которого определяется степенью устойчивости. Выражение $\frac{\mathrm{d}w}{\mathrm{d}t}$ не подходит для цифровых расчетов, поэтому его заменяют разностью

$$T_w \left(w(t + \Delta t) - w(t)\right) = \delta f_s' q_j.$$

Выбор интервала Δt — процедура неформальная.

Перейдем к предыдущему слою. Обозначим i — предыдущий слой, j — последующий слой. Для перехода к предыдущему слою используется так называемый закон передачи силы:

$$F_i q_i = F_j q_j;$$
$$\frac{F_i}{F_j} = \frac{q_j}{q_i} = k_{ij}.$$

Величина k_{ij} задается неформально. Часто задается и q_i. В этом случае имеются данные для слоя i, который рассчитывается так же, как и последующий слой.

Более детально схема обучения показана на рис. 5.8.

Следует отметить, что при обучении [29] используются обучающие и тестовые (не входящие в обучающие) данные.

Обобщение — способность сети корректно осуществлять преобразование вход-выход для обоих видов данных, т.е. способность к интерполяции сходных данных.

Наилучшим вариантом обучения является случай, показанный на рис. 5.9. Однако часто обобщение не имеет места (рис. 5.10). Отсутствие способности к обобщению называют *переобучением*.

Выделяют ошибки обучения и ошибки тестирования (рис. 5.11). Точка *A* является моментом остановки обучения: ошибка тестирования начинает расти при уменьшении ошибки обучения. Если ошибка тестирования начинает расти, а ошибка обучения не достигла заданной величины, то сеть называется *недообученной*. Это свидетельствует о неправильно выбранных архитектуре и параметрах.

Условиями существования обобщения сети являются гладкость нелинейного преобразования сети и представительность набора данных. Время обработки не должно быть очень большим, однако набор должен учитывать все особенности решаемой задачи.

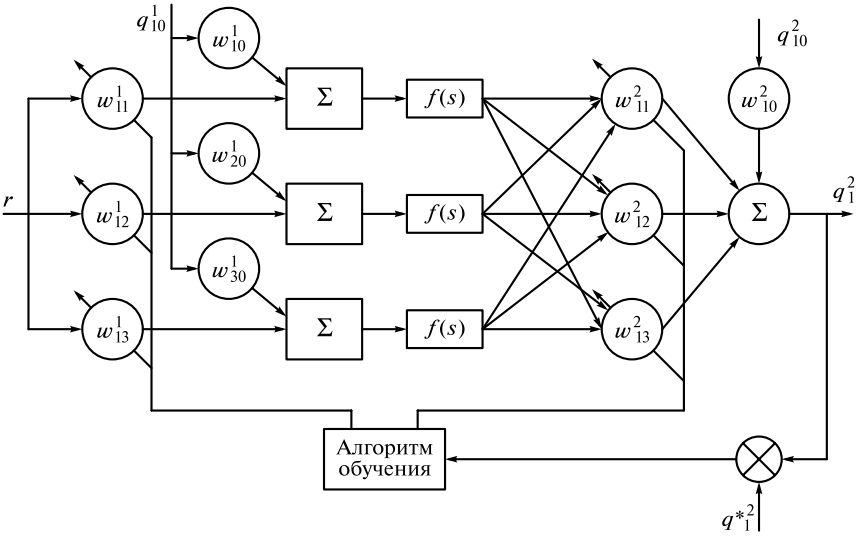


Рис. 5.8. Детальная схема обучения ИНС

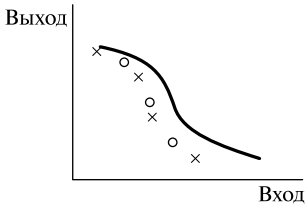


Рис. 5.9. Успешное обучение:
x — обучающие; o — тестовые

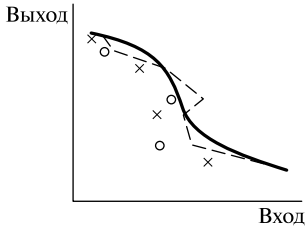


Рис. 5.10. Отсутствие обобщения при обучении:
x — обучающие; o — тестовые

Если при обучении невозможно сформулировать схему с учителем, то для обучения используют метод самоорганизации с использованием сети Кохонена. В этом случае зависимости между элементами многомерных данных стараются отобразить в простых топологических отношениях в пространстве меньшей размерности, предпочтительнее — в двумерном или одномерном.

Весовые коэффициенты настраиваются так, чтобы близко расположенные друг к другу нейроны выходного слоя активизировались при подаче на вход сходных входных сигналов. По этому принципу работает человеческий мозг, в котором есть области, отвечающие за движения, речь и т.д.

Как уже отмечалось, в ИНС при обучении, а также при управлении изменяются веса. Реализация такого изменения осуществляется с помощью множительных устройств. Такие элементы по своей сути нелинейны. Фактически в ИНС происходит не сигнальная настройка, а параметрическая. Такой режим очень сложно описать математически.

В настоящее время программный продукт реализации ИНС может быть составной частью другого продукта, в частности пакета MatLab, либо быть самостоятельным.

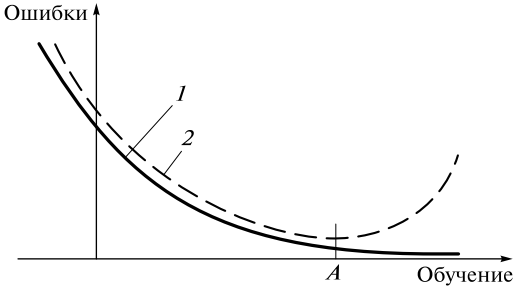


Рис. 5.11. Соотношение ошибок обучения (*1*) и тестирования (*2*)

5.2. Технология построения многоагентных систем

Основными направлениями научного поиска в области мультиагентных систем являются следующие:

- теории агентов (agent theories), которые рассматривают математические методы и формализмы абстрактного представления структуры и свойств агентов и способы построения рассуждений (логических выводов) в таких формальных системах;
- методы коллективного поведения агентов;
- архитектуры агентов и МАС;
- методы, языки и средства коммуникации агентов;
- языки программирования агентов;
- методы и средства автоматизированного проектирования МАС;
- методы и средства обеспечения мобильности агентов.

Трудности формального представления таких интенсиональных (ментальных) понятий (intentional notions), как убеждения, намерения, желания, были показаны М. Вулдриджем и Н. Дженнингсом (M. Y. Wooldridge, N. R. Jennings). В разработке логических формализмов ментальных понятий возникают две основные проблемы: описание синтаксиса языка формализации и создание его семантической модели.

Синтаксис описывают, используя два фундаментальных подхода:

1) применение модальных логических систем, содержащих не истинностные модальные операторы;

2) применение метаязыков, основанных на многосортной логике первого порядка, содержащей термины, которые определяют формулы некоторого другого объектного языка.

Формализация знаний и убеждений в модальных логических системах приводит к тому, что агенту должны быть известны все общезначимые формулы (пропозициональные тавтологии), а поскольку количество таких формул бесконечно, то такие модели оказываются неприменимыми в реальных технических системах с ограниченными ресурсами.

Использование метаязыков позволяет представить отношения между терминами метаязыка, определяющего агента, и терминами объектного языка, определяющего некоторые формулы.

Недостаток формализма метаязыков — противоречивость некоторых из них, тем не менее существуют и достаточно успешные метаязыковые конструкции.

Модели возможных миров для логики знаний и убеждений опираются на основополагающие работы Я. Хинтикки (J. Hintikka) и формулировку нормальной модальной логики, данную С. Крипке (S. Kripke). Поскольку такие семантические теории агентов приводят к серьезным внутренним общезначимым проблемам, то исследуются и альтернативные подходы. Среди них следует назвать теорию наме-

рений Коэна и Левескью (Cohen, Levesque); модель Рао и Георгиева (Rao & Georgeff), рассматривающую три примитивные модальности (убеждение, желание и намерение); семейство логик Синга (Singh) для представления намерений, убеждений, знаний и коммуникаций агентов в сети с передачей временного управления.

Сложилась традиция выделения трех базовых классов архитектур агентных систем и соответствующих им моделей интеллектуальных агентов (табл. 5.1): делиберативные; реактивные; гибридные.

Таблица 5.1. **Свойства моделей ИА в МАС**

Авторы	Достоинства	Недостатки
	Делиберативные модели	
М. Дж. Уолдридж	Логическая полнота модели. Возможность применения различных внутренних логических языков	Отсутствует явное понятие цели. Поведение определяется алгоритмом «цикла действий»
М. Дж. Уолдридж, М. Д. Инверно, Дж. Кинни, М. Люк	Логическая полнота модели. Наличие точной формальной семантики. Наличие библиотеки планов, обеспечивающих вариативность поведения ИА	Цель определяется только логическими средствами. Не рассматривается понятие мультиагентного мира, не определяется, с кем и как взаимодействует агент. Отсутствует иерархия ИА
Б. Ван Линдер, В. Ван Нок, Дж. Мейер	Формальная строгость модели. Формализация достаточно трудноопределимых понятий, таких как предпочтения и обязательства	Высокая сложность формальной модели. Специализированная теоретическая направленность. Отсутствует конструктивное определение цели. Отсутствует модель поиска решения
	Реактивные модели	
К. Сентарвич, Е. Навареский, М. Забинский	Предусматривается мобильность агентов. Предполагается множество стратегий. Предусматривается возможность адаптации к условиям среды через операцию обучения	Отсутствует иерархия ИА. Связь агента и пространства E точно не определена. Не определяются логические взаимоотношения агентов и отсутствует логический вывод. Поведение агента точно не определено. Не учи-

Окончание табл. 5.1

Авторы	Достоинства	Недостатки
		тывается предыстория системы. Не предлагается конструктивного механизма реализации стратегий
Дж. Фербер, О. Лаббани, Дж.-Р. Маллер, А. Буржолт	Относительная простота модели	Отсутствует понятие цели, модель носит исключительно реактивный характер «стимул — реакция». Поведение агента точно не определено
Дж. Фербер, О. Лаббани, Дж.-Р. Маллер, А. Буржолт	Относительная простота модели	Отсутствует понятие цели, модель носит исключительно реактивный характер «стимул — реакция». Поведение агента точно не определено
В. В. Денисов, Д. В. Пузанков, М. Г. Пантелеев, Г. Г. Колосов, М. А. Натей-Голенко	Оригинальная и глубоко проработанная модель реактивного ИА. Наличие системы скелетных планов, обеспечивающих приемлемое поведение в реальном масштабе времени	Отсутствует иерархия ИА. Отсутствует иерархия целей
Гибридные модели		
С. Амброзкевич, С. Билка, Ж. Комар	Относительная простота формирования целей	Цели являются статическими и предустановленными. Нет иерархии целей. Отсутствует конструктивное определение окрестности цели, не ясно, от каких агентов цель будет зависеть, от каких нет
К. Фишер, Ж. П. Мюллер, М. Пишелл	Наличие иерархии агентов	Отсутствие строгой формальной модели

Реактивный подход позволяет эффективно использовать множество довольно простых сценариев поведения агентов в рамках уста-

новленных реакций на определенные события окружающей среды, но его ограниченность проявляется в практической невозможности полного ситуативного анализа всех возможных активностей агентов.

Делиберативные архитектуры и модели. Делиберативную архитектуру принято определять как архитектуру агентов, содержащих точную символическую модель мира и принимающих решения на основе логического вывода. Теоретическим основанием для построения моделей делиберативных агентов послужила гипотеза физических символьных систем, сформулированная Г. Л. Ньювеллом и Г. А. Саймоном. Физическая символьная система по определению должна быть физически реализуемым множеством физических сущностей или символов, которые комбинируются в определенные структуры и способны запускать процессы, оперирующие этими символами в соответствии с символически закодированными множествами инструкций. Данная гипотеза постулирует утверждение о том, что такая система способна на интеллектуальное поведение в достаточно общем понимании этого термина. Согласно трактовке М. Р. Генезерита делиберативный агент должен обладать следующими свойствами:

- содержать эксплицитно представленную базу знаний, заполненную формулами в некотором логическом языке, представляющую его убеждения;
- функционировать в следующем цикле: восприятие обстановки (обсервация) — логический вывод — действие ...;
- принимать решения о действиях на основе некоторых форм логического вывода.

Основы строгой формализации знаний и действий делиберативного агента заложены К. Конолиге, предложившим иерархический метаязык для описания и логического вывода на знаниях и действиях вычислительных агентов. Предполагается, что объектный язык есть стандартный язык первого порядка, а для каждого примитивного выражения объектного языка e предполагается существование соответствующего терма e' в метаязыке. Термы метаязыка, обозначающие составные формулы объектного языка, конструируются с использованием метаязыковых функций *and*, *or*, *not* и т. д.

Для обозначения того, что формула объектного языка отражает истинное состояние мира, К. Конолиге использует истинностный предикат метауровневого языка TRUE, приписывая ему следующие аксиомы:

$$\forall f \cdot \neg TRUE(f) \Leftrightarrow TRUE(not(f))$$

$$\forall f \cdot \forall g \cdot \neg TRUE(f) \vee TRUE(g) \Leftrightarrow TRUE(or(f,g))$$

etc.

К. Конолиге использует синтаксический подход для описания убеждений агентов: каждому агенту назначается множество формул объектного языка (теория данного агента) и убеждения θ, если θ яв-

ляются доказуемыми в этой теории. Функция метауровневого языка *th* задается на множестве термов, определяющих агентов, и возвращает множество формул объектного языка, представляющих теорию агента. Предикат метаязыка $FACT(t, f)$ говорит о том, что f принадлежит теории t. Общий факт есть формула, истинная для всех агентных теорий:

$$\forall f \cdot CFACT(f) \Leftrightarrow \forall a \cdot FACT(th(a)), f \,\&\, TRUE(f).$$

Далее вводится бинарный предикат доказуемости *PR*, использующий в качестве аргументов теорию и формулу. Система аксиом для этого предиката использует аксиоматику объектного языка и поэтому включает в себя такие правила, как модус поненс, рефлексивность и т. д.:

$$\forall t \cdot \forall f \cdot \forall g \cdot PR(t, imp(f, g)) \,\&\, PR(t, f) \Rightarrow PR(t, g)$$

$$\forall t \cdot \forall f \cdot PR(t, f) \Rightarrow PR(t, f)$$

etc.

Убеждения определяются через предикат метаязыка, связывающий агента и формулу объектного языка:

$$\forall a \cdot \forall f \cdot BEL(a, f) \Leftrightarrow PR(th(a), f).$$

Знание определяется как истинное убеждение.

Далее К. Конолиге расширяет свою модель в трех направлениях, вводя:

- функцию стандартных имен η и денотационную функцию Δ для упрощения некоторых синтаксических проблем, связанных с метаязыком;
- вложенные убеждения (т. е. убеждения об убеждениях), расширяя двухязыковую иерархию в трехязыковую иерархию;
- ситуации (в смысле ситуационного исчисления) в область метаязыка для выводов об изменяющемся мире.

Основные проблемы, выявленные в данной модели, состоят в том, что трехуровневая иерархия устанавливается автором произвольно. С вычислительной точки зрения (ad hoc) модель К. Конолиге представляется неуправляемой из-за наличия метауровней, а некоторые аксиомы, предложенные для общих фактов, неверны.

Согласно М. Дж. Уолдриджу интеллектуальный агент определяется как структура:

$$(\Delta_0, \rho, \beta, l, MR, AR),$$

где $\Delta_0 \in Belset$ — начальное множество убеждений; $\rho \subseteq Drule$ — множество правил вывода для L; $\beta \in Brf$ — функция ревизии убеждений; $l \in Mes\sin t$ — функция интерпретации сообщений; $MR \subseteq Mrule$ —

множество правил сообщений; $AR \subseteq Arule$ — множество правил действий; L — внутренний логический язык.

Поведение агента определяется базовым циклом, включающим в себя следующие шаги:

- интерпретацию любых принятых сообщений;
- изменение убеждений через обработку эпистемических входов с учетом предыдущих действий и интерпретации сообщений через функцию ревизии убеждений;
- построение дедуктивного замыкания множества убеждений;
- извлечение множества возможных сообщений, выбор одного из возможных и отправка его получателю;
- извлечение множества возможных действий, выбор одного из них и выполнение этого действия;
- возврат к первому шагу.

В формализованном виде функционирование агента представляется следующими выражениями:

$$Belcet = powerset\ Form(L)$$
$$Close\_Belset \times powerset\ Drule \rightarrow Belset$$
$$Epin = powerset\ Form(L)$$
$$Brf = Belset \times powerset\ Epin \rightarrow Belset$$
$$Action = Belset \rightarrow Epin$$
$$Cond = Form(L) \cup \{true\}$$
$$Arule = Cond \times Action$$
$$Mrule = Cond \times Mess$$

В последнее десятилетие было предложено несколько архитектур для делиберативных агентов. Большинство из них развертывались только в ограниченных искусственных средах, и лишь очень немногие были применены для решения реальных задач. Незначительное количество архитектур было доведено до стадии реальных корпоративных приложений, отлаженных в конкретной предметной области. Одной из таких архитектур является Multi-Agent Reasoning System (dMARS), основанная на более ранней системе Procedural Reasoning System (PRS) и использующая концептуальную основу BDI-модели практического вывода.

Была предложена абстрактная спецификация модели делиберативного агента с расширениями, которые могли бы удовлетворять различным возможным аксиомам BDI-теории. Однако данная спецификация является высокоуровневой абстракцией и не допускает непосредственной технической реализации. Известна формализация делиберативного агента средствами языка AgentSpeak(L). Этот язык представляет собой язык программирования, основанный на абстракции PRS-архитектуры.

Далее была предложена развернутая формальная модель делиберативной архитектуры dMARS, использующая язык *Z*, который является модельно-ориентированным языком формальных специфи-

каций, основанным на теории множеств и логике первого порядка. Ключевыми компонентами *Z*-спецификации являются пространство состояний системы и множество возможных операций, переводящих одно состояние в другое.

Модели агентов в dMARS и PRS являются примерами наиболее популярной в настоящее время парадигмы, известной как Belief, Desire, Intentions или BDI-подход. BDI-архитектура, как правило, содержит четыре ключевые структуры данных: убеждения, цели, намерения и библиотеку планов.

Агентные убеждения соответствуют информации агента о мире и могут быть неполными и некорректными. Обычно агенты в BDI-модели хранят убеждения в символьной форме подобно фактам в языке Prolog. Желания агентов (или цели) интуитивно соответствуют задачам, назначенным данным агентам. Для действующих BDI-агентов требуется, чтобы желания были логически непротиворечивы, хотя человеческие желания часто этому требованию не соответствуют. Агенты не могут в общем случае достичь всех своих желаний, даже если эти желания непротиворечивы. Агенты должны зафиксировать некоторое множество достижимых желаний и передать ресурсы для их достижения. Эти выбранные желания являются намерениями, и агент будет стремиться достичь намерения до тех пор, пока его убеждения соответствуют желанию либо желание при данных убеждениях не является достижимым.

Каждый агент в dMARS имеет библиотеку планов, определяющих варианты возможных действий, которые могут быть предприняты агентом для достижения его намерений. Планы, таким образом, реализуют процедурные знания агента. Каждый план содержит несколько компонентов. Триггер, или условие вызова, определяет обстоятельства (обычно в терминах событий), при которых план должен рассматриваться как возможный для применения. План имеет контекст, или предусловия, определяющие обстоятельства, при которых выполнение плана может начаться. План может иметь также главное условие, которое должно быть истинным во время выполнения плана. План имеет тело, которое может содержать цели (подцели) и примитивные действия (могут быть процедурными вызовами).

События, воспринимаемые агентом, помещаются в очередь событий. Внутренний интерпретатор агента отвечает за его поведение и непрерывно выполняет следующий цикл:

- проводит обзор мультиагентного мира, внутреннего состояния агента и изменяет очередь событий;
- генерирует новые возможные желания (задачи), находя те планы, чьи триггеры событий включены;
- выбирает из этого множества включенных планов один для выполнения;
- помещает желаемое значение в существующий или новый стек, в соответствии с тем, имеется или нет подцель;

- выбирает стек намерений, читает план, находящийся в вершине стека и выполняет следующий шаг из этого плана; если шаг есть действие — выполняет его, если это подцель — посылает эту подцель в очередь событий;
- возвращается к шагу обозрения.

Предпоследний шаг этого цикла реализует способ выполнения плана, когда подцель помещается в очередь событий, что вызывает активизацию очередного плана.

Содержательно модель агента в dMARS представляет собой ситуационную формальную систему.

Развитие моделей делиберативных агентов идет по пути попыток формализации новых мотивационных свойств и отношений в комбинации с поведением и действиями агентов. Такой подход приводит к созданию абстрактных логических моделей, претендующих на строгое формальное описание всех релевантных свойств рациональных агентов в целях анализа, спецификации и верификации МАС.

Создание делиберативных архитектур требует решения таких проблем, как построение адекватного символического описания реального мира (модель мира), учитывающего сложность происходящих во времени процессов и действующих объектов; организация из имеющихся знаний логического вывода, который должен приводить к определенным действиям агентов. Примеры систем, построенных в этой архитектуре: Integrated Planning Execution and Monitoring (IPEM), Intelligent Resource-bounded Machine Architecture (IRMA), AUTODRIVE, Homer.

Достоинством делиберативных моделей и архитектур является возможность применения строгих формальных методов и хорошо отработанных технологий традиционного искусственного интеллекта, позволяющих относительно легко представлять знания в символьной форме и переносить их в агентную систему.

В то же время создание полной и точной модели некоторой предметной области реального мира, формализация ментальных свойств агентов и процессов рассуждения в этих когнитивных структурах представляют существенные трудности для технической реализации.

Поиски путей разрешения проблем, возникающих при использовании в агентных системах классических методов ИИ, привели к появлению нового класса — реактивных архитектур. Основоположником этого направления принято считать Р. Брукса (R. Brooks), который так сформулировал ключевые идеи бихевиористического взгляда на интеллект:

- интеллектуальное поведение может создаваться без явного символьного представления знаний;
- интеллектуальное поведение может создаваться без явного абстрактного логического вывода;

- интеллект является внезапно возникающим свойством некоторых сложных систем.

В реальном мире интеллект не является экспертной системой или машиной логического вывода, а интеллектуальное поведение возникает как результат взаимодействия агента со средой. Известными примерами таких архитектур являются Pengi, архитектура ситуационных автоматов Agent Behaviour Language(ABLE).

Гибридные архитектуры и модели. В большинстве проектов и действующих систем используются гибридные архитектуры, спектр вариантов которых достаточно широк. Особенности построения таких архитектур рассмотрим на примере системы InteRRaP, соединяющей в себе свойства BDI-архитектур и многослойных (layered) архитектур. В основе BDI-архитектур лежит довольно строгая теоретическая концепция, но они недостаточно приспособлены для реального проектирования ресурсно-ограниченных и целенаправленных агентных приложений. Многослойные архитектуры хорошо поддерживают моделирование различных уровней абстракции, ответственности и сложности представления знаний, но слишком сложны для формального назначения свойств агентов и во многом зависят от интуиции разработчика.

Архитектура InteRRaP использует ментальные категории, определенные в BDI-теории для описания знаний агентов, целей и состояний. Графическое представление архитектуры InteRRaP дано на рис. 5.12.

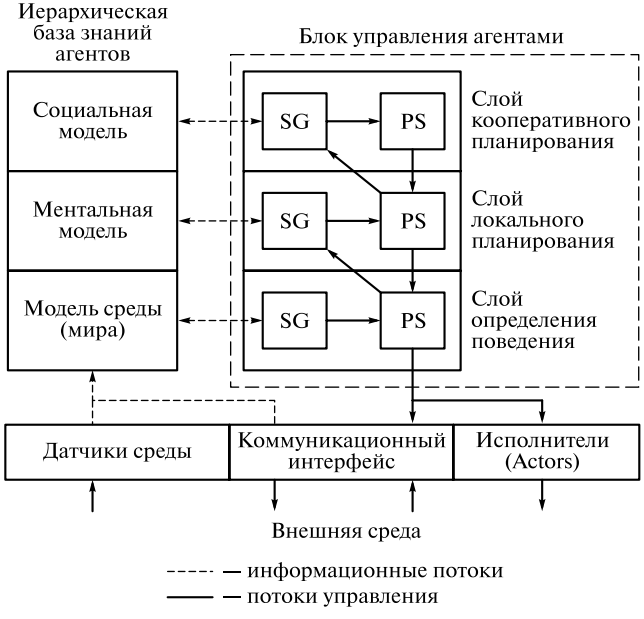


Рис. 5.12. Архитектура агента в системе InteRRaP

Таблица 5.2. **Базовые функции**

Функция	Слой определения поведения	Слой локального планирования	Слой кооперативного планирования
BR	Генерация и развитие убеждений (модель среды)	Абстракция локальных убеждений (ментальная модель)	Установление моделей других агентов (социальная модель)
SG	Активация образцов поведения	Распознавание ситуаций, требуемых локальным планированием	Распознавание ситуаций, требуемых кооперативным планированием
PS	Прямая связь от ситуации к последовательности действий	Модификация локальных намерений; локальное планирование	Модификация присоединенных намерений; кооперативное планирование

В данной архитектуре реализованы три базовые функции:

$BR(P, B) = B'$ — функция ревизии убеждений и абстрактных знаний, переводящая текущее состояние P агента и его старое убеждение B в новое убеждение B'; проявляется через иерархическую базу знаний агентов (см. рис. 5.12).

$SG(B, G) = G'$ — функция распознавания ситуации и активации цели, выводящая новую цель G' из убеждений агента B и его текущей цели G;

$PS(B, G, I) = I'$ — функция планирования и составления расписания, выводящая I' новых намерений, основанных на убеждениях B, целях G, выбранных функцией SG, и текущей внутренней структуры намерений I данного агента.

Базовые функции в зависимости от уровня иерархии получают свою направленность, что отражает табл. 5.2.

Каждый уровень иерархии содержит два процесса, реализующих функции SG и PS. Процесс SG_i распознает ситуации, интересующие соответствующий уровень, и выполняет активацию цели. Процесс PS_i реализует переход от целей к намерениям и действиям, принимая по входу пары «цель — ситуация», созданные процессом SG_i, определяет планы достижения целей и отслеживает выполнение шагов плана.

Если процесс PS_i не компетентен для ситуации S, он посылает требование активации, содержащее соответствующую пару «ситуация — цель» к PS_{i+1}, где описание ситуации обрабатывается за счет дополнительных знаний, доступных этому процессу, чтобы произвести соответствующее описание цели. В результате обработки ситуация S возвращается обратно к PS_i. Так работает механизм управления, основанный на компетентности.

Ситуация же определяется как множество формул

$$S \equiv S_B \vee S_L \vee S_C,$$

где $S_B \subseteq WM$, $S_L \subseteq MM$, $S_C \subseteq SM$; B— слой определения поведения; L — слой локального планирования; C — слой кооперативного планирования; WM — модель среды; MM — ментальная модель; SM — социальная модель.

Таким образом, ситуация описывается в контексте состояний внешней среды, целей и намерений агента и его убеждений о других агентах.

Внутренняя организация архитектуры InteRRaP представляется весьма логичной и убедительной, но такая сложная иерархия функций, процессов и моделей, естественно, не поддается строгой формализации, и работоспособность системы во многом обеспечивается интуицией и опытом разработчиков.

Область применения данной архитектуры — задача управления интерактивными роботами, выполняющими транспортные задачи в загрузочном цехе. Роботы-агенты принимают задачи загрузки и разгрузки грузовиков и могут вступать в конфликты с другими роботами, так как могут блокировать друг друга на подъездных путях, стремиться попасть на площадь, занятую другим роботом.

Нетрудно видеть, что решаемая транспортная задача является весьма типичной для такого рода автоматизированных и автоматических систем и предполагает действие однотипных агентов-роботов в общей среде, состояние которой, хотя бы в принципе, может быть точно определено. Типы ситуаций также составляют конечное и довольно ограниченное множество.

В корпоративных системах управления (масштаб предприятия, организации) интеллектуальные агенты, представляющие те или иные объекты и субъекты системы управления, будут заведомо неравноправны и должны быть включены в жесткую иерархическую структуру с определенными правами и обязанностями. При этом возникают проблемы распределения глобальных задач на подмножества агентов-исполнителей.

Рассмотренный пример хорошо демонстрирует достоинства и недостатки гибридных архитектур, которые позволяют гибко сочетать возможности различных моделей, но в большинстве случаев сильно зависят от специфики приложений, для которых они разрабатываются. Результаты проведенного анализа моделей интеллектуальных агентов сведены в табл. 5.1.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Для чего необходимы многослойные ИНС?
2. Какие существуют варианты использования ИНС в области управления?

3. В каких областях можно применять ИНС?
4. Опишите структуру и назовите особенности сети Хопфилда.
5. Как соотносятся процедуры обучения и управления?
6. Назовите группы и методы обучения ИНС.
7. Опишите δ-метод.
8. Перечислите основные характеристики интеллектуальных агентов.
9. Дайте определение мультиагентной системы.
10. Перечислите основные направления исследований в области МАС.
11. Дайте характеристику основным видам архитектур МАС.
12. Приведите примеры предметных областей, в которых необходимо применение МАС.
13. В чем состоят сложности реального применения МАС?
14. Какие свойства интеллектуальных агентов наиболее трудно поддаются формализации?
15. Предложите примеры практических задач управления и обработки информации, в которых можно было бы эффективно использовать МАС.
16. Что такое модальная логика?
17. Дайте классификацию уровня способностей.
18. Перечислите типы агентов и их свойства.

Глава 6

ТЕХНОЛОГИЯ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ УПРАВЛЕНИЯ

6.1. Процедура адаптации при переходе на выпуск новой продукции

В процессе управления организационно-экономическими системами изменяется структура спроса, которая определяет изменение цели работы системы. Под структурой понимается совокупность элементов и их связей. Для организационно-экономических систем управления введение (удаление) новых структурных элементов (реконструкция) связано с серьезными затратами, поэтому компенсация изменения состава вектора цели осуществляется изменением состава векторов плана и управляющих воздействий, т.е. изменением структурных связей системы (маршрутов — в терминах предметной области) и, возможно, весов отдельных составляющих целевых функций. В связи с этим в данной работе под изменением структуры подразумевается изменение связей при фиксированных структурных элементах.

На практике используется понятие «гибкость» — способность системы изменять цели функционирования за счет изменения структурных связей без существенных затрат (перевооружения производства). Гибкость, таким образом, определяет частный случай процедуры адаптации.

Гибкие системы, имеющие многоэлементный состав, относят к интеллектным системам с наличием человеческой составляющей.

Элементы управляющей части характеризуются в процессе эксплуатации (функционирования системы) наличием целенаправленности, экономического интереса, который в математической форме представлен целевой функцией. Работа целенаправленной системы проходит более успешно при согласовании экономических интересов.

Следует отметить, что источниками данных для математической модели таких систем являются документы и база данных. Если в первом случае сведения для объекта управления достаточно полны, то решения управляющей части далеко не всегда фиксируются документально. Данное обстоятельство характеризует неопределенность в процессе моделирования.

Отметим разницу процедур проектирования и эксплуатации в планировании старой и новой продукции. План P_3 для старой про-

дукции составляется задолго до его выполнения. Следовательно, имеется время для подготовки выпуска продукции в соответствии с новым спросом R_3. Можно считать, что план реализуется мгновенно и переходный процесс для P_3 отсутствует. Иными словами, процесс планирования — статический.

План P_4 для новой продукции при спросе R_4 должен быть оперативно рассчитан и немедленно реализован в процессе функционирования. В этом случае невозможно не учитывать инерционность постановки продукции на выпуск и инерционность выполнения плана, определяемую длительностью технологического цикла.

Графическая иллюстрация изменения состава вектора цели и возможных вариантов реакции системы на эти изменения представлена на рис. 6.1, *а—в*. Выпуск новой продукции может сопровождаться снижением спроса на старую продукцию со значения R_3 до R_3' (рис. 6.1, *г*).

Возможны следующие варианты выпуска новой и прекращения производства старой продукции.

I. Продукция P_3 снимается постепенно за время

$$T_c = \tau_3 = \max_j \tau_3,$$

где $\tau_3 = \{\tau_{3j}, j \in J_3 \subset J; J, J_3\}$ — множества выпускавшейся ранее (кривые *2* и *4* на рис. 6.1, *г*) и снимаемой продукции — вектор постоянных времени, а для новой продукции используется:

1) последовательный непрерывный способ запуска (только после снятия старой продукции) с мгновенным выпуском (кривая *7*, рис. 6.1, *б*);

2) последовательный непрерывный способ с выпуском продукции через время $T_п$ (кривая *8*, рис. 6.1, *б*);

3) последовательный прерывный способ (рис. 6.1, *в*);

4) параллельный способ запуска (в процессе снятия старой продукции — кривая *5*, рис. 6.1, *а*);

5) параллельный способ запуска (в процессе снятия старой продукции — кривая *6*, рис. 6.1, *а*).

II. Старая продукция P_3 снимается мгновенно (кривые *1* и *3*, рис. 6.1, *г*, $T_c = 0$), а в момент времени $(t) = (\tau - 1)$:

1) мгновенно производится новая продукция $P_4[\tau] \geq R_4[\tau]$ (кривая *5*, рис. 6.1, *а*);

2) начинается выпуск новой продукции с временем $T_п$ выхода на устойчивый уровень (кривая *6*, рис. 6.1, *а*).

Из рис. 6.1 видно, что процесс планирования в ИСУ является динамическим. В связи с этим используем для математического описания динамическое линейное программирование (ДЛП). ДЛП пригодно и для описания процесса управления. Метод с использованием ДЛП для процессов планирования и управления называется однородным.

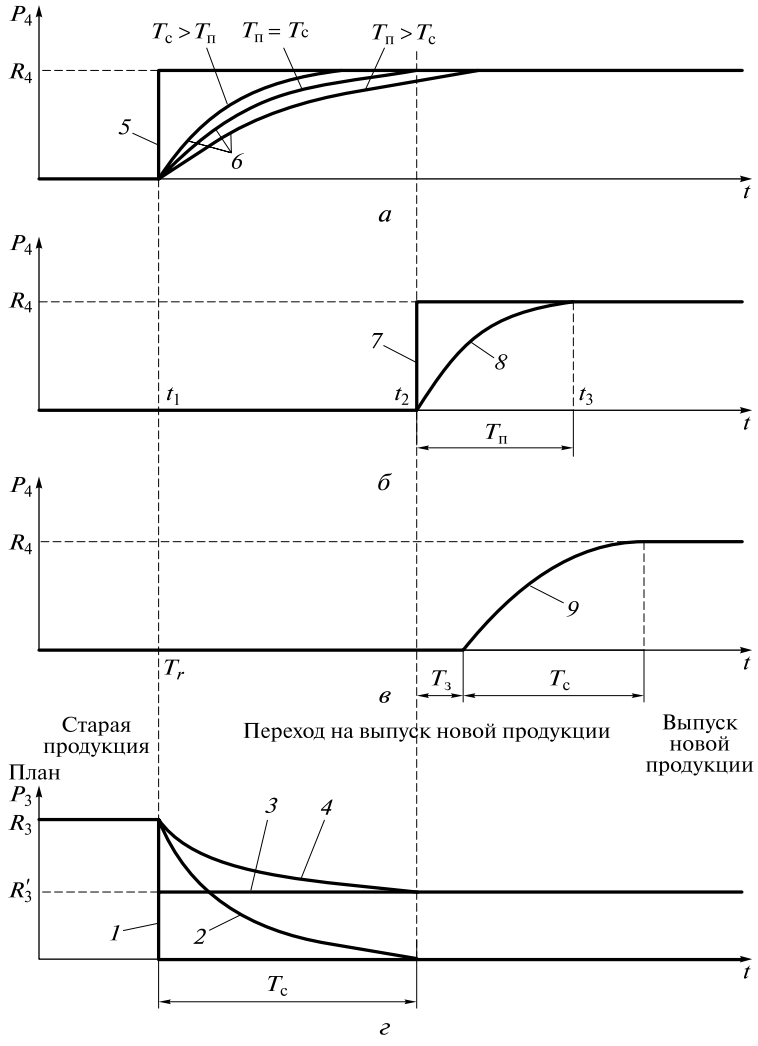


Рис. 6.1. Процедура ежедневного перехода на выпуск новой (P_4) продукции и одновременного снятия старой (P_3) продукции:

1, *2* — мгновенное и постепенное полное снятие старой продукции; *3*, *4* — мгновенное и постепенное частичное снятие старой продукции; *5*, *6*, *7*, *8* — мгновенная и постепенная постановка на выпуск новой продукции во время и после снятия старой продукции; *9* — постепенная постановка на выпуск новой продукции после снятия старой; $T_с$, $T_п$ — длительности переходных процессов

В описании процессов выделим этапы описания отдельного структурного элемента независимо от уровня иерархии (общее описание), описание с учетом уровней, взаимодействие элементов.

Начнем с общего описания.

Для процесса планирования оно имеет следующий вид:

$$\mathbf{P}(T) \geq \mathbf{R}(T); \tag{6.1}$$

$$\mathbf{P}(t_i) = \mathbf{P}(t_{i-1}) - \mathbf{p}(t_i);$$

$$\mathbf{z}(t_i) = \mathbf{A}\mathbf{z}(t_{i-1}) + \mathbf{B}\mathbf{p}_1(t_{i-1}),\ \mathbf{z}(0) = \mathbf{z}_0;$$

$$\mathbf{p}(t_i) = \mathbf{C}\mathbf{z}(t_i);$$

$$\mathbf{D}\mathbf{p}_1(t_i) \leq \mathbf{b}(t_{i-1}); \tag{6.2}$$

$$G = -\mathbf{F}\mathbf{P}(T) > \rightarrow \min, \tag{6.3}$$

где $\mathbf{z}$, $\mathbf{p}$, $\mathbf{P}$ — незавершенное производство, планы текущий и с накоплением; $\mathbf{p}_1$ — запуск комплекта материалов в производство; $\mathbf{R}$ — спрос; $\mathbf{D}$ — матрица норм расходов; $\mathbf{b}$ — наличное количество ресурсов; $\mathbf{F}$ — прибыль от выпуска единицы продукции; $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ — матрицы соответствующих размерностей; T, υ — интервалы времени; $T = N\upsilon$.

Если процесс планирования статический, то используются выражения (6.1), (6.2), (6.3).

Процесс управления получает такое описание.

Объект управления системы может быть представлен как

$$\mathbf{z}(t_i) = \mathbf{A}\mathbf{z}(t_{i-1}) + \mathbf{B}\mathbf{u}(t_{i-1});$$

$$\mathbf{y}(t_i) = \mathbf{C}\mathbf{z}(t_i);$$

$$\mathbf{D}\mathbf{u}(t_i) \leq \mathbf{b}(t_{i-1}),$$

где $\mathbf{p}$, $\mathbf{z}$, $\mathbf{u}$, $\mathbf{y}$, $\mathbf{b}$ — векторы плана, состояния, управления размерности ресурсов, выхода, наличных ресурсов, поступления ресурсов; $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ — матрицы, характеризующие динамику; $\mathbf{D}$ — матрица норм расхода ресурсов.

Управляющая часть

$$\varepsilon(t_i) = \mathbf{p}(t_i) - \mathbf{y}(t_i);$$

$$J = \sum_{i=0}^{N} \left\{ \mathbf{C}_1 \varepsilon(t_i) + \mathbf{C}_2 \mathbf{u}(t_i) \right\} \rightarrow \min,$$

где $\mathbf{C}_1$, $\mathbf{C}_2$ — вектор-строки потерь за счет отклонения от плана и потребностей в дополнительных ресурсах для управления; $\varepsilon(t) = \mathbf{p}(t_i) - \mathbf{y}(t_i)$ — вектор отклонений; T, υ — интервалы времени; $T = N\upsilon$.

Теперь можно проанализировать процесс работы системы управления подробнее.

Пусть новая продукция $\mathbf{P}_4[\tau]$, старая продукция $\mathbf{P}_3[\tau]$, продукции $\mathbf{P}_3[\tau]$ и $\mathbf{P}_2[\tau]$ имеют попарно общие ресурсы и не имеют общих ресурсов с $\mathbf{P}_1[\tau]$.

Очевидно, что старый план $\mathbf{P}^{c}[\tau] = \{\mathbf{P}_1^T[\tau], \mathbf{P}_2^T[\tau], \mathbf{P}_3^T[\tau]\}$, $|\mathbf{P}^{c}| = |\mathbf{P}_1| + |\mathbf{P}_2| + |\mathbf{P}_3|$, где $|.|$ — размерность вектора; индекс с — для старого плана; требуемые ресурсы $\mathbf{b}^{cT}(\tau - 1) = \{\mathbf{b}_1^T(\tau - 1), \mathbf{b}_2^T(\tau - 1), \mathbf{b}_3^T(\tau - 1)\}^T$, получаемая прибыль $\mathbf{F}^{c} = \{\mathbf{F}_i,\ i = 1, 3\}$.

Для выпуска новой продукции выделяются дополнительные ресурсы $\mathbf{b}_4(\tau - 1)$, $\mathbf{b}_3^1(\tau - 1)$, $\mathbf{b}_2^1(\tau - 1)$. Предполагаем, что наличие ресурсов $\{\mathbf{b}_2(\tau - 1) + \mathbf{b}_2'(\tau - 1)\}$, $\{\mathbf{b}_3(\tau - 1) + \mathbf{b}_3'(\tau - 1)\}$, $\mathbf{b}_4(\tau - 1)$ обеспечивает выполнение нового плана $\mathbf{P}^{н}[\tau] = \{\mathbf{P}_1^{T}[\tau], \mathbf{P}_2^{T}[\tau], \mathbf{P}_3^{T}[\tau], \mathbf{P}_4^{T}[\tau]^{T}$ при $\mathbf{P}_4[\tau] \geq \mathbf{R}_4[\tau]$. В противном случае следует решить вопрос ресурсного обеспечения.

Тогда новый план связан с ограничениями

$$\begin{vmatrix} \mathbf{D}_1 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_2 & \mathbf{D}_6 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{D}_9 & \mathbf{D}_{11} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{D}_{12} \end{vmatrix} \times \begin{vmatrix} \mathbf{P}_1[\tau] \\ \mathbf{P}_2[\tau] \\ \mathbf{P}_3'[\tau] \\ \mathbf{P}_4[\tau] \end{vmatrix} \leq \begin{vmatrix} \mathbf{b}_1 \\ \mathbf{b}_2(\tau-1) + \mathbf{b}_2'(\tau-1) \\ \mathbf{b}_3(\tau-1) + \mathbf{b}_3'(\tau-1) \\ \mathbf{b}_4(\tau-1) \end{vmatrix}, \tag{6.4}$$

где $\mathbf{D}_1$, $\mathbf{D}_2$,$\mathbf{D}_6$, $\mathbf{D}_9$, $\mathbf{D}_{11}$, $\mathbf{D}_{12}$ — подматрицы соответствующей размерности матрицы $\mathbf{A}$ норм расхода ресурсов; $\mathbf{P}_3'$ — выпуск старой продукции после начала выпуска новой продукции; $\mathbf{b}_1$ — ресурс для плана P_1.

Старый план был рассчитан до появления новой продукции и поэтому, предполагая, что истрачены все ресурсы, неравенство (6.4) можно заменить на равенство:

$$\begin{vmatrix} \mathbf{D}_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_2 & \mathbf{D}_6 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{vmatrix} \times \begin{vmatrix} \mathbf{P}_2[\tau] \\ \mathbf{P}_2[\tau] \\ \mathbf{P}_3[\tau] \end{vmatrix} = \begin{vmatrix} \mathbf{b}_1 \\ \mathbf{b}_2(\tau-1) \\ \mathbf{b}_3(\tau-1) \end{vmatrix}. \tag{6.5}$$

Основная идея формирования модели нового плана — запись в *отклонениях* (6.4) от известного старого плана (6.5), что позволяет получить результат для нового плана с неизвестными значениями $\mathbf{P}_4[\tau]$ и $\mathbf{P}_3'[\tau]$:

$$\mathbf{P}_4[\tau] \Leftarrow \mathbf{R}_4[\tau],$$

$$\begin{vmatrix} \mathbf{D}_6 & 0 \\ \mathbf{D}_9 & \mathbf{D}_{11} \\ 0 & \mathbf{D}_{12} \end{vmatrix} \times \begin{vmatrix} \mathbf{P}_3'[\tau]\ \mathbf{P}_3[\tau] \\ \mathbf{P}_4[\tau] \end{vmatrix} \leq \begin{vmatrix} \mathbf{b}_2'(\tau-1) \\ \mathbf{b}_3'(\tau-1) \\ \mathbf{b}_4(\tau-1) \end{vmatrix}, \tag{6.6}$$

где $\mathbf{D}_6$, $\mathbf{D}_9$, $\mathbf{D}_{11}$, $\mathbf{D}_{12}$ — подматрицы соответствующей размерности матрицы $\mathbf{A}$ норм расхода ресурсов.

Поскольку часто $|\mathbf{P}_4| << |\mathbf{P}|$, $|\mathbf{P}_3| << |\mathbf{P}|$, то возможно существенное снижение размерности (а следовательно, и времени расчета) задачи (6.6) по сравнению с исходной задачей.

Для интеллектных систем исследуем два случая принятия решений по выбору цели: предварительное решение без учета динамики процесса перехода на выпуск новой продукции и окончательное решение с учетом динамичности процесса. В первом случае проводится укрупненная оценка целесообразности перехода на выпуск новой продукции с использованием статического описания процесса планирования.

Если значение $\mathbf{P}'_3[\tau]$ задано и фиксировано, целевая функция получает вид

$$G_4 = \mathbf{F}_4\mathbf{P}_4[\tau] \to \max.$$

Отметим, что в варианте I для случаев 1, 3, 5 потери за счет незавершенного производства при снятии с производства старой продукции $\mathbf{P}_3$ равны нулю ($\Delta G_3 = 0$), тогда как для случая 4, а также 1 и 2 в варианте II эти потери ненулевые — $\Delta G_3 \neq 0$.

Если укрупненная оценка определяет, что переход на выпуск новой продукции выгоден, то можно провести уточненную оценку.

Случай 1 в варианте II также является идеальным, однако здесь следует учитывать потери от незавершенного производства и выражение $\Delta G_3 = \mathbf{F}_3(\mathbf{z}_3 - \mathbf{z}_3') \le \mathbf{F}_4,\mathbf{P}_4$. Здесь $\mathbf{z}_3$, $\mathbf{z}'_3$ определяются как незавершенное производство для $\mathbf{P}_3$ и $\mathbf{P}'_3$ соответственно.

Нетрудно заметить, что для уточненной оценки условия перехода на выпуск новой продукции следует использовать ДЛП.

6.2. Описание процессов планирования и управления с учетом специфики уровней

Рассмотрим специфику описания разных уровней трехуровневой структуры управления (рис. 6.2).

Процесс планирования. Для уровня $h = 3$ справедливо

$$\mathbf{z}_{\text{п}}(t) = \mathbf{z}_{\text{п}}(t-1) + [t](\mathbf{p}_1[t] - \mathbf{p}[t]),\ \mathbf{z}_{\text{п}}(0) = \mathbf{z}_{\text{п}0};$$

$$\mathbf{p}(t) = \mathbf{C}\mathbf{z}^{\text{п}}(t-1);$$

$$\mathbf{P}(t) = \mathbf{P}(t-1) + \mathbf{p}(t),\ \mathbf{P}(0) = 0;$$

$$\mathbf{D}\mathbf{p}_1(t) \le \mathbf{b}(t);$$

$$\text{P(T)} \ge \text{R(T)};$$

$$\mathbf{b}_m(t) = \mathbf{b}_m(t-1) + \Delta\mathbf{b}_m(t) - \mathbf{A}_m\mathbf{p}_1(t);$$

$$\mathbf{b}_\psi(t) = \mathbf{b}_\psi(t-1) + \Delta\mathbf{b}_\psi(t);$$

$$J = \mathbf{C}_3\mathbf{P}(T) > \to \min,$$

где $\mathbf{z}_{\text{п}}$, $\mathbf{p}$ — незавершенное производство и ежедневный план; $\mathbf{p}_1$ — запуск комплекта материалов в производство; $\mathbf{D}$ — матрица норм

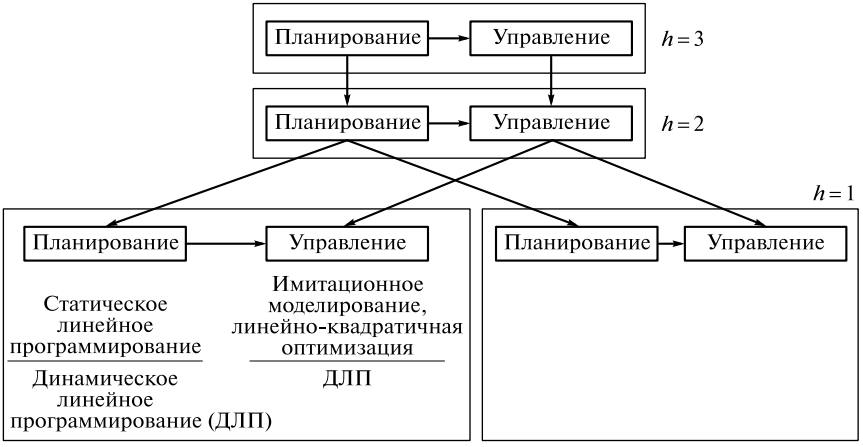


Рис. 6.2. Связь и управление по горизонтали и по вертикали

расходов, **R** — спрос; **b** — наличное количество ресурсов; $\Delta\mathbf{b}$ — поступление ресурсов; m, ψ — материальные и другие виды ресурсов.

Для уровня $h = 1$ процесс планирования состоит из статической части

$$\sum_{j=1}^{J} d_{\psi jk}\, p_{jk}(t_i) \le b_{\psi k}(t_{i-1});$$

$$\sum_{j=1}^{J} d_{mjk}\, p_{jk}(t_i) \le b_{mk}(t_{i-1});$$

$$\sum_{i=1}^{I} d_{mjk}\, p_{jk}(t_i)\Big|_{k=1} \le b_m(0); \qquad (6.7)$$

$$\sum_{i=1}^{I} p_j(t_i)\Big|_{k=K} \le P_j(T);$$

$$F_k = \sum_{j=1}^{J}\sum_{i=1}^{I} C_{jk}\, p_{jk}(t_i) \to \max$$

и динамической части

$$\mathbf{z}_{\text{п}jk}(t_i) = \mathbf{z}_{\text{п}jk}(t_{\text{i}-1}) + [t](\mathbf{p}_{1jk}[t_i] - \mathbf{p}_{jk}[t_i]),\ \mathbf{z}_{\text{п}jk}(0) = \mathbf{z}_{\text{п}jk0};$$

$$\mathbf{p}_{jk}[t_i] = F_{jk} z_{\text{п}jk}(t_{i-1});$$

$$\mathbf{P}_{jk}(t_i) = \mathbf{P}_{jk}(t_{i-1}) + \mathbf{p}_{jk}[t_i],\ \mathbf{P}_{jk}(0) = 0;$$

$$\mathbf{b}_{mk}(t_i) = \mathbf{b}_{mk}(t_{i-1}) + \Delta\mathbf{b}_{mk}[t_i] - \mathbf{A}_m \mathbf{p}_{1jk}[t_i];$$

$$\mathbf{b}_{\psi k}(t) = \mathbf{b}_{\psi k}(t_{i-1}) + \Delta b_{\psi k}(t),$$

где $m = 1$, M — виды материальных ресурсов; $\psi = 1$, Ψ — виды прочих ресурсов; $i = 1$, I — моменты времени; $k = 1$, K — номер подразделения.

Для уровня $h = 2$ вторая и последняя строки в неравенствах (6.7) трансформируются:

$$\sum_{i=1}^{I} d_{mjk}\, p_{jk}(t_i) \le \sum_{i=1}^{I} p_{m,k-1}(t_{i-1});$$

$$F = \sum_{k=1}^{K} F_k \to \max.$$

Процесс управления. Объект управления в общем случае представляется в виде

$$\dot{x}_k(t) = \mathbf{A}_k \mathbf{z}_k(t) + \mathbf{B}_k \mathbf{u}_k(t) + \sum_{j=1, j\ne k}^{K} \mathbf{A}_{kj} \mathbf{z}_j(t) + \mathbf{B}_{0k} \mathbf{U}_0(t) + \mathbf{w}_k(t);$$

$$k = 0,\ K;\ \mathbf{B}_{00} = \mathbf{B}_0;$$

$$\mathbf{z}_k(0) = \mathbf{z}_{k0};\ \mathbf{y}_k(t) = \mathbf{C}_k \mathbf{z}_k(t);\ \mathbf{u}_k(t)_{|k=0} = \mathbf{U}_0(t)$$

или

$$\mu \dot{\mathbf{z}}_k(T) = \mathbf{A}_k \mathbf{z}_k(T) + \mathbf{B}_k \mathbf{u}_k(T) + \sum_{j=1, j\ne k}^{K} \mathbf{A}_{kj} \mathbf{z}_j(T) + \mathbf{B}_{0k} \mathbf{U}_0(T) + \mathbf{w}_k(T);$$

$$k = 0,\ K;$$

$$\mathbf{z}_k(0) = \mathbf{z}_{k0};\ \mathbf{y}_k(T) = \mathbf{C}_k \mathbf{z}_k(T)$$

при

$$\mathbf{B}_k = \begin{cases} \mathbf{B}_k, & k = 1, K; \\ 0, & k = 0; \end{cases} \qquad \mu = \begin{cases} \mu, & k = 1, K; \\ 0, & k = 0, \end{cases}$$

где $\mathbf{z}_k$, $\mathbf{u}_k$, $\mathbf{y}_k$, $\mathbf{w}_k$ — вектор-столбцы состояний, управления, выхода, возмущений; $\mathbf{A}_k$, $\mathbf{B}_k$, $\mathbf{A}_{kj}$, $\mathbf{B}_{0k}$, $\mathbf{B}_0$, $\mathbf{C}_k$ — матрицы подходящих размерностей; $[t] = \mu^*[T]$, $\mu = 1/25$ — малый параметр, учитывающий изменение масштаба по времени; K — число элементов системы. Отсчет по времени t называют быстрым, а по T — медленным.

Описание управляющей части имеет вид:

$$\boldsymbol{\varepsilon}_k(\beta) = \mathbf{p}_k(\beta) - \mathbf{y}_k(\beta);\ \beta = t \text{ или } \beta = T;$$

$$J_k = \int_0^{\gamma} \{\mathbf{Q}_k\, \boldsymbol{\varepsilon}_k(\beta) + \mathbf{R}_k \mathbf{u}_k(\beta)\}\, d\beta \to \min;\ \gamma = T \text{ или } \gamma = \tau;$$

$$J = \sum_{k=1}^{K} J_k,$$

где $\boldsymbol{\varepsilon}_k$ — вектор-столбец отклонений; $\mathbf{p}_k(\beta)$ — непрерывный план; $\mathbf{Q}_k$, $\mathbf{R}_k$ — положительно полуопределенная и определенная матрицы; J_k — целевая функция.

6.3. Компьютерная реализация интеллектуальных систем управления

Компьютерная реализация ИСУ с трехуровневой структурой может быть осуществлена на СУБД InterBase, в комплексе MatLab, на языках программирования Pascal и Java.

Далее рассмотрим реализацию трехуровневой структуры ИСУ в рамках комплекса MatLab, позволяющего строить широкий спектр моделей достаточно высокой размерности.

В реализации трехуровневой структуры ИСУ следует выделить две составляющих: отдельный элемент и совокупность связанных элементов.

Отдельный элемент. Для компьютерной реализации описаний элементов и проверки работоспособности алгоритмов необходимы числовые данные. Их можно получить двумя способами: 1) из реальной системы; 2) моделированием на компьютере. Первый способ связан с серьезными трудностями. В связи с этим используем второй способ.

Генератором задачи статического линейного программирования (СЛП) может служить алгоритм и программа отдельного элемента, приведенные в работе [2]. Следует отметить, что для отдельного элемента получение числовых данных упрощается, поскольку может быть взят любой числовой пример задачи СЛП из научной или учебной литературы.

П л а н и р о в а н и е. Задачу СЛП в рамках MatLab можно реализовать либо прямым набором программы для стандартного алгоритма линейного программирования, либо, что предпочтительнее, с использованием пакета SIMULINK (рис. 6.3).

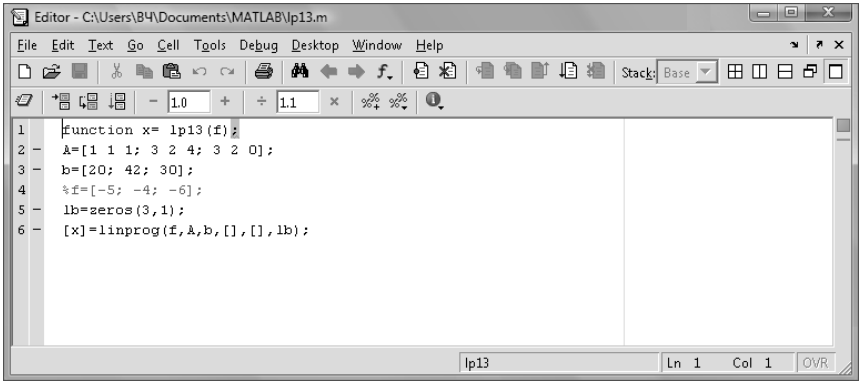


Рис. 6.3. Реализация отдельного элемента (задача СЛП)

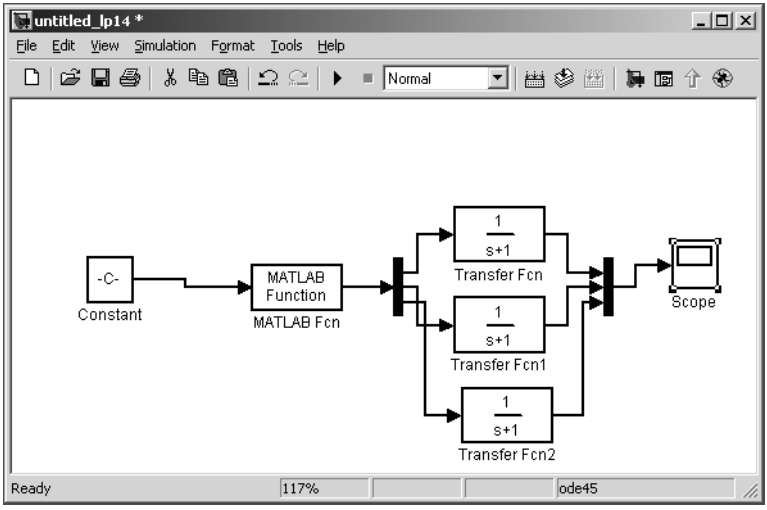


Рис. 6.4. Реализация отдельного элемента (задача ДЛП)

Этот пакет позволяет легко реализовать задачи динамического линейного программирования (рис. 6.4, 6.5) и перехода на выпуск новой продукции (рис. 6.6, 6.7).

У п р а в л е н и е. Сложнее обстоит дело с процессом управления, поскольку нужны элементы SIMULINK с двумя векторными входами. В этом случае приходится составлять программу.

Пусть объект управления описывается выражением $W_0(s) = 1/(s + 1)$, управляющая часть представляет собой ПИ-регулятор с передаточной функцией $W_{ПИ}(s) = (s + 1)/s$. Эта же задача может быть пред-

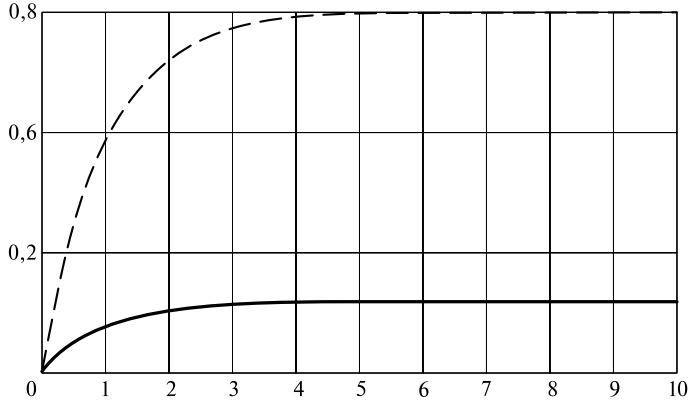


Рис. 6.5. Переходный процесс при планировании с ДЛП

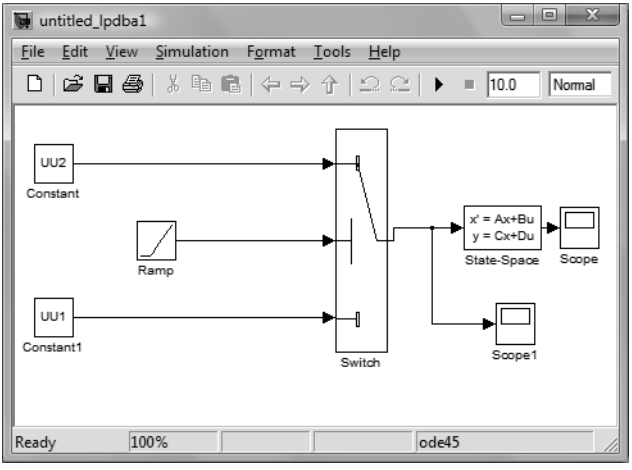


Рис. 6.6. Переход на выпуск новой продукции

ставлена в дискретном матричном виде с шагом дискретности h = = 0,1:

$$(\mathbf{I}+h\mathbf{A})=\begin{vmatrix}1 & 0{,}1 & 0\\ 0 & 1 & 0{,}1\\ 0 & -0{,}1 & 0{,}9\end{vmatrix}\qquad h\mathbf{B}=\begin{vmatrix}0\\ 0\\ 0{,}1\end{vmatrix}\qquad \mathrm{C}=|\,1\quad 1\quad 0\,|,$$

где **I** — единичная матрица.

Развернем преобразование динамики для пяти интервалов времени, введя обозначения (**I** + h**A**) = **A**, h**B** = **b**:

$$z(1) = Az(0) + bu(0),$$

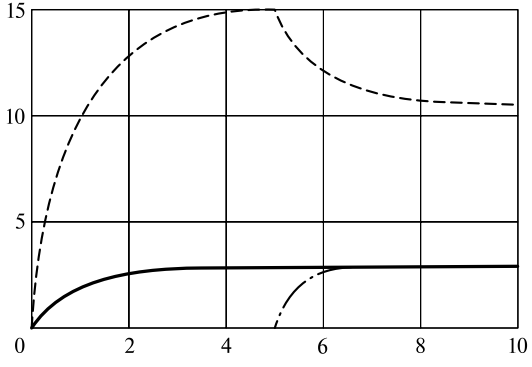


Рис. 6.7. Планирование при появлении новой продукции

$$y(1) = Cz(1) = C(Az(0) + bu(0));$$

$$\varepsilon(1) = p(1) - y(1) = p(1) - C(Az(0) + bu(0));$$

...

$$z(5) = Az(4) + bu(4) = A^5z(0) + A^4bu(0) + A^3bu(1) + A^2bu(2) +$$
$$+ Abu(3) + bu(4);$$

$$y(5) = Cz(5) = C(A^5z(0) + A^4bu(0) + A^3bu(1) + A^2bu(2) +$$
$$+ Abu(3) + bu(4));$$

$$\varepsilon(5) = p(5) - y(5) = p(5) - C(A^5z(0) + A^4bu(0) + A^3bu(1) +$$
$$+ A^2bu(2) + Abu(3) + bu(4));$$

$$c_1\varepsilon(5) + c_2(u(0)\ u(1)\ u(2)\ u(3)\ u(4)) = c_1(A^5z(0) + A^4bu(0) +$$

$$+ A^3bu(1) + A^2bu(2) + Abu(3) + bu(4)) + c_2(u(0)\ u(1)\ u(2)\ u(3)\ u(4)).$$

Исходя из преобразований можно написать следующую программу, в которой матрицы **A**, **B**, **C** обозначены через **A0**, **B0**, **C0** соответственно.

Программа 1.

```
function U = zet1(A0, B0, C0);
A0 = [1 0.1 0; 0 1 0.1; 0 −0.1 0.9];
B0 = [0; 0; 0.1];
C0 = [1 1 0];
C0*A0^0*B0;
C1 = 1.5; C2 = ones(1,30);
M(1,1) = ans; S(1,1) = ans;
for i = 2:30 C0*A0^(i-1)*B0; M(1,i) = ans;
if i = = 2, S(1,2) = M(1,1) + ans, else S(1,i) =
S(1,i-1) + ans;end
M1 = rot90(M,2);
M2 = −M1;
S2 = −rot90(S,2);
R11 = C1*S2;
p = ones(30,1);
b1 = ones(31,1);
b2 = −0.4;
b = cat(1,b1,b2);;
lb = zeros(30,1);
R2 = eye(30,30);
save zet11.mat;end
f1 = R11 + C2;
f = f1';
```

```
A = cat(1,M1,R2,M2);
[x] = linprog(f,A,b,[],[],lb)
Y1(1) = M1(1)*x(1);
for i = 2:30 Y1(i) = Y1(i-1) + M1(i)*x(i); end
Y = rot90(Y1,3);
E = p-Y;
W = M1*x
V = f1*x + 30
save zet11.mat;
%function [x] = lp1_dlp5(f,A,b,lb);
M
S
M1
M2
S2
R11
b1
b
lb
R2
A
Y
Y1
f1
end

Решение x =

1.0000
  1.0000
  1.0000
  1.0000
  1.0000
  1.0000
  1.0000
  0.5567
  0.0000
  0.0000
  0.0000
  0.0000
  0.0000
  0.0000
  0.0000
```

```
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
  0.0000
  0.0000
```

Целевая функция $V = 4{,}6015$, значение глобального ограничения $W = 0{,}4000$.

Совокупность элементов. П л а н и р о в а н и е. Реализовать процесс планирования совокупности структурных элементов диспетчерского уровня ($h = 2$) можно с помощью схемы, показанной на рис. 6.8.

Однако в этом случае возникает задача согласования данных связей элементов. Ее можно решить, используя упомянутый алгоритм (программу) отдельного элемента. На его основе можно создать числовые модели для следующих случаев (рис. 6.9):

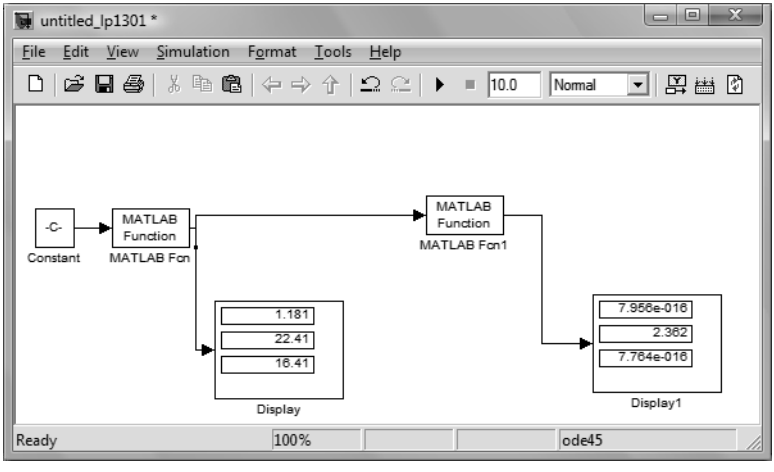


Рис. 6.8. Процесс планирования уровня $h = 2$

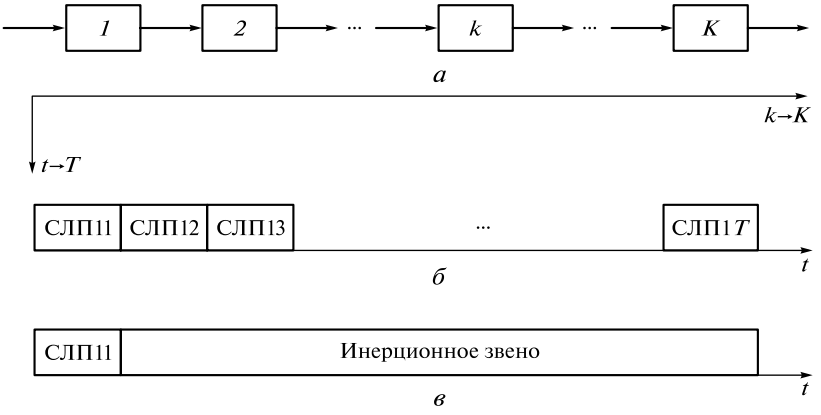


Рис. 6.9. Схема модели диспетчерского уровня (*а*), изменение параметров задачи СЛП на интервалах времени *t* (*б*) и создание задачи ДЛП (*в*)

1) последовательная цепочка из *K* элементов для одного интервала времени (рис. 6.9, *а*);

2) добавляется (рис. 6.9, *б*) дискретная шкала времени с интервалами *t* и временем моделирования *T*. В каждом интервале времени задача СЛП *IJ* имеет свои числовые данные, вводимые в режиме диалога, где *I* — номер элемента; *J* — интервал времени;

3) в случае 2 задача СЛП генерируется только на интервале $t = 1$, а на остальных интервалах (рис. 6.9, *в*) «работает» инерционное звено, трансформируя задачу СЛП в задачу ДЛП;

4) в случае 2 до интервала $t_1 < T$ план не меняется, а в момент t_1 изменяется размерность плана. Таким способом моделируется процедура перехода на выпуск новой продукции;

5) агрегация данных для формирования элемента более высокого уровня ($h = 3$). Ценность этого случая в том, что он «привязан» и согласован с другими случаями.

Наибольший интерес представляет общий случай 2, из которого, как частные, следуют все остальные. Рассмотрим его подробнее.

Математическое описание отдельного элемента уровня $h = 2$ с использованием задачи линейного программирования имеет вид:

$$\mathbf{DN} \le \mathbf{X} \le \mathbf{DV};$$

$$\mathbf{BN} \le \mathbf{AX} \le \mathbf{BV};$$

$$G = \mathbf{CX} \to \max,$$

где **X** — вектор искомого плана; **DN, DV** — векторы нижнего и верхнего ограничений на план; **A** — матрица норм расходов ресурсов;

C — вектор прибыли от единицы плановой продукции; **BN, BV** — векторы нижней и верхней границ ресурсов. Выходом является вектор **X**, входом — вектор **B** = (**BN**, **BV**).

Введем понятия элементов $k = 1, K$ и интервалов времени $t = 1, T$, на которых работает цепочка элементов. Программно интервалы вводятся циклами, число которых задается в режиме диалога. На каждом интервале вручную вводится матрица **A** для каждого элемента. Для интервала $t = 1$ задается в диалоге план для последнего элемента в цепочке. Кроме того, для всех интервалов с индексом $t > 1$ вводится составляющая ΔB дополнительных ресурсов для корректировки плана при переходе от элемента к элементу.

Работа с программой имеет следующий вид.

Первоначально осуществляется задание параметров в режиме диалога.

```
Vvedite chislo vremennyh intervalov //Приглашение
ввести число временных интервалов, на которых будет
решаться последовательность задач
Vvedite chislo reshaemyh zadach //Приглашение
ввести число задач в цепочке
Vvedite razmernost zadachi #1
N = //Ввод числа прямых ограничений
M = //Ввод числа основных ограничений
Vvedite plan //Ввод плана
X[1] =
X[2] =
…
Vvedite dB dlya zadachi #2 //Ввод изменения вектора
B для интервалов более одного
dB[1] =
dB[2] =
…
Vvedite matricu A //Ввод матрицы A
A[1,1] =
A[1,2] =
…
```

Затем программой проводятся вычисления, а результаты помещаются в файлы. Файлы результатов по задачам носят имена OUT_1_1. TXT, OUT_1_2. TXT ..., где первое число — номер временного интервала, а второе число — номер задачи в цепочке. Результаты приведены в табл. 6.1, в которой введены дополнительные обозначения: CX — оптимальное значение целевой функции; Xopt — оптимальный план; **B** — вектор **AX**.

Таблица 6.1. **Результаты расчета**

OUT_1_1.TXT	OUT_1_2.TXT
3< = = N 2< = = M C, CX = 1031.40 108.30 111.30 92.40 DN 0.00 0.00 0.00 DV 4.00 4.00 4.00 BN 0.00 0.00 BV 26.00 30.00 A 1.00 4.00 2.00 5.00 2.00 3.00 X 4.00 4.00 4.00 Xopt 2.00 4.00 4.00 B 26.00 30.00	2< = = N 4< = = M C, CX = 77746.00 1541.00 1256.00 DN 0.00 0.00 DV 26.00 30.00 BN 0.00 0.00 0.00 0.00 BV 172.00 142.00 190.00 164.00 A 2.00 4.00 2.00 3.00 5.00 2.00 4.00 2.00 X 26.00 30.00 Xopt 26.00 30.00 B 172.00 142.00 190.00 164.00
OUT_2_1.TXT	**OUT_2_2.TXT**
C, CX = 2562.50 431.20 491.00 383.10 DN 1.00 0.00 0.00 DV 3.00 2.00 3.00 BN 15.00 18.00 BV 190.00 190.00 A 1.00 4.00 2.00 5.00 2.00 3.00 X 3.00 2.00 3.00 Xopt 1.00 2.00 3.00 B 15.00 18.00	C, CX = 43630.40 1153.70 960.70 DN 15.00 18.00 DV 190.00 190.00 BN 0.00 0.00 0.00 0.00 BV 134.00 109.00 135.00 118.00 A 2.00 4.00 2.00 3.00 5.00 2.00 4.00 2.00 X 190.00 190.00 Xopt 17.00 25.00 B 134.00 109.00 135.00 118.00

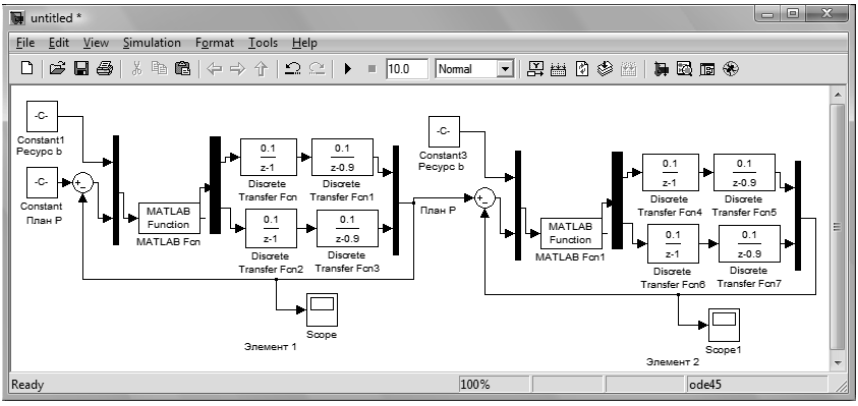


Рис. 6.10. Схема управления с оптимизацией

Сгенерированная числовая модель позволяет легко решать задачу согласования экономических интересов.

У п р а в л е н и е. Наиболее предпочтительным вариантом реализации процесса управления была бы схема, показанная на рис. 6.10. К сожалению, элемент MatLab Fcn не может работать с двумя векторными входами. Вместе с тем для процесса управления достаточно ввести в описанную ранее программу 1 (уровень $h = 2$) циклы для элементов и времени. Программа 1 позволяет формировать случаи, описанные в процессе управления.

Сформированная таким образом модель процесса управления дает возможность решать задачи обеспечения неколебательности переходных процессов и точности реакции системы на скачок входного сигнала.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое адаптация и гибкость?
2. Почему используется задача ДЛП, а не СЛП?
3. Дайте математическое описание отдельных элементов процессов планирования и управления с применением ДЛП.
4. Дайте математическое описание процессов планирования и управления элементов разных уровней.
5. Перечислите случаи генерации числовых моделей совокупностей элементов.

Глава 7

ТЕХНОЛОГИЯ ИНТЕЛЛЕКТУАЛЬНОГО АНАЛИЗА ДАННЫХ

7.1. Характеристика технологий интеллектуального анализа данных

Суть интеллектуального анализа данных. Современные системы управления имеют две тенденции: усложнение структуры и распространение процесса управления на более длительные интервалы времени. В этих условиях усложняются как процедура сбора и обработки ретроспективной и текущей числовой информации, так и ее обработка, прежде всего, для получения числовых результатов прогнозирования.

Специфический подкласс информационных технологий, ориентированных на прогнозирование состояния сложных динамических систем в нестационарных и неоднородных средах, разработку сценариев развития ситуаций в условиях комплексной динамической неопределенности, ситуационный анализ текущей обстановки, называют аналитическими технологиями или технологиями интеллектуального анализа данных (ИАД).

Под ситуацией, оценка которой предполагает определение количественных и качественных характеристик, в первую очередь, интегрального уровня, понимается совокупность характеристик объекта управления и среды его функционирования в заданный момент времени.

Технология ИАД используется на этапе учета в цикле управления при принятии стратегических решений.

Потребность формирования аналитических технологий определяется следующими обстоятельствами:

- при реализации современных систем управления следует учитывать как прогресс развития любого объекта управления и среды, так и их усложнение, выражающееся в том числе и в увеличении числа исследуемых динамических характеристик;
- необходимость как следствие хранения временны́х рядов, характеризующих наблюдаемые ситуации;
- рост требований к эффективности управления (оперативность, обоснованность и достоверность процессов принятия решений), определяющей качество функционирования системы в целом, в условиях возрастающей конкурентной рыночной борьбы неантагонистического или антагонистического характера.

Формированию технологий ИАД способствуют:

- значительное количество информационных систем, которые относятся к классу информационных систем операционной обработки данных в процедурах планирования, учета и контроля (транзакционные системы OLTP). OLTP-системы создают возможность накопления первичных данных в виде временны́х рядов наблюдений для их последующего анализа в интересах подготовки и принятия решений на основе прогноза развития ситуаций;
- существенное возрастание вычислительных мощностей компьютеров с возможностью реализации эффективного диалога с пользователями, объемов хранимой оперативно-доступной информации с возможностью для формирования значительных информационных пространств предприятий, организаций, корпораций;
- резкое снижение удельной стоимости вычислительных ресурсов.

Аналитические технологии ориентированы на решение задач поддержки принятия решений:

- оценка текущего и прогнозируемого состояния объекта управления и (или) среды его функционирования;
- обнаружение и исследование закономерностей, факторов, тенденций и взаимосвязей локальных процессов;
- обобщение информации в виде агрегации и интеграции сведений различного характера;
- формирование альтернативных решений с выбором оптимального решения;
- моделирование процесса эволюции состояния объекта в нестационарной неоднородной среде.

Повышение качества процессов обобщения может быть достигнуто автоматизацией процедуры сбора информации с количественной оценкой факторов и использованием опыта руководителей.

Перечисленные ключевые возможности аналитических технологий позволяют рассматривать их как аналитическую и информационную основу не только процессов обобщения и анализа информации о текущем и прогнозируемом состояниях объекта управления и среды, но и процессов выработки и оценки вариантов решений.

Следовательно, аналитическая технология — это средство решения задач анализа и прогнозирования состояния объектов в нестационарных средах на основе накопления «истории», ретроспективного анализа накапливаемых динамических данных и формирования (улучшения) формальных моделей явлений и процессов. Основным содержанием ИАД является, таким образом, автоматизированный прецедентный анализ сверхбольших объемов ретроспективной информации, сконцентрированных в информационных хранилищах.

Основная задача ИАД — вскрытие неявных закономерностей, взаимозависимостей и факторов влияния в интересах задач ситуационного анализа и прогноза на основе широкого спектра математических инструментов, включающих в себя как фундаментальные

Рис. 7.1. Основные методы ИАД

статистические методы регрессионного и многофакторного дисперсионного анализа, нейронные сети, эволюционное моделирование, так и методы неформального, качественного анализа, опирающегося на субъективные знания экспертов и эвристические приемы их работы (рис. 7.1).

ИАД (Data Mining) как мультидисциплинарная область возникла и развивается на базе таких наук, как прикладная статистика, распознавание образов, искусственный интеллект, теория баз данных. ИАД — это процесс обнаружения в «сырых» данных неочевидных (ранее неизвестных, нетривиальных), объективных, практически полезных и доступных интерпретации знаний, необходимых для принятия решений в различных сферах человеческой деятельности.

Неочевидные закономерности не обнаруживаются стандартными методами обработки информации или экспертным путем.

Объективные закономерности, в отличие от субъективного экспертного мнения, будут полностью соответствовать действительности.

Практически полезные закономерности имеют конкретное значение, которому можно найти практическое применение.

Извлечение полезных сведений невозможно без хорошего понимания сути данных. Необходимы тщательный выбор модели и интерпретация зависимостей или шаблонов, которые обнаружены. Поэтому работа с такими средствами требует тесного сотрудничества между экспертом в предметной области и специалистом по инстру-

ментарию ИАД. Построенные модели должны быть грамотно интегрированы в бизнес-процессы для возможности оценки и обновления моделей.

Успешный анализ требует качественной предобработки данных. По утверждению аналитиков и пользователей баз данных процесс предобработки может занять до 80 % всего ИАД-процесса. С помощью ИАД можно отыскивать действительно ценную информацию, которая вскоре даст большие дивиденды в виде финансовой и конкурентной выгоды. Многие специалисты утверждают, что ИАД-средства могут выдавать огромное количество статистически недостоверных результатов. Чтобы этого избежать, необходима проверка адекватности полученных моделей на тестовых данных.

Качественная ИАД-программа может быть достаточно дорогой для компании. Вариантом служит приобретение уже готового решения с предварительной проверкой его использования.

Средства ИАД теоретически не требуют наличия строго определенного количества ретроспективных данных. Эта особенность может стать причиной обнаружения недостоверных (ложных) моделей и принятия на их основе неверных решений. Необходимо, следовательно, осуществлять контроль статистической значимости обнаруженных знаний.

Специфика интеллектуального анализа данных определяет применяемые методы. К ним относятся: искусственные нейронные сети, деревья решений, символьные правила, методы ближайшего соседа и k-ближайшего соседа, метод опорных векторов, байесовские сети, линейная регрессия, корреляционно-регрессионный анализ, иерархические методы кластерного анализа, неиерархические методы кластерного анализа (в том числе алгоритмы k-средних и k-медианы, методы поиска ассоциативных правил), метод ограниченного перебора, эволюционное программирование, генетические алгоритмы, разнообразные методы визуализации данных.

Следует отметить, что большинство методов ИАД были разработаны в рамках теории искусственного интеллекта.

Средства интеллектуального анализа данных. Сочетание прецедентного анализа ретроспективных данных с возможностями прогностических методов позволяет рассматривать средства ИАД как составную и наиболее важную часть систем поддержки принятия решений (СППР), способных осуществлять подготовку и количественную оценку возможных вариантов решений.

OLTP-системы представляют собой базы данных (БД), которые реализуются, как правило, на основе стандартных систем управления базами данных. Если OLTP-систему использовать для обработки ретроспективной информации, то возникает ряд сложностей:

- распределенное хранение в нескольких разнородных БД существенно затрудняет комплексное использование этих данных из-за сложности их одновременной выборки из нескольких источ-

ников, возможных пересечений перечней хранимых данных и нарушений их согласованности (целостности);

- хранение ретроспективных данных резко увеличивает объем БД и снижает ее быстродействие;
- глубина ретроспективного анализа определяется ограничениями, принятыми для длительности периода архивного хранения данных.

В связи с этим ретроспективные данные выделяются из БД с текущими данными и помещаются в *хранилище данных* (ХД).

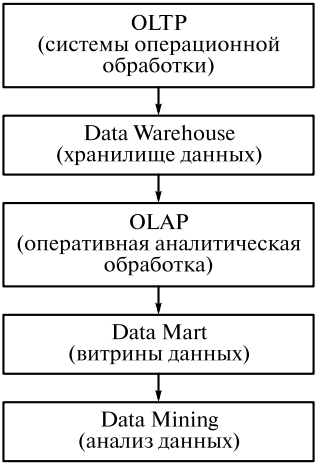


Рис. 7.2. Совокупность средств АИС

ХД (Data Warehouse) — предметно-ориентированный, интегрированный, неизменчивый, поддерживающий хронологию набор данных, организованный для целей поддержки управления. В силу огромного количества информации в ХД требуется введение высокоскоростных методов доступа к данным. Эти методы составляют технологию оперативной аналитической обработки данных (OLAP) и значительно отличаются от методов, используемых в БД (рис. 7.2).

К числу главных преимуществ ХД относят:

- единый источник выверенной информации для информационной среды с единым интерфейсом, унифицированными структурами хранения, общими справочниками и другими корпоративными стандартами, на которой будут строиться все справочно-аналитические приложения предметной области;
- физические структуры хранилища оптимизированы для выполнения произвольных выборок, что позволяет строить быстрые системы запросов;
- интеграция данных из разных источников уже выполнена и потому не надо каждый раз производить соединение данных для запросов информации из нескольких источников;
- информационное хранилище данных нацелено на долговременное хранение информации за период 10 — 15 лет с адаптацией хранимой информации к изменениям структуры и параметров: появляется возможность осуществлять исторический анализ информации;
- разделение информационного хранилища и OLTP-систем существенно снижает нагрузку на OLTP-системы со стороны аналитических приложений.

Основные принципы оперативной аналитической обработки данных и интеллектуального анализа данных сформулировал в 1993 г.

Е. Ф. Кодд. Позднее они были оформлены в так называемый FASMI-тест, требующий, чтобы OLAP-приложение предоставляло возможности быстрого анализа разделяемой многомерной информации:

- Fast (быстрый) — анализ должен производиться одинаково быстро по всем аспектам информации;
- Analysis (анализ) — должна быть возможность осуществлять основные типы числового и статистического анализа, предопределенного разработчиком приложения или произвольно определяемого пользователем;
- Shared (разделяемой) — множество пользователей должно иметь доступ к данным, при этом необходимо контролировать доступ к конфиденциальной информации.
- Information (информации) — приложение должно иметь возможность обращаться к любой нужной информации, независимо от ее объема и места хранения;
- Multidimensional (многомерной) — это основная, наиболее существенная характеристика OLAP.

Хранилища данных используют многомерную модель данных (ММД), у которой имеются три разновидности: MOLAP, ROLAP, HOLAP.

MOLAP предполагает формирование так называемого многомерного куба (гиперкуба). Примером куба может служить система прямоугольных координат, по осям которой откладываются шифры документов, индексы, а на их пересечении находится числовое значение. (На практике многомерная модель применяется в АТС, в которой по каждой из семи (десяти) координат фиксируются цифры от 0 до 9.)

Каждая координата куба — *измерение*. Измерения реализуются с помощью индексов, которые позволяют резко увеличить скорость доступа к данным. По разным оценкам скорость доступа в ММД в 10 — 100 раз выше, чем в реляционных моделях данных. Значение результата в многомерной модели помещается на пересечении (в ячейке) соответствующих измерений.

Аналогом такой модели можно считать задание функции, например $z = f(x, y)$. В ММД координаты x и y суть измерения, а z — значение-результат. Как и в функции нескольких переменных, в ММД должна быть предусмотрена возможность «движения» как внутри одного измерения, так и перехода с одного измерения на другое (например, от «линии» к «плоскости»).

За повышенную скорость доступа приходится «платить» увеличенным объемом памяти. Соотношение между полезным и требуемым объемами памяти в многомерной модели данных в 5 — 10 раз больше, чем в реляционной модели данных. Кроме того, в ММД имеется много пустых ячеек и требуются специальные эффективные программы хранения значения NULL. Желательно удалять пустые ячейки. В многомерной модели выполняются четыре основные операции: агрегация (свертка) данных; дезагрегация (детализация) дан-

ных; сечение (при фиксированных значениях одного или нескольких измерений); вращение (для двумерного случая это аналог транспонирования матрицы).

В **ROLAP** используются реляционные «составляющие» с двумя типами таблиц. Основной является фактографическая таблица, в которой собственно и хранятся данные. Ей подчинено несколько справочных таблиц, каждая из которых характеризует одну из размерностей куба. Возможны две схемы структуры: «снежинка» и «звезда». В «снежинке» справочные таблицы имеют дополнительные подчиненные им таблицы. В «звезде» справочные таблицы не имеют подчиненных таблиц. При использовании схемы «звезда» требуется провести денормализацию данных.

ROLAP позволяет дать характеристику размерности хранилища данных (табл. 7.1).

Сравнение MOLAP и ROLAP дает следующие результаты:

- MOLAP обладает высоким быстродействием, но возникают проблемы с хранением больших объемов данных;
- ROLAP не имеет ограничений на объем данных, однако обладает гораздо меньшим быстродействием.

HOLAP совмещает достоинства MOLAP и ROLAP. Дело в том, что все данные ХД одновременно никогда не требуются. Каждый раз используется лишь их часть. В связи с этим целесообразно данные разделить на предметные подобласти, которые называют *киосками* (магазинами, витринами) *данных*.

Киоск данных — специализированное тематическое хранилище, обслуживающее одно из направлений деятельности фирмы. В этом случае центральное хранилище может быть реализовано с использованием реляционной БД, а основные данные хранятся в многочисленных киосках.

Имеются следующие варианты киосков данных:

- усеченное ХД — хранилище, относящееся только к некоторой тематике (например, экономика и финансы предприятия, производство продукции);
- ХД, создаваемое для решения частной задачи и уничтожаемое после того, как необходимость в нем отпала.

Таблица 7.1. **Размерность хранилища данных**

Размерность	Предельный объем, Гбайт	Число строк в фактографической таблице, млн
Маленькая	3	Несколько
Средняя	25	100
Большая	200	Более 100
Сверхбольшая	Более 200	Более 1 000

При создании ХД выполняются следующие работы:

- формируется состав итоговой информации с предельно-допустимым временем отклика и предельным сроком хранения детальной информации;
- определяется предполагаемый набор запросов на основе детальных данных. При этом следует найти компромисс между созданием итоговой статистической информации и ее вычислениями на основе детальных данных;
- выбирается способ хранения данных «время» в таблицах;
- осуществляется выбор СУБД, который должен учесть и потребности системы OLTP. Наиболее подходящей является объектно-ориентированная СУБД с использованием многомерной модели данных MOLAP. Определяются размерности модели данных. В то же время можно использовать и реляционную СУБД с применением разновидности ROLAP. Следует выбрать схему («звезда» или «снежинка»). Тогда полезно построить таблицу фактов и сопровождающие ее справочные таблицы с минимальным изменением ключей в них. При использовании схемы «звезда» следует провести денормализацию;
- определяются потребности в дополнительных данных, отсутствующих в OLTP, и удаляются ненужные, лишние, столбцы в детальных данных.

Задачи интеллектуального анализа данных. ИАД позволяет решать следующие задачи.

В результате решения задачи *классификации* обнаруживаются признаки, которые характеризуют классы объектов исследуемого набора данных. По этим признакам новый объект можно отнести к тому или иному классу.

Кластеризация является логическим продолжением идеи классификации и отличается тем, что классы объектов изначально не определены. Результатом кластеризации является разбиение объектов на группы.

В ходе решения задачи поиска *ассоциативных* правил отыскиваются закономерности связи между несколькими событиями, происходящими одновременно.

Последовательность позволяет найти временны́е закономерности между транзакциями, позволяя установить закономерности в цепочке событий, связанных во времени. Ассоциация является частным случаем последовательности с временным лагом, равным нулю.

В результате решения задачи *прогнозирования* на основе особенностей исторических данных оцениваются пропущенные или же будущие значения целевых численных показателей.

Определение отклонений или *выбросов* (*анализ отклонений*, или *выбросов*) — обнаружение и анализ данных, наиболее отличающихся от общего множества данных, выявление так называемых нехарактерных шаблонов.

Задача *оценивания* сводится к предсказанию непрерывных значений признака.

Анализ связей — задача нахождения зависимостей в наборе данных.

В результате *визуализации* графическими методами создается графический образ анализируемых данных.

Подведение итогов — задача, целью которой является описание конкретных групп объектов из анализируемого набора данных.

7.2. Последовательность реализации процесса интеллектуального анализа данных

Процесс ИАД является своего рода исследованием. Он неразрывно связан с принятием решений и включает в себя следующие этапы: анализ предметной области; постановка задачи; подготовка данных; построение моделей; проверка и оценка моделей; выбор модели; применение модели; коррекция и обновление модели.

Анализ предметной области. В процессе изучения предметной области должны быть созданы предпосылки для формирования текстовой модели.

Постановка задачи. Этап включает в себя формулировку и формализацию задачи.

В формулировку задачи входит описание статического и динамического поведения исследуемых объектов. Описание статики подразумевает описание объектов и их свойств, описание динамики — поведение объектов и причины такого поведения. Иногда этапы анализа предметной области и постановки задачи объединяют в один этап.

Подготовка данных. Цель этапа — разработка базы данных. Подготовка данных является важнейшим этапом, от качества выполнения которого зависит возможность получения качественных результатов всего процесса. На этап подготовки данных может быть потрачено до 80 % всего времени, отведенного на проект.

Этап может состоять из двух или трех стадий.

Стадия 1. Выявление закономерностей (свободный поиск). Осуществляется исследование набора данных для поиска скрытых закономерностей[1].

Свободный поиск предполагает выявление закономерностей условной логики и выявление закономерностей ассоциативной логики; выявление трендов и колебаний.

---

1 ***Закономерность*** — существенная и постоянно повторяющаяся взаимосвязь, определяющая последовательность и формы процесса становления, развития различных явлений или процессов.

Стадия 2. Использование выявленных закономерностей для предсказания неизвестных значений (прогностическое моделирование). В процессе прогностического моделирования решают задачи классификации и прогнозирования.

При решении задачи классификации результаты работы первой стадии (индукции правил) используют для отнесения нового объекта, с определенной уверенностью, к одному из известных, предопределенных классов.

При решении задачи прогнозирования результаты первой стадии (определение тренда или колебаний) используют для предсказания неизвестных (пропущенных или же будущих) значений целевой переменной (переменных).

Иногда дополнительно вводят стадию валидации для проверки достоверности найденных закономерностей.

Стадия 3. Анализ исключений предназначен для выявления и объяснения аномалий (отклонений) в найденных закономерностях. Для выявления отклонений необходимо определить норму, которая рассчитывается на стадии свободного поиска.

На этапе подготовки данных используются следующие методы:

- методы хранения данных в явном детализированном виде;
- методы выявления формализованных закономерностей, или дистилляция шаблонов — один образец (шаблон) информации извлекается из исходных данных и преобразуется в некие формальные конструкции;
- логические методы, включающие в себя нечеткие запросы и анализы;
- символьные правила, деревья решений, генетические алгоритмы;
- методы кросс-табуляции, такие как агенты, байесовские (доверительные) сети, кросс-табличная визуализация;
- методы на основе уравнений — статистические методы и искусственные нейронные сети, эволюционное программирование; ассоциативная память (поиск аналогов, прототипов); нечеткая логика; системы обработки экспертных знаний.

Этап подготовки данных включает в себя определение и анализ требований к данным; определение необходимого количества данных.

При определении необходимого количества данных следует учитывать упорядоченность данных. Если данные упорядочены и идет работа с временны́ми рядами, желательно знать, включает ли такой набор данных сезонную (цикличную) компоненту. В случае присутствия этой компоненты необходимо иметь данные, как минимум, за один сезон (цикл).

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

- количество записей в наборе данных должно быть значительно больше количества факторов;

- недостаточное количество записей в наборе данных может стать причиной построения некорректной модели;
- алгоритмы, используемые для построения моделей на сверхбольших базах данных, должны быть масштабируемыми;
- набор данных должен быть репрезентативным и представлять как можно больше возможных ситуаций;
- анализировать можно как качественные, так и некачественные данные.

Для обеспечения качественного анализа необходимо провести предварительную обработку данных.

Данные, полученные в результате сбора, должны соответствовать критериям качества данных, определяющим полноту, точность, своевременность и возможность интерпретации данных.

Данные высокого качества — полные, точные, своевременные данные, поддающиеся интерпретации и поддерживающие процесс принятия решений.

Данные низкого качества — грязные, или «плохие», данные: отсутствующие, неточные или бесполезные с точки зрения практического применения (например, представленные в неверном формате, не соответствующем стандарту). Грязные данные могут появиться из-за ошибок при вводе данных, использования иных форматов представления или единиц измерения, несоответствия стандартам, отсутствия своевременного обновления, неудачного обновления всех копий данных, неудачного удаления записей-дубликатов.

Наличие грязных данных может привести к финансовым потерям и правовой ответственности, если их появление не предотвращается или они не обнаруживаются и не очищаются.

Специальные средства очистки не могут справиться со всеми видами грязных данных. Наиболее распространенными видами грязных данных являются пропущенные значения, дубликаты данных, шумы и выбросы.

Некоторые значения данных могут быть пропущены в связи с тем, что данные вообще не были собраны или применяемые атрибуты не подходят для взятых объектов. С пропущенными значениями можно бороться следующим образом:

- исключая объекты с пропущенными значениями из обработки;
- рассчитывая новые значения для пропущенных данных;
- игнорируя пропущенные значения в процессе анализа;
- заменяя пропущенные значения на возможные.

Наличие дубликатов записей с одинаковыми значениями всех атрибутов в наборе данных может явиться способом повышения значимости некоторых записей. Однако в большинстве случаев продублированные данные — результат ошибок при подготовке данных.

Существует два варианта обработки дубликатов. В первом варианте удаляется вся группа записей, содержащая дубликаты. Этот вариант используется в том случае, когда наличие дубликатов вы-

зывает недоверие к информации и полностью ее обесценивает. Второй вариант состоит в замене группы дубликатов на одну уникальную запись.

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

Достаточно распространена практика проведения двухэтапного анализа: анализ с шумами и выбросами и с их отсутствием; сравнение полученных результатов.

Различные методы ИАД обладают разной чувствительностью, что необходимо учитывать при выборе метода анализа данных. Некоторые инструменты ИАД имеют встроенные процедуры очистки от шумов и выбросов.

Очевидно, что результаты ИАД на основе грязных данных не могут считаться надежными и полезными. Вместе с тем всегда должен быть разумный выбор между наличием грязных данных и стоимостью и/или временем, необходимым для их очистки.

В процессе очистки данных выявляют и удаляют ошибки и несоответствия в данных для улучшения качества данных. Когда интеграции подлежит множество источников данных, необходимость в очистке данных существенно возрастает: источники часто содержат разрозненные данные в различном представлении. Для обеспечения доступа к точным и согласованным данным необходимы консолидация представлений данных и исключение повторяющейся информации.

Специальные средства очистки обычно предназначены для работы с конкретными областями (имена и адреса) или для исключения дубликатов. Преобразования обеспечиваются либо в форме библиотеки правил, либо пользователем в интерактивном режиме. Преобразования данных могут быть автоматически получены с помощью средств согласования схемы.

Метод очистки данных должен выявлять и удалять все основные ошибки и несоответствия как в отдельных источниках данных, так и при интеграции нескольких источников. Метод должен поддерживаться определенными инструментами, чтобы сократить объемы ручной проверки и программирования, и быть гибким в работе с дополнительными источниками.

Очистка данных не должна выполняться в отрыве от схемы преобразования сложных метаданных. Функции маппирования для очистки и других преобразований данных должны быть определены декларативным образом и подходить для использования в других источниках данных и в обработке запросов.

Очистка данных включает в себя этапы анализа данных; определения порядка и правил преобразования данных; подтверждения; преобразования; противотока очищенных данных.

Построение модели. После окончания этапа подготовки данных можно переходить к построению модели.

С одной стороны, можно говорить, что построенная модель выделила наиболее существенные (значимые) факторы с точки зрения решаемой задачи. С другой стороны, модель может обладать свойством неполноты. Примером неучтенного фактора могут быть, например, природные катаклизмы, которые повлияли на желание клиента пользоваться услугами туристического агентства.

Идеальной модели, которая бы позволила решать разнообразные задачи, не существует. Поэтому многие разработчики включают в инструменты ИАД возможность построения различных моделей или обеспечивают возможность расширяемости моделей. Некоторые инструменты ИАД создаются специально для конкретных областей применения.

Иногда для выявления искомых закономерностей требуется использовать несколько методов и алгоритмов. В таком случае одни методы используют в начале моделирования, другие — в дальнейшем.

Выбор метода, на основе которого будет построена модель, должен осуществляться с учетом постановки задачи, особенностей набора исходных данных, специфики решаемой задачи, результатов, которые должны быть получены на выходе.

Проверка и оценка моделей. Проверка модели подразумевает проверку ее достоверности или адекватности и заключается в определении степени соответствия модели реальности. Адекватность модели — соответствие модели моделируемому объекту или процессу. Понятия «достоверность» и «адекватность» являются условными, поскольку невозможно рассчитывать на полное соответствие модели реальному объекту. В противном случае это был бы сам объект. В связи с этим в процессе моделирования следует учитывать адекватность не модели вообще, а именно тех ее свойств, которые являются существенными с точки зрения проводимого исследования.

Проверка модели подразумевает также определение той степени, в которой она действительно помогает менеджеру при принятии решений. Оценка модели подразумевает проверку ее правильности и осуществляется путем тестирования. Оно заключается в «прогонке» построенной модели с данными для определения ее характеристик и проверки работоспособности.

Тестирование модели включает в себя проведение множества экспериментов. На вход модели могут подаваться выборки различного объема. С точки зрения статистики точность модели возрастает с увеличением количества исследуемых данных. Алгоритмы, являющиеся основой для построения моделей на сверхбольших базах данных, должны обладать свойством масштабирования.

Если модель сложна и требует много времени на обучение и последующую оценку, то можно построить и протестировать модель на небольшой части выборки. Однако этот вариант подходит только для однородных данных. В противном случае необходимо использовать все доступные данные.

Выявленные на модели соотношения и закономерности должны быть проанализированы экспертом в предметной области. Если результаты полученной модели эксперт считает неудовлетворительными, следует вернуться на один из предыдущих шагов подготовки данных, построения модели. Если результаты моделирования эксперт считает приемлемыми, ее можно применять для решения реальных задач.

Выбор модели. Если в результате моделирования было построено несколько разных моделей, то на основании их оценки необходимо выбрать лучшую из них.

Основные характеристики модели, определяющие ее выбор, — точность и эффективность работы алгоритма.

В некоторых программных продуктах реализован ряд методов, разработанных для выбора модели. Многие из них основаны на так называемой конкурентной оценке моделей: формирование для одного и того же набора данных различных моделей и последующее сравнения их характеристик.

Применение модели. На этом этапе выбранная модель используется применительно к новым данным для решения задач, поставленных в начале проектирования процесса ИАД.

Коррекция и обновление модели. Через установленный промежуток времени с момента использования модели следует проанализировать полученные результаты и определить, действительно ли она успешно работает или возникли проблемы и сложности в ее функционировании.

Следует учитывать возможность изменения внешних факторов. При появлении новых данных требуется повторное обучение модели, т.е. обновление модели. Работы, проводимые с моделью на этом этапе, — контроль и сопровождение модели.

7.3. Реализация интеллектуального анализа данных в форме автоматизированных информационных систем

Автоматизированные информационные системы (АИС) ориентированных на поддержку принятия решения.

В общем случае АИС можно рассматривать как организационно-технический комплекс автоматизированной поддержки принятия решений, создаваемый для стратегического анализа ситуаций и про-

гнозирования в разных областях управленческой и исследовательско-аналитической деятельности.

Основными условиями, необходимыми для реализации содержательных возможностей АИС по сбору и аналитическому анализу информации, являются следующие:

- наличие совокупности математических методов, адекватных задачам анализа информации и специфике обрабатываемых данных;
- организация хранения больших массивов фактографических и документальных сведений и оперативного доступа к ним с использованием гибкой системы;
- организация сбора и хранения сведений, характеризующих управляемый объект и среду его функционирования с требуемой достоверностью, полнотой и непротиворечивостью;
- наличие совокупности программных и технических средств, обеспечивающих функционирование АИС.

В качестве алгоритмического инструментария АИС выступает комплекс методов и средств стратегического анализа и прогнозирования сложных нестационарных процессов, включая процессы со скачкообразным изменением состояния. Комплекс основан на сочетании классических методов прогнозирования (экстраполяция, статистические методы предсказания) и новых, но уже известных методов прогностики (эволюционное моделирование, нейронные сети и т.д.).

Создание информационной системы базируется на концепции и технологии ХД и OLAP.

Реализация вычислительной среды и средств сбора информации базируется на известных технологиях и средствах.

Процессы функционирования АИС формализуются как три взаимосвязанных уровня взаимодействия:

1) стратегический (управляющий) — уровень управления в рамках динамически формулируемых конкретных проблем анализа и прогностики;

2) оперативный (основной) — уровень отдельных задач в рамках общей проблемы, на котором по отношению к каждой задаче выполняются этапы постановки, анализа и решения;

3) технологический (обеспечивающий) — уровень функционирования служб АИС, составляющих ее технологические профили (телекоммуникации, разграничение доступа, защита информации, архивация результатов решения задач).

Функции пользователей АИС разграничены. Их выполнение возложено на следующих специалистов:

- лицо, принимающее решения (ЛПР), — руководитель осуществляет постановку проблем и задач и является для АИС конечным потребителем получаемых результатов;
- предметный эксперт (ПЭ) — специалист предметной области, несущий основную нагрузку по определению состава и по содержа-

тельному анализу предметных сведений и результатов их обработки;

- эксперт-аналитик (ЭА) — специалист, осуществляющий выбор методик решения задач и выполняющий решение задач на основе формальных методов обработки;
- администратор данных (АД) — лицо, выполняющее действия по сбору необходимых предметных сведений и помещению их в информационное хранилище;
- администратор системы — лицо, управляющее службами АИС, ее конфигурированием и содержательным наполнением.

Взаимодействие ПЭ и ЭА осуществляется путем их совместного участия в решении задач с последовательным уточнением и согласованием каждого этапа. Детальная архитектура любой АИС и методология ее создания определяются конкретными техническими решениями.

В сложных, нестационарных ситуациях, характеризующихся большим числом разнообразных гетерогенных взаимосвязанных факторов, эксперты, как правило, не находят рациональных решений. Мнения экспертов оказываются субъективными и противоречивыми. В большинстве случаев мозг эксперта не способен экстраполировать развитие ситуаций, находящихся под воздействием более 3—5 независимых факторов. Для взаимосвязанных воздействий даже опытный эксперт не способен корректно учесть более трех факторов влияния. В то же время большинство реальных стратегических ситуаций требуют учета, как минимум, от 6 до 20 значимых факторов влияния.

Можно ожидать, что внедрение аналитических информационных систем позволит существенно повысить технологический уровень существующих и вновь создаваемых аналитических служб и центров.

В то же время следует подчеркнуть, что содержательное наполнение аналитических информационных систем в большой степени зависит от специфики предметной области и объекта управления. Формальных методов определения наполнения в настоящее время не существует и выбор конкретных математических методов и обрабатываемых данных осуществляется исключительно эвристически с последующим их уточнением в ходе применения системы.

К создаваемой АИС предъявляют следующие требования.

1. *Наличие интуитивного интерфейса*. Интерфейс — среда передачи информации между программной средой и пользователем. Интерфейс подразумевает грамотное расположение различных элементов: блоков меню, информационных полей, графических блоков, блоков форм.

Для удобства работы пользователя необходимо, чтобы интерфейс был интуитивным. Это позволяет ему легко и быстро воспринимать элементы интерфейса, вследствие чего диалог программная среда — пользователь становится проще и доступнее. Интуитивный интер-

фейс подразумевает знакомую окружающую среду и наличие понятной терминологии (например, для сообщения пользователю о совершенной ошибке).

2. *Удобство экспорта/импорта данных*. При работе с инструментом ИАД-пользователь часто применяет разнообразные наборы данных, работает с различными источниками данных. Это могут быть текстовые файлы, файлы электронных таблиц, файлы баз данных. Инструмент ИАД должен иметь удобный способ загрузки (импорта) данных. По окончании работы пользователь также должен иметь удобный способ выгрузки (экспорта) данных в удобную для него среду. Программа должна поддерживать наиболее распространенные форматы данных.

3. *Наглядность и разнообразие получаемой отчетности*. Подразумевается получение отчетности в терминах предметной области, в качественно спроектированных выходных формах.

4. *Легкость обучения работы с инструментарием*.

5. *Прозрачные и понятные шаги ИАД-процесса*.

6. *Удобное руководство пользователя с пошаговым описанием шагов генерации моделей ИАД*.

7. *Удобство и простота использования*. Существенно облегчает работу начинающего пользователя возможность использовать программу Мастер.

8. *Наличие руссифицированной версии инструмента и документации*.

9. *Наличие демонстрационной версии* (*демоверсии*) *с решением конкретного примера*.

10. *Возможности визуализации*. Графическое представление информации существенно облегчает интерпретируемость полученных результатов.

11. *Наличие значений параметров*, *заданных по умолчанию*. Это особенно важно для начинающих пользователей. Особенно много таких параметров в инструментах, реализующих метод нейронных сетей. В нейросимуляторах чаще всего заранее заданы значения основных параметров и часто неопытным пользователям или пользователям, недостаточно овладевшим используемым программным продуктом, не рекомендуется на начальных стадиях работы изменять эти значения.

12. *Разнообразие методов и алгоритмов компьютерной реализации*. Во многих инструментах Data Mining реализовано сразу несколько методов, позволяющих решать одну или несколько задач. Пользователь получает возможность сравнивать характеристики моделей, построенных при помощи разных методов.

13. *Высокая скорость вычислений и высокая скорость представления результатов*.

14. *Консультационная поддержка квалифицированным консультантом-ассистентом по выбору методов и алгоритмов*.

15. *Возможности поиска*, *сортировки*, *фильтрации данных*. Такая возможность полезна как для входных данных, так и для выходной информации. Применяется сортировка по различным критериям (полям).

При условии фильтрации входных данных появляется возможность построения модели ИАД на одной из выборок набора данных. Фильтрация выходной информации полезна с точки зрения интерпретации результатов. Например, при построении деревьев решений результаты получаются слишком громоздкими, и здесь могут оказаться полезными функции как фильтрации, так и поиска и сортировки. Дополнительное удобство для пользователя предоставляет цветовая подсветка некоторых категорий записей.

16. *Защита*, *пароль*. Очень часто при помощи Data Mining анализируется конфиденциальная информация, поэтому наличие пароля доступа в систему является желательной характеристикой для инструмента.

17. *Разнообразие платформ для поддержания работы инструмента*.

В создании и использовании АИС принимают участие следующие специалисты:

- администратор баз данных — специалист, имеющий знания о том, где и каким образом хранятся данные, как получить к ним доступ и связать эти данные между собой;
- специалист по извлечению данных — специалист по анализу данных, который владеет, как минимум, основами статистических знаний и может осуществлять постановку задач;
- дополнительно вовлекаются следующие специалисты: менеджер проектов, специалист по архитектуре информационных технологий, специалист по архитектуре решений; специалист по архитектуре данных, специалист по моделированию данных, эксперт ИАД, деловой аналитик.

7.4. Состояние и перспективы применения интеллектуального анализа данных в научных исследованиях

Одним из эффективных направлений применения ИАД являются научные исследования. ИАД в части реализации тесно связан с системами поддержки принятия решений (СППР). В основу таких систем могут быть положены алгоритмы любого из рассмотренных классов интеллектуальных систем, однако чаще применяют классы МАС, ИНС, ЭС. В то же время результаты исследований ИАД могут быть широко применены и в системах на естественном языке.

Необходимость повышения эффективности систем поиска и анализа информации (оперативность и обоснованность получаемых ре-

шений) обусловливает быструю динамику их развития на научно-методологическом, технологическом и аппаратно-программном уровнях.

Анализ результатов разработок в области информационно-поисковых и аналитических систем позволяет выявить основные тенденции их развития. Этапы эволюции поисковых систем могут быть представлены в виде табл. 7.2, из которой видна явно выраженная тенденция интеллектуализации как механизмов поиска, так и аналитической обработки текстов.

Так, Web-поисковые системы имеют дополнительные возможности организации поиска на уровне метаданных с применением ряда методов статистического анализа и масштабируемости; а поисковые системы 3-го поколения предоставляют дополнительные возможности семантического и контекстного анализа как структурированных, так и неструктурированных, полуструктурированных XML-данных.

Новая парадигма представления смыслового содержания отдельных документов и их тематических ассоциаций на основе онтологий является перспективным направлением эффективной организации информационных ресурсов человечества.

Механизм онтологий позволяет унифицировать на единой методологической основе внешнюю и внутреннюю среды поиска релевантной информации, достичь значительного ускорения поиска за счет формирования и использования семантического индекса про-

Таблица 7.2. **Этапы эволюции поисковых систем**

Составляющие	1-е поколение — информационный поиск	2-е поколение — Web-поиск	3-е поколение — Text Mining
Пользователи	Эксперт	Любой человек	Любой человек или группа (мультиагентность)
Пространство	Малые закрытые коллективные документы	WWW	Структурированная, полуструктурированная и неструктурированная информация
Технологии	Поиск в соответствии с шаблоном с приоритетом важности	Поиск по шаблону с анализом ссылок для релевантности, ранжирования и классификации	Лингвистическое распознавание, статистический и семантический анализ

странства поиска и автоматической генерации смыслового содержания найденного документа по этому индексу.

Сравнение информационных технологий анализа информации, содержащейся в текстовых документах, представлено в табл. 7.3.

Наметилась тенденция к стандартизации описания структурированных, неструктурированных и полуструктурированных текстов при помощи XML-технологии.

В то же время к недостаткам существующих коммерческих ИПС, в первую очередь, следует отнести:

- поиск по ключевым словам дает слишком много часто бесполезных ссылок;
- огромное количество поисковых машин с разными пользовательскими интерфейсами порождает проблему когнитивной перегрузки;
- методы индексирования баз данных, как правило, семантически связаны с информационным содержанием;
- неадекватные стратегии поддержки каталогов часто приводят к тому, что выдаются ссылки на информацию, которой уже нет в Интернете и БД;
- при имеющемся уровне доступа практически невозможно сделать обоснованный вывод о полезности источника.

Данные недостатки заставляют разработчиков информационных систем искать новые принципы их функционирования и новые подходы к организации интеллектуальной поддержки различных технологий работы пользователей.

Такой парадигмой является самоорганизация всех основных элементов ИПС на основе мультиагентной их организации, которая предполагает отказ от алгоритмической парадигмы к изменению состояния элемента системы и переходу к наиболее эволюционно возможному в условиях конфликтного развития ситуаций.

Такая поисковая машина «думает» и действует сама. Пользователь обучает агента, затем агент в Интернете из множества доступных документов выбирает нужные и дает им оценку. Пользователь может в любой момент «отозвать» агента и посмотреть, как продвигается работа, или продолжить его обучение на основе найденной информации. В табл. 7.4 приведены примеры интеллектуальных агентов и их характеристики.

Агенты выполняют ряд инструкций от имени пользователя или другой программы, могут работать независимо и иметь некоторую степень автономности в сети. Между интеллектуальными агентами и Java-апплетами существуют некоторые различия. Java-апплеты загружаются из Интернета и работают на машине пользователя. Агенты способны «понимать», какая именно информация нужна пользователю, могут быть запрограммированы на изменение поведения в зависимости от накопленного опыта и взаимодействий с другими агентами.

Таблица 7.3. **Сравнительный анализ информационных технологий анализа текстов и приложений**

Задача	Технология	Характеристика	*Прикладные системы*
		Существующие	
Поиск	Информационный поиск	Множество запросов и документов	Интеллектуальные системы и управление данными, почтовые приложения, системы поддержки
Предобработка	Классификация документов	Отнесение документов к определенному классу	CRM, почтовые сервера
Предобработка	Кластеризация документов	Выявление группы подобных документов	Text Mining
		Перспективные	
Выявление	Выделение сущностей	Выявление имен и терминов, имен компонент, дат и других характеристик	Подход, основанный на Text Mining
Выявление	Целевой анализ	Выявление значений и параметров в текстах	CRM-интеллектуальные генераторы отчетов
Выявление	Выделение связей	Измерение или выявление ключевых слов и их связей в тексте	Нечеткие интеллектуальные системы нахождения связей, описание ПО на языке онтологий
Подготовка	Построение семантической сети	Выявление содержания	Выявление семантики в XML-данных

Окончание табл. 7.3

Задача	Технология	Характеристика	*Прикладные системы*
Поиск	Управление сценариями поиска	Управление технологиями работы пользователей с помощью естественного языка	Поиск с использованием естественного языка
Поиск	Аннотирование	Генерация краткого содержания документов	Поисковые машины, системы документооборота
Выявление	Перевод	Синтаксические, статистические, семантические модели	Многоязычные системы перевода

Таблица 7.4. **Примеры интеллектуальных агентов и их характеристики**

Интеллектуальные агенты	Что они умеют
Агент «ежедневные новости»	Выполняет автоматический быстрый Web-серфинг в поисках текущей информации. Доставляет важную деловую информацию тем, кто в ней нуждается. Осуществляет мониторинг специализированных внешних Web-сайтов и поиск релевантной информации
Агент «профиль пользователя»	Генерирует профиль интересов пользователя, наблюдая его реакцию на сообщения; запоминает, какие из сообщений отмечались как важные. Запоминает, в какое время дня пользователь предпочитает отвечать на сообщения. Если интересы пользователя меняются, меняется и профиль. Взаимодействует с другими аналогичными агентами, уточняя, какие сообщения считаются важными
Агент «предупреждения о событиях»	Извещает пользователя о событиях через электронную почту, динамические Web-страницы последних новостей и другие «push»-технологии. Осуществляет мониторинг новостей и доставку информации. В случае необходимости распространяет сообщения о важных событиях
Агент рабочих групп	Обеспечивает прозрачное создание пользовательских групп по интересам. Обеспечивает простой обмен данными между агентами. Поддерживает аутентификацию пользователей по входному паролю. Уменьшает затраты, обнаруживая другие организации, занимающиеся аналогичными вопросами. Позволяет осуществлять эффективную передачу знаний в пределах данной организации

Ведущие компьютерные фирмы мира пытаются осуществить сочетание грамматических, статистических и семантических методов для повышения эффективности мультиагентных поисковых машин. Однако без общей архитектуры системы этого не добиться.

Между тем в настоящее время не существует инвариантной архитектуры таких систем, что предопределяет необходимость разработки предметно-ориентированной архитектуры информационной системы накопления и обработки информации о новых достижениях

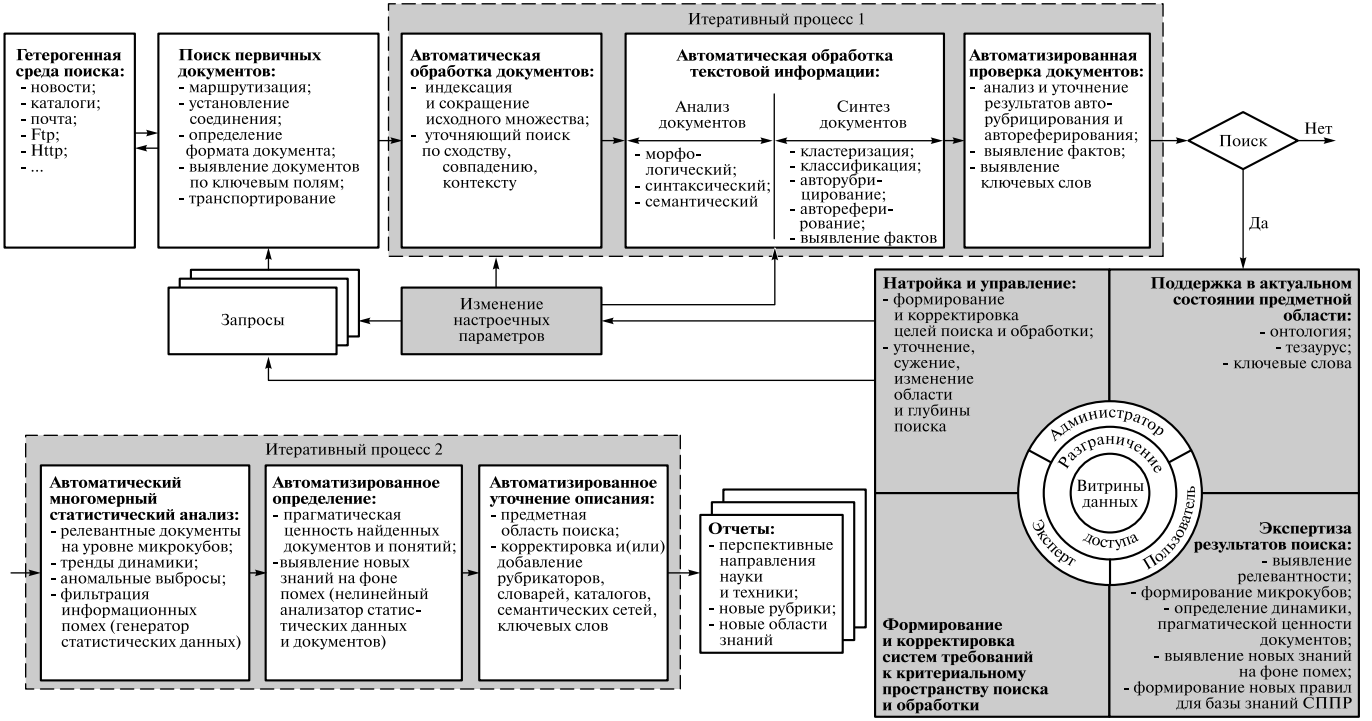


Рис. 7.3. Интеллектуальная технология поиска перспективных направлений науки и техники

науки и техники, базирующейся на выявленных принципах построения и функционирования.

Анализ новых информационных технологий и способов их реализации в информационных системах поиска, накопления и аналитической обработки информации о новых достижениях науки и техники показал необходимость и возможность создания предметно-ориентированных средств автоматизации поддержки принятия решений, сочетающих возможности интеллектуальных информационно-поисковых и аналитических систем.

Осуществление пользователями и экспертами поиска и многомерного анализа результатов поиска документов без должной информационной поддержки характеризуется высокой трудоемкостью и низкой эффективностью и приводит к противоречию между высокими требованиями к оперативности и возможностями существующей технологии поиска релевантных документов. Сложности усугубляются динамическим ростом больших объемов структурированных и неструктурированных документов, наличием недостоверной информации.

Одним из способов разрешения противоречия является предлагаемая интеллектуальная технология поиска перспективных направлений науки и техники, представленная на рис. 7.3. Принципиальная особенность этой технологии — наличие двух взаимосвязанных автоматизированных итеративных процессов, работающих в контуре

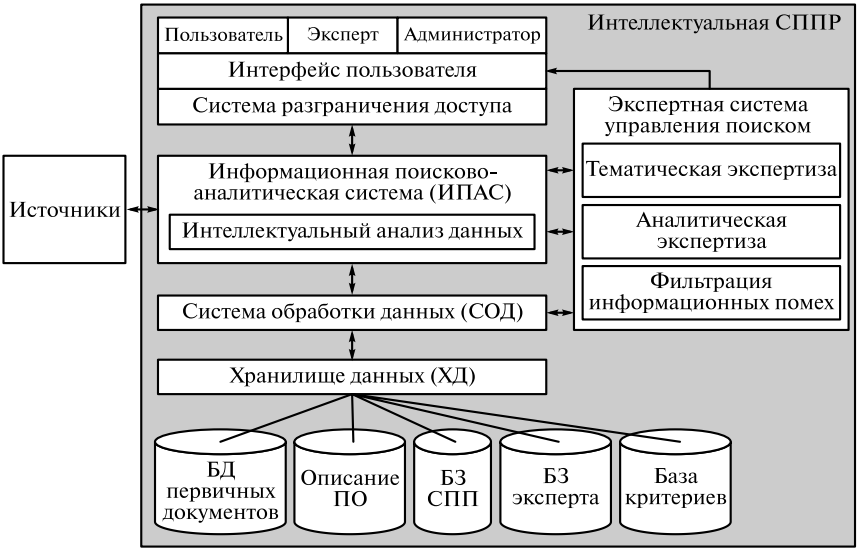


Рис. 7.4. Системы поддержки принятия решений специалиста по поиску перспективных направлений науки и техники

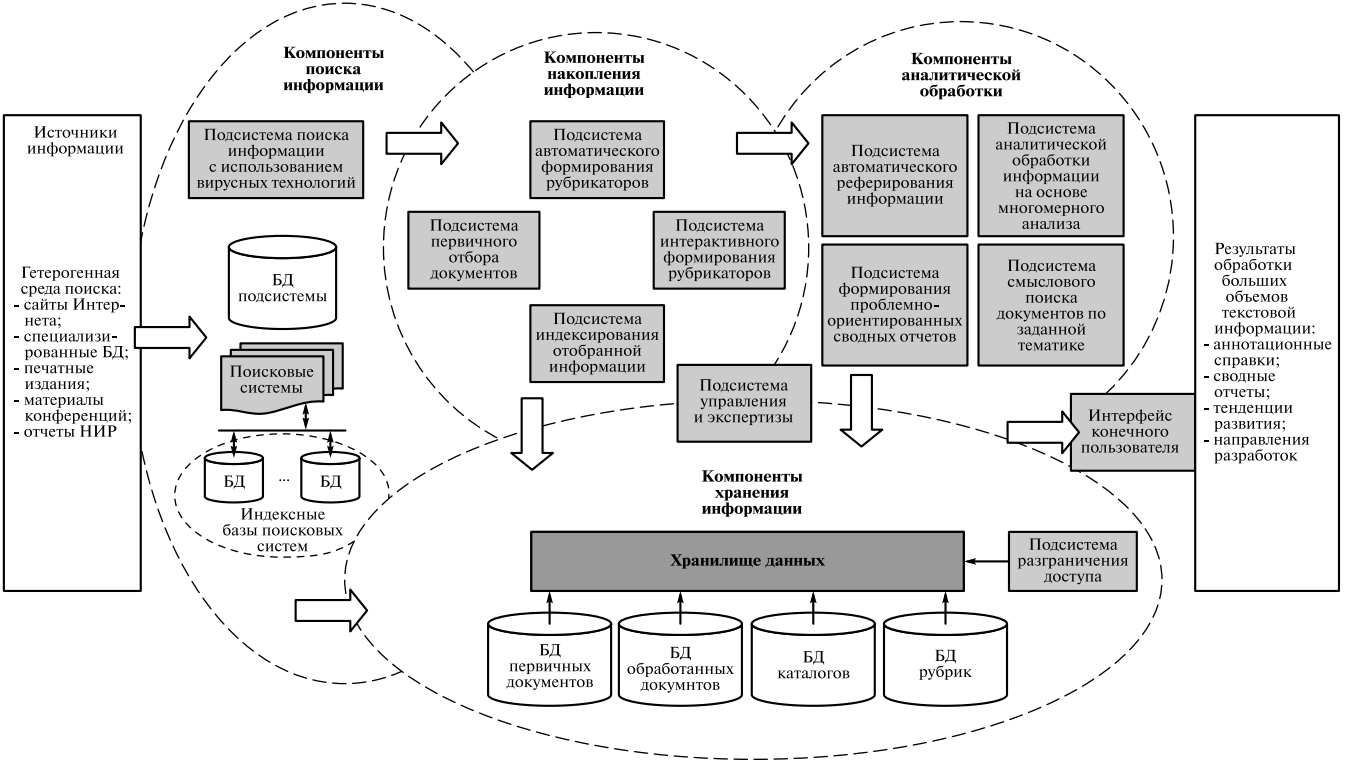


Рис. 7.5. Обобщенная структурная схема интеллектуальной ИПАС

управления поиском релевантных документов. Управление, в свою очередь, информационно поддерживается сервисом автоматизированных экспертиз результатов поиска, решающих задачи выявления релевантности; формирования микрокубов; определения динамики; прагматической ценности документов; выявления новых знаний на фоне помех; формирования новых правил для базы знаний системы поддержки принятия решений (СППР).

Реализацию данной технологии предлагается осуществить в рамках системы поддержки принятия решений специалиста по поиску перспективных направлений науки и техники (рис. 7.4). Ядро данной СППР — интеллектуальная поисково-аналитическая система (ИПАС), структурная схема которой приведена на рис. 7.5.

В основу построения ИПАС положена концепция двухступенчатости направленного автоматизированного поиска с использованием результатов автоматизированного анализа найденных документов для выбора стратегии, направления и глубины поиска релевантных документов.

Двухступенчатая схема организации поиска позволяет обеспечить приемлемое время нахождения релевантного документа за счет сочетания грубых методов поиска первичных документов (по сходству, совпадению, контексту) и тонких языковых методов обработки документов (морфологический, синтаксический, семантический анализ) и их тематических ассоциаций (кластеризация, классификация, авторубрицирование, автореферирование, выявление фактов).

Недостаток решения — существенное сужение исходного пространства поиска с большой вероятностью его смещенного нахождения от центра тематического пространства гетерогенной среды поиска. Для его устранения необходима итеративная процедура автоматизированного совмещения центров тематического и поискового пространств посредством корректировки запросов и параметров механизмов поиска первичных документов. Управление поиском и обработкой предполагается строить на основе автоматизированных экспертиз, необходимый состав которых требуется разработать как по форме, так и по содержанию.

7.5. Структура системы интеллектуального анализа данных в научных исследованиях

Последовательность действий пользователя-эксперта обеспечивается с помощью следующих подсистем информационной системы (ИС): первичного отбора информации; формирования рубрикатора предметной области; накопления отобранной информации; поиска документов по заданной тематике; аналитической обработки накопленной информации.

Подсистема первичного отбора информации. Подсистема обеспечивает:

- поиск научно-технической и патентной информации о новых достижениях науки и техники, организациях-разработчиках в локальной БД, сформированной на основе информации, содержащейся в существующих отечественных и зарубежных базах данных (РОСПАТЕНТ, РФФИ, РАСПРИ), системе Интернет; автоматическое определение языка, формата и типа кодирования входного документа;
- индексирование информации из различных источников с помощью роботов поисковых интернет-машин;
- автоматизированное выявление и сохранение основных характеристик документа (автор, тема, дата создания, источник);
- автоматическое аннотирование информации — выделение основных идей документа (только подокументно);
- интерактивное редактирование автоматически выделенных полей;
- фильтрацию информационного потока документов в соответствии с заданной структурой тематических рубрик;
- интерактивный отбор документов.

В данной подсистеме клиентом являются программы-браузеры просмотра конкретного информационного ресурса типа Netscape Navigator. Такая программа обеспечивает просмотр документов World Wide Web, Gopher, Wais, FTP-архивов, почтовых списков рассылки и групп новостей Usenet. В свою очередь, все эти информационные ресурсы являются объектом поиска информационно-поисковой системы. Разработан вариант Search engine, т.е. специализированной поисковой машины, которая обеспечивает трансляцию запроса пользователя, подготавливаемого на информационно-поисковом языке (ИПЯ), в формальный запрос системы для поиска ссылок на информационные ресурсы сети и выдачу результатов этого поиска пользователю. В связи с использованием в ИС интернет (интранет)-технологий индексация web-ресурсов осуществляется по глобальному индексу, а поиск релевантного документа в пространстве первичнонайденных осуществляется по локальному индексу, формируемому средствами внутренней поисковой машины ИС:

- Index database служит для поиска адреса информационного ресурса. Архитектура индекса устроена таким образом, чтобы поиск происходил максимально быстро и при этом можно было бы оценить ценность каждого из найденных информационных ресурсов сети;
- Index robot — робот-индексировщик служит для сканирования Интернета и поддержки базы данных индекса в актуальном состоянии. Эта программа является основным источником информации о состоянии информационных ресурсов сети;
- WWW sites — весь Интернет. Если говорить точнее, то это те информационные ресурсы, просмотр которых обеспечивается программами просмотра.

В ИС может быть реализован усеченный вариант информационного хранилища (ИХ) на основе высокопроизводительной СУБД с языком запросов SQL.

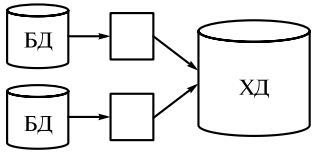


Рис. 7.6. Перегрузка данных:
ХД — хранилище данных

Периодическая загрузка данных в хранилище осуществляется из основных БД ИС (рис. 7.6). Для обработки разнообразных запросов, в том числе и от web-сервера, используется промежуточная БД высокой производительности (рис. 7.7). Информационное наполнение промежуточной БД осуществляется специализированным программным обеспечением на основе содержимого основных баз данных.

Основой повышения производительности обработки web-запросов и резкого увеличения скорости разработки web-интерфейсов является использование внутренних языков СУБД информационного хранилища для создания гипертекстовых документов. Для загрузки содержимого основной БД в информационное хранилище можно использовать любые решения (языки программирования, интегрированные средства), а также специализированные средства перегрузки, поставляемые с SQL-сервером, и продукты поддержки информационных хранилищ.

Подсистема осуществляет формирование массива первичных документов, служащих исходными данными для других подсистем, взаимодействующих через хранилища данных.

Подсистема формирования рубрикатора предметной области.

Подсистема обеспечивает:

- автоматическое отображение терминов предметной области из наборов найденных документов актуальной тематики с заданной глубиной поиска, определяющих структуру рубрикатора и интерактивное изменение тезауруса предметной области;
- интерактивное создание и редактирование структуры и содержания тематических рубрик.

Рубрикатор в ИС выполнен в виде иерархического дерева, состоящего из тематических рубрик (подрубрик) и тезауруса предметной области и реализуется двумя способами:

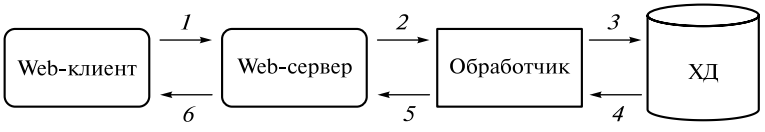


Рис. 7.7. Обработка запросов с использованием промежуточной БД:
1—6 — этапы обработки

1) на основе известной информации о предметной области, что дает возможность сформировать первичное поисковое пространство по имеющейся у эксперта априорной информации о предметной области;

2) на основе новых ключевых слов, найденных в результате обработки релевантных документов, что дает возможность управлять процессом поиска на основе изменения тезауруса подсистемы.

Тематическая структура документа описывается содержанием анализируемых текстов в виде иерархии связанных тем и подтем, раскрывающих содержание тем.

Использование рубрикатора и тезауруса предметной области позволяет осуществлять более полный поиск интересующей эксперта информации в массивах данных по сравнению с традиционными подходами в информационно-поисковых системах.

Подсистема накопления отобранной информации. Подсистема обеспечивает:

- сохранение документа в исходном виде;
- выявление новых словарных терминов предметной области;
- автоматизированное выявление дубликатов документов;
- классификацию информации — отнесение документов к одной или нескольким тематическим рубрикам.

В ИС накопленная информация сохраняется в трех видах: найденные документы в исходном виде (сохранение на диске); поисковый образ в виде инвертированного списка (контекст); каталоги из обработанных экспертом документов, структура которых задана экспертом.

Приписывание весов терминам используется для индексирования в целях выбора лучших представителей для документов.

В ИС реализованы стратегии взвешивания терминов, базирующиеся на частотной и разрешающей способности термина.

Частотностная способность термина. Частотность f_i^k k-го термина в документе D_i есть общее число вхождений данного термина в данный документ. Общее число вхождений термина во все множество документов обозначается F^k: $F^k = \sum_{i=1}^{m} f_i^k$. Вес термина k в документе D_i определяется как $w_{ik} = f_i^k / F^k$.

Определим *частотность документа* F_k^d как общее число документов, содержащих термин k (или представимых термином k):

$$F_k^d = \sum_{i=1}^{m} f_i^d, \quad \text{где} \quad f_i^d = \begin{cases} 1, & \text{если } k \in D_i; \\ 0 & \text{в противном случае.} \end{cases}$$

Тогда w_{ik} может быть определено также через F_k^d как $w_{ik} = f_i^k / F_k^d$, что соответствует весу термина k во всем наборе документов.

Если термин часто встречается в документе, следовательно, он важен для этого документа. Однако если один и тот же термин встречается с высокой частотой во всей базе данных, его важность для отдельного документа уменьшается, особенно в качестве его специфичности. Обе приведенные меры w_{ik} могут выделить термины, встречающиеся с высоким весом в отдельных наборах документов, хотя и имеющие относительно малое значение для всей базы данных.

Взвешивание терминов может быть основано также на теории передачи информации, которая описывает шум как функцию равномерности распределения термина k среди документов. Содержание шума и сигнала в термине k определяется следующим образом:

$$N^k = \sum_{i=1}^{m} \frac{f_i^k}{F^k} \log \frac{F^k}{f_i^k}; \quad S^k = \log F^k - N^k.$$

Если $f_i^k = 1$ для всех i, то $N^k = \log m$, а $S^k = 0$. Если же термин встречается только в одном документе и $f_i^k = F^k$, то $N^k = 0$, $S^k = \log F^k$.

Весовая функция, основанная на понятиях сигнала и шума, могут иметь вид $w_{ik} = S^k/N^k$ или $(S^k/N^k)S^k$.

Система определения веса по отношению к понятиям «сигнал» и «шум» является неоптимальной, поэтому взвешивание терминов может быть основано также на вариации распределения термина k в документах. Вариация будет небольшой для терминов, встречающихся равномерно или в малом числе документов. Если же термин распределяется неравномерно по достаточно большому набору документов, вариация становится высокой, и это свойство можно использовать при взвешивании.

Разрешающая способность термина. Эта модель изучает сходство документов, обозначенных каким-либо термином, в пространстве документов. При этом вычисляется сходство между документом, представленным этим термином, и всеми другими документами или документами-центроидами. Если термин редко встречается в документах, его удаление увеличит их сходство. Соответствующие изменения сходства могут быть описаны различающей способностью DV_k для каждого термина k. Тогда можно определить весовую функцию

$$w_{ik} = \left(\frac{f_i^k}{F^k} \right) DV_k.$$

Для того чтобы вычислить DV_k, возьмем центроид документов и определим функцию плотности в пространстве документов SD как сумму коэффициентов сходства всех документов с центроидом G:

$$SD = \sum_{i=1}^{m} S(G, D_i).$$

Если $0 \le S \le 1$, то $0 \le SD \le m$. Обозначим через SD_k плотность пространства документов после удаления термина k из всех векторов документов. Тогда различающая способность DV_k равна $SD_k - SD$. Для хороших различителей DV_k положительна и удаление хороших различителей делает пространство документов более плотным.

В двоичных схемах учитывается только наличие или отсутствие термина k в документе i ($f_i^k = 1$ или 0 соответственно).

Подсистема поиска документов по заданной тематике. Данная подсистема осуществляет поиск документов по заданной тематике, удовлетворяющих выбранному критерию:

- ключевой поиск (булева комбинация ключевых терминов; поиск по образцу);
- поиск по подобию и сходству (документы, наиболее близкие тематике данного документа и набору ключевых слов, соответственно).

Реализованный в ИС способ поиска предусматривает формирование на устройстве хранения данных архива документов и их фрагментов, в котором все сохраняемые фрагменты документов проиндексированы. Фрагмент документа представляет собой любую выбранную последовательность исходных слов, входящих в документ, и состоит, по меньшей мере, из одного исходного слова. В качестве фрагмента документа используют, например, предложение, абзац, параграф или раздел документа. Индексирование заносимых в архив документов осуществляют таким образом: при занесении в архив документа определяют фрагменты, на которые он будет разбит, и осуществляют его индексирование для дальнейшего индексного поиска фрагментов документов.

Определение методики разбиения документа на фрагменты не зависит от количества слов, входящих в заданную последовательность слов, которые составляют фрагмент.

При индексном поиске фрагментов указывают методику их поиска. Поиск фрагментов документов можно осуществлять с использованием одной и более методик. Программа предусматривает установку параметров использования методики по умолчанию.

Методику разбиения документов на фрагменты определяют пользователи.

Программа предусматривает настройку режима, при котором осуществляется самообучение системы для дальнейшего автоматического разбиения документов на фрагменты в соответствии с определенными правилами. Такой режим предусмотрен для работы с теми документами, формат которых представляет собой упорядоченную последовательность блоков информации. В качестве примера можно

привести текст патентной заявки, включающий в себя обязательные разделы: «Описание изобретения», «Формула изобретения», «Реферат» и т.д. Несколько раз обратившись к документу такого типа и разделив его на фрагменты с использованием конкретной методики, пользователь неявно формирует правила для дальнейшего автоматического разбиения на фрагменты всех текстов патентных заявок. Система «запоминает» правила разбиения документа.

Программа позволяет осуществлять поиск фрагментов, похожих по текстовому и/или смысловому содержимому на выбранный, одновременно для одного и нескольких фрагментов, а также предусматривает формирование тематик из одного и более фрагментов. При этом множество фрагментов для формирования конкретной тематики определяет пользователь.

В качестве функций, посредством которых формируется множество уникальных слов для поиска, используют следующие:

- функцию формирования списка уникальных слов для выбранного фрагмента;
- функцию подсчета количества вхождений уникальных слов в текстовое содержимое фрагментов выбранной тематики;
- функцию определения частоты вхождения уникальных слов в текстовое содержимое фрагментов выбранной тематики.

В ряде случаев для формирования множества уникальных слов для поиска гораздо удобнее воспользоваться такими функциями:

- функцией формирования списка уникальных слов для выбранного фрагмента. Данная функция предназначена для формирования множества уникальных слов, присущих выбранному фрагменту;
- функцией подсчета количества вхождений уникальных слов в выбранную тематику. Для уникального слова сформирована группа соответствия, полученная в процессе предварительной обработки. Для каждого элемента группы соответствия, связанной с уникальным словом, определяют количество его вхождений в текстовое содержимое фрагментов выбранной тематики. Затем суммируют показатели количества вхождений всех элементов групп соответствия, составляющих конкретную группу;
- функцией определения частоты вхождений уникальных слов в выбранную тематику. Для уникального слова определяют частоту вхождения в текстовое содержимое фрагментов выбранной тематики. Частота вхождения исчисляется в процентах от количества вхождений уникального слова, наиболее часто используемого в текстовом содержимом фрагментов выбранной тематики.

Использование функций позволяет пользователю создавать различные правила (логики) для определения множества уникальных слов, используемых в качестве параметров поиска.

В ИС использованы три логики похожих документов.

Первая логика позволяет осуществлять выбор установленного количества слов из полученного списка уникальных слов, упорядочен-

ного в соответствии с общим количеством вхождений в выбранную тематику всех слов, полученных в процессе предварительной обработки уникального слова. Данная логика использует функцию формирования списка уникальных слов для выбранного фрагмента и функцию подсчета вхождений уникальных слов в выбранную тематику.

Для выбранного фрагмента определяют список уникальных слов и для каждого из них подсчитывают количество вхождений в выбранную тематику. Далее список сортируют в порядке возрастания количества вхождений уникальных слов в выбранную тематику (на первом месте в полученном списке будет располагаться уникальное слово с наименьшим числом вхождений). После этого из упорядоченного списка уникальных слов по правилам пользователем формируется множество слов для поиска. Такими правилами является указание интервала для выбора слов. Интервал для выбора слов может быть любым.

Основная задача данной логики — выбор для поиска установленного количества наиболее редких или наиболее часто используемых в выбранной тематике уникальных слов.

Вторая логика позволяет из множества уникальных слов, полученных для выбранного фрагмента, выбрать слова, входящие в выбранную тематику с заданной частотой. Логика предполагает использование функции формирования списка уникальных слов для выбранного фрагмента и функции определения частоты вхождений уникальных слов в выбранную тематику.

Для выбранного фрагмента формируют множество уникальных слов и для каждого из них определяют частоту его вхождения в выбранную тематику. После этого из всего множества уникальных слов, полученных для выбранного фрагмента, выбирают уникальные слова по определенным пользователем правилам. В качестве правил указывают интервал частоты вхождения, в соответствии с которым формируют список уникальных слов для поиска, или количество слов с наименьшим или наибольшим показателем частоты вхождения.

Использование уникальных слов с заданным диапазоном частоты вхождения позволяет, в отличие от первой логики, более гибко формировать поисковый запрос.

Указание количества слов (первая логика), даже редко входящих в выбранную тематику, не всегда позволяет сформировать оптимальный запрос. Использование второй логики позволяет устранить эту проблему.

Третья логика дает возможность формировать для поиска множество уникальных слов с наибольшим количеством вхождений в выбранную тематику, при этом имеющих наименьшее количество вхождений в весь хранимый архив. Логика используют функцию формирования списка уникальных слов для выбранного фрагмента и

функцию определения частоты вхождений уникальных слов в выбранную тематику.

Для выбранного фрагмента формируют список уникальных слов. Для каждого из полученных уникальных слов определяют частоту его вхождения в текстовое содержимое фрагментов выбранной тематики. Далее последовательно обращаются к каждому уникальному слову из полученного списка и определяют частоту его вхождения в весь архив. Таким образом, для каждого уникального слова определяют два показателя: частоту вхождения в выбранную тематику (Ч1) и частоту вхождения в весь архив, за исключением выбранной тематики (Ч2). Затем для каждого уникального слова определяют разницу показателей (Р = Ч1 – Ч2), после чего список уникальных слов сортируют в порядке убывания показателя Р. Из сформированного списка для запроса выбирают установленное количество уникальных слов с наибольшим показателем Р.

Преимущество данной логики заключается в возможности осуществлять поиск по ключевым для интересующей пользователя тематики словам. Данная логика позволяет исключить из запроса общеупотребимые слова и сформировать запрос из множества уникальных слов, наиболее точно характеризующих текстовое содержимое выбранной тематики. Для поиска могут быть выбраны уникальные слова, показатель Р которых находится в заданном диапазоне.

Пользователь может использовать другие комбинации функций для создания новых логик.

В информационно-поисковой системе нахождение релевантных документов осуществляется в соответствии с терминами, заданными в запросе пользователя. Поиск ведут, применяя два основных подхода: точное совпадение терминов и с использованием меры сходства (несходства). Поиск на точное совпадение терминов, в свою очередь, можно разделить на поиск с инвертированием (индексированием) текста и на поиск по образцу в полном тексте.

При *поиске по совпадению терминов* задается полное и/или частичное совпадение терминов (ключевых слов) для нахождения идентификаторов документов, содержащих эти ключевые слова.

Для представления несущественных символов в запросе используется вопросительный знак и в позиции, где стоит «?», будет принят любой символ. На самом деле в этой позиции вообще не требуется производить сравнение. Несущественные символы могут начинать слово, заканчивать его или находиться в середине. В каждом случае их может быть фиксированное или переменное число, что соответствует поиску по заданному или произвольному количеству несущественных символов.

Следовательно, для выполнения запросов на поиск в строке нужны следующие структуры поиска: 1) на точное совпадение; 2) на частичное совпадение с использованием различных вхождений несущественных символов.

Поиск с использованием логических выражений заключается в составлении поисковых запросов с помощью поисковых терминов и логических связок AND (И), OR (ИЛИ), XOR (ИСКЛЮЧАЮЩЕЕ ИЛИ) и NOT (НЕ) между отдельными поисковыми подвыражениями (рекурсивное определение), каждое из которых, в свою очередь, может быть поисковым термином.

При контекстном поиске указываются содержание (абзац, предложение) и расстояние между поисковыми терминами, которым должен удовлетворять документ. Документ, таким образом, должен удовлетворять некоторым требованиям к его естественно-языковой структуре.

Для поиска по образцу в полном тексте необходимо иметь образец, строку текста, в которой надо выполнить поиск, и указатель на текущую позицию в строке. Перед поиском образец размещается в начале текста, но сравнение начинается с конца образца. Последний символ образца сравнивается с соответствующим символом строки.

Если они совпадают, сравнение продолжается справа налево. Если же они различны, образец сдвигается вправо и поиск продолжается с пропуском некоторых символов текста.

Чтобы достигнуть конца текста как можно скорее, было предложено использовать два варианта сдвигов Δ_1 и Δ_2, которые выбирают по следующим критериям:

- когда обнаруживается несовпадение, пытаются найти сравниваемый символ текстовой строки в остальной (расположенной левее) части образца. Если эта попытка успешна, образец сдвигается вправо на Δ_1 для того, чтобы выровнять найденные совпадающие символы в образце и в тексте. Если же такого символа в левой части образца нет, образец сдвигается вправо на Δ_1 для того, чтобы первый его символ находился сразу за рассмотренным символом текста. Наибольшее значение Δ_1 равно длине образца;
- если часть образца совпадает с текстом, делается попытка найти вхождения этого фрагмента образца в левой его части. Как только такое вхождение найдено, образец сдвигается вправо на Δ_2, чтобы выровнять совпадающие части.

Поиск по сходству реализует выборку релевантных документов, а релевантность — понятие относительное. Все методы поиска в ИПС можно отнести к поиску по сходству.

Сходство можно интерпретировать как расстояние между двумя объектами: чем больше похожи эти объекты, тем они ближе. В другой интерпретации сходство понимается как угол между двумя векторами, представляющими объекты в векторном пространстве. Можно сравнивать объекты с объектами, документы с документами, запросы с документами. Объекты лучше рассматривать как векторные, а не точечные величины, поскольку сравниваются не сами объекты,

а их представления, а сходство можно интерпретировать как расстояние между двумя объектами.

Например, в ИПС принято представлять документ в виде вектора терминов (ключевых слов) в индексе. Соответственно вектор D_i документа i будет иметь вид $D_i = (t_{i1}, t_{i2}, ..., t_{in})$ для n ключевых слов. Каждый из терминов t_{ij} для базы из m документов ($1 \le i \le m$), ($1 \le j \le n$) может иметь следующий вид:

а) 1 или 0, обозначающие наличие или отсутствие термина j в документе i, если используется двоичное представление;

б) константа, представляющая вес термина j в документе i, если используется какая-либо схема со взвешиванием.

Можно представить базу данных в виде матрицы размером $m \times n$:

	ТЕРМИНЫ				
ДОКУМЕНТЫ	t_{11}	t_{12}	t_{13}	...	t_{1n}
	t_{21}	t_{22}	t_{23}	...	t_{2n}
	...	...	...	...	...
	t_{m1}	t_{m2}	t_{m3}	...	t_{mn}

Как видно из матрицы, каждый документ может быть представлен как вектор (строка матрицы) и каждому термину соответствует вектор-столбец матрицы, показывающий вхождения этого термина в документы. Аналогично каждый запрос может быть представлен вектором $Q_i = (t_{i1}, t_{i2}, ..., t_{in})$, показывающим наличие или отсутствие (или вес) ключевых слов базы данных в запросе i.

После того как представление выбрано, можно вычислить меру сходства между ключевыми словами в матрице. Это позволяет классифицировать (группировать) термины и строить словари синонимов для терминов (тезаурусы). Похожим образом можно вычислить сходство между строками матрицы документов, классифицируя документы для большей эффективности выборки. Наконец, при поиске можно вычислить сходство между строкой, представляющей запрос, и каждой из строк матрицы документов и выбрать те из документов, для которых значение меры сходства превышает некоторый порог T, заданный пользователем. В базах данных, содержащих большое количество документов, классификация документов становится абсолютно необходимой. В этом случае вместо вычисления сходства между вектором запроса и каждым из многочисленных векторов документов вычисляют сходство запроса с представителями кластеров (центроидов).

Вектор центроида G представляется в виде

$$G_i = g_{i1}, g_{i2}, \ldots, g_{in}, \; g_{ij} = \frac{1}{m}\sum_{i=1}^{m} t_{ij},$$

где G_i — центроид i-го кластера документов для n терминов ($1 \le j \le n$); g_{ij} — средний вес для m документов.

Наряду с понятием «сходство» используется понятие «несходство». Иногда его удобно применять в математических исследованиях. Любая функция несходства может быть преобразована в функцию сходства с помощью уравнения: сходство = (1 + несходство)$^{-1}$. Обратная зависимость не всегда справедлива.

Существуют различные меры сходства, или функции совпадения. Если необходимо измерить сходство документа D_i и запроса Q_i, можно использовать, например, следующие известные функции сходства:

Коэффициент Дайса: $S(D_i,Q_j) = \dfrac{2(\sum_{k=1}^{n} t_{ik} \cdot t_{jk})}{\sum_{k=1}^{n} t_{ik} + \sum_{k=1}^{n} t_{jk}}$.

Коэффициент Жаккара: $S(D_i,Q_j) = \dfrac{\sum_{k=1}^{n} t_{ik} \cdot t_{jk}}{\sum_{k=1}^{n} t_{ik} + \sum_{k=1}^{n} t_{jk} - \sum_{k=1}^{n} t_{ik} \cdot t_{jk}}$.

Коэффициент косинуса: $S(D_i,Q_j) = \dfrac{\sum_{k=1}^{n} t_{ik} \cdot t_{jk}}{\sqrt{\sum_{k=1}^{n} t_{ik}^2 \cdot \sum_{k=1}^{n} t_{jk}^2}}$.

Коэффициент перекрытия: $S(D_i,Q_j) = \dfrac{\sum_{k=1}^{n} t_{ik} \cdot t_{jk}}{\min(\sum_{k=1}^{n} t_{ik}, \sum_{k=1}^{n} t_{jk})}$.

Например, вычислим коэффициенты сходства следующих векторов документа и запроса.

D_i = (0, 3, 4, 0, 1, 3, 2).

Q_i = (1, 1, 2, 1, 0, 0, 4) — веса в векторе запроса, заданные пользователем.

Коэффициент Дайса: $\dfrac{38}{22} = 1{,}73$.

Коэффициент Жаккара: $\dfrac{19}{3} = 6{,}33$.

Коэффициент косинуса: $\frac{19}{\sqrt{897}} = 0{,}63.$

Коэффициент перекрытия: $\frac{19}{9} = 2{,}11.$

В этих уравнениях эксперт определяет меру сходства благодаря наличию произведений терминов, а знаменатели задают различные способы нормировки.

Подсистема аналитической обработки накопленной информации. Подсистема обеспечивает:

- анализ тематико-временны́х зависимостей (изменение интенсивности работ по тематике во времени);
- выявление факта изменения терминологии предметной области и экспертное определение новых направлений тематики работ;
- выявление развивающихся направлений науки и техники лицом, принимающим решение, на основе многомерного анализа данных;
- автоматизированную выдачу реферативных справок по интересующей тематике работ, организации, разработчику в рамках информации, содержащейся в каталоге.

Подсистема OLAP-анализа выполнена на основе web-компонентов Office. Она обеспечивает просмотр и анализ данных, представленных в виде микрокубов, и выпуск отчетов. В основу технологии многомерного анализа данных положена многомерная модель представления данных.

Микрокуб хранит данные в сжатом виде, что позволяет применять его для передачи больших объемов данных в Интернет, по электронной почте. Микрокуб может содержать данные, выгруженные из произвольных реляционных БД, Excel-файлов, текстовых файлов, например протоколов сайтов или других источников.

В окне микрокуба отображаются такие элементы:

- динамическая таблица — OLAP-таблица с данными микрокуба. Таблиц в микрокубе может быть несколько. Переход от одной таблицы к другой осуществляется с помощью закладок с названиями таблиц;
- диаграмма — синхронное с таблицей графическое изображение. Диаграмм может быть несколько.

Переход от одной диаграммы к другой осуществляется с помощью закладок с названиями диаграмм.

Расположение названий закладок с таблицами и диаграммами можно изменить. Для манипулирования представлением данных в окне существуют панели инструментов. С помощью кнопок этой панели инструментов можно изменять представление данных в таблице (детализировать и обобщать числовые данные, удалять нули, менять строки и колонки местами и пр.).

7.6. Компьютерная реализация ИАД в научных исследованиях

Практическая реализация информационной системы накопления и обработки информации о достижениях науки и техники подразумевает использование технологии разработки ее основных компонентов. К таким системообразующим компонентам, в первую очередь, относится СППР специалиста по поиску и обработке текста и хранилище данных.

Одним из первых производителей таких систем стала компания Arbor Software, выпустившая продукт Essbase. Компания Oracle предлагает систему Oracle Express, интегрированную с универсальным Oracle Server. Известны и другие производители MOLAP-систем, например SAS Institute. Однако, в отличие от Essbase, их продукты часто интегрированы в приложения, созданные для конкретных вертикальных или горизонтальных рынков, и поставляются лишь в составе этих приложений.

Для систем ROLAP гиперкуб — это лишь пользовательский интерфейс, который эмулируется на обычной реляционной СУБД. В этой структуре можно хранить очень большие объемы данных, однако для нее характерны низкая и неодинаковая эффективность OLAP-операций.

Примерами промышленных ROLAP-систем служат MetaCube фирмы Informix и Discoverer 3.0 фирмы Oracle. На практике иногда реализуется комбинация этих подходов.

Пример технологии проектирования OLAP. Несколько фирм предлагают системы построения витрин данных: Informatica (PowerMart Suite), Sagent Technology (Data Mart Solution) и Oracle (DataMart Suite). Для иллюстрации процесса разработки витрины данных можно использовать в сокращенном виде состав и функциональность пакета DataMart Suite.

Пакет включает в себя пять основных компонентов: Data Mart Designer, Data Mart Builder, Oracle 7 Enterprise Server, Web Server и Discoverer 3.0.

Data Mart Designer позволяет описывать структуру витрины данных и запоминать ее в репозитарии. На выходе Data Mart Designer порождает описание на языке DDL SQL, которое затем подается на вход Oracle 7 Enterprise Server. В результате создается структура базы данных, реализующая витрину данных. В ходе ее построения пользователь может применять существующие описания структур или строить витрину «с нуля». Кроме того, Data Mart Designer позволяет строить приложения для Oracle Web Server на базе PL/SQL.

Data Mart Builder извлекает данные из внешних источников и заполняет витрину. Он обладает наглядным специализированным интерфейсом, отображающим потоки данных при заполнении хранилища данных. Data Mart Builder способен извлекать данные из

реляционных СУБД и CSV-файлов. Web Server предоставляет открытую платформу для разработки Web-приложений. Он включает Web Request Broker (WRB), реализованный на основе технологии картриджей и позволяющий разрабатывать Web-приложения, встраиваемые в Web Server. В качестве средств разработки могут использоваться языки Java, PL/SQL, LiveHTML, C и C++. Discoverer 3.0 — средство конечного пользователя, позволяющее генерировать отчеты и выполнять некоторые OLAP-операции с витриной данных. Отчеты, построенные с помощью Discoverer 3.0, можно экспортировать в формате HTML, делая их доступными для Web-браузеров. Discoverer 3.0 позволяет создавать и поддерживать таблицы агрегированных данных. DataMart Suite включает готовое приложение Sales Analyzer.

Предположим теперь, что в общем случае имеется корпоративное хранилище данных и ряд витрин данных. Каким образом следует организовать доступ к информации для анализа? Требуется обеспечить возможность анализа данных (OLAP) как из витрин, так и непосредственно из хранилища. Разница в анализе определяется не столько размером базы (витрина может лишь ненамного уступать хранилищу), сколько тем, что витрины, как правило, не содержат детальных (неагрегированных) данных. Это означает, что анализ данных витрины не требует глубокой детализации и часто может быть выполнен более простыми средствами.

Наряду с мощными серверами многомерных баз данных и ROLAP-серверами на рынке предлагаются клиентские OLAP-серверы, предназначенные, главным образом, для работы с небольшими объемами данных и ориентированные на индивидуального пользователя. Подобные системы были названы настольными, или DOLAP-серверами (Desktop OLAP). В этом направлении работают фирмы Business Objects (Business Objects 5.0), Andyne (CubeCreator, PaBLO), Cognos, Brio Technology.

Лидером пока считается компания Cognos, поставляющая продукты PowerPlay, Impromptu и Scenario. PowerPlay — это настольный OLAP-сервер; для извлечения данных из реляционных баз данных (Paradox, dBase, Clipper), «плоских» файлов и электронных таблиц (Microsoft Excel). Используется генератор запросов и отчетов Impromptu. Затем специальный компонент, называемый Transformer, помещает извлеченные данные в клиентскую многомерную базу, которая называется PowerCube. Потребителям предоставляются широкие возможности по управлению PowerCube: передавать ее от пользователя к пользователю по запросу и принудительно, помещать на сервер для разделения доступа к ней или пересылать по электронной почте. Cognos постаралась сделать свой продукт максимально открытым:

PowerCube может быть помещен в реляционные базы Oracle, Informix, Sybase, MS SQL Server на платформах UNIX, HP/UX, Sun Solaris, IBM AIX;

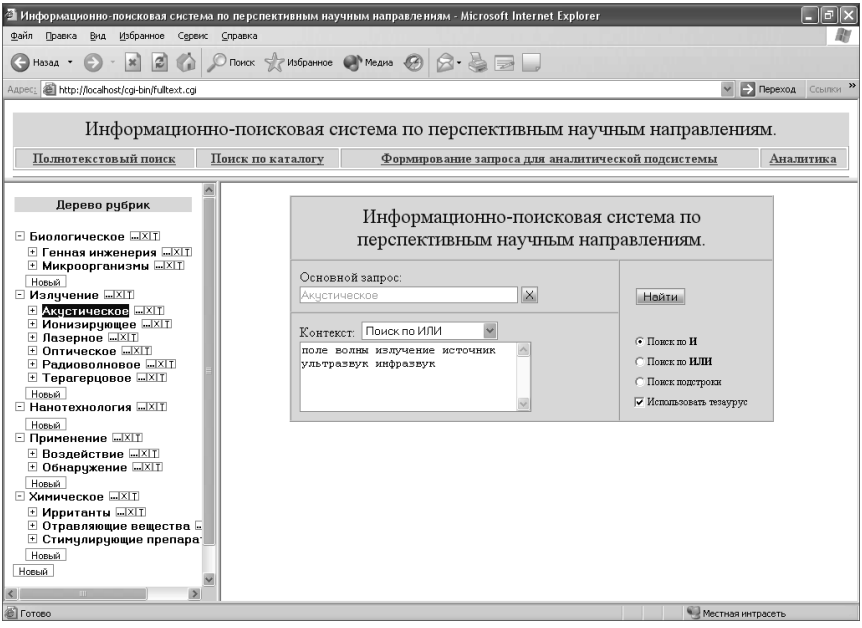


Рис. 7.8. Представление дерева рубрик (слева) и формирование запроса (справа)

PowerPlay способен анализировать содержимое не только PowerCube, но и других многомерных баз данных.

Стоит отметить, что все эти фирмы объединяет стремление включить в свои продукты компоненты, предназначенные для интеллектуального анализа данных. Например, усилия Business Objects и Cognos направлены на подготовку окончательных версий компонентов Business Miner и Scenario, соответственно, предназначенных именно для ИАД.

Использование в качестве основы иерархических рубрикаторов тематических структур. Тематическая структура описывает содержание анализируемых текстов в виде иерархии связанных тем и подтем, раскрывающих содержание тем. Все темы и подтемы выражаются понятиями исходных текстов и соответствуют элементам семантической сети. Однако связи между понятиями односторонни и направлены от главного понятия к подчиненным.

В результате представление тематической структуры текста оказывается иерархическим — от каждой темы раскрываются связи только к ее подтемам, от них — к подтемам следующего уровня и так далее, вплоть до самых незначительных тем, уже не имеющих нисходящих связей. Тематическая структура, таким образом, имеет вид дерева, в корне которого находятся главные темы, в ветвях — подтемы.

В остальном с точки зрения интерфейса работа с тематической структурой, описываемая ниже, полностью аналогична работе с семантической сетью.

Тематическая структура представляется в окне в виде дерева рубрик (рис. 7.8).

Понятия представляют названия тем. Некоторые темы имеют раскрывающиеся ветви связей с подтемами. Понятия в корне дерева представляют список главных тем анализируемых текстов, а связанные с ними элементы в ветвях последующих уровней дерева — списки подтем, в которых разворачиваются главные темы.

Щелчок левой клавишей мыши по значку <+> возле выбранного понятия раскрывает список всех понятий, связанных с ним. Щелчком по значку <–> возле понятия с раскрытым списком можно его закрыть.

Можно настраивать вид тематической структуры на экране, изменяя количество отображаемых понятий и связей, способ их сортировки, а также количество кустов в дереве. Получаемые результаты могут использоваться в готовом виде (например, в качестве электронного глоссария) или после предварительной корректировки экспертом служить основой для построения более «строгих» классификаторов.

Рубрикатор, сформированный на базе эталонных текстов, можно использовать для автоматической классификации и маршрутизации новых документов.

Детальный контроль доступа к данным базируется на механизме динамической модификации запросов, который позволяет реализовать правила защиты, связанные с конкретными объектами.

SQL-запросы пользователей к таблицам базы данных модифицируются с помощью соответствующих правил защиты, накладываемых посредством динамически вычисляемой декларации WHERE. Такая декларация вырабатывается специальной функцией, реализующей правила защиты; это может быть любой предикат, выражение или формула, возвращаемая функцией. Функции, которые возвращают предикаты, могут также вызывать другие функции, которые могут либо получать доступ к информации операционной системы, либо возвращать условия WHERE, извлеченные из файла операционной системы или центрального хранилища правил. Благодаря такому решению достигается дополнительная гибкость, поскольку оно может возвращать различные предикаты для каждого пользователя, каждой группы пользователей или каждого приложения.

Решения о предоставлении доступа могут приниматься на основе информации о пользователе, например о его должности, подразделении и статусе. Механизм контекстов приложений (Application Context) позволяет определять, устанавливать и проверять атрибуты защиты, которые помогают осуществлять детальный контроль доступа, реализуя виртуальные частные базы данных.

Контексты приложений выступают как своего рода защищенные кэши данных, которые могут применяться для организации детального контроля доступа к конкретной таблице. При регистрации пользовательского сеанса в базе данных Oracle устанавливает некоторый контекст приложения. Информация в контексте приложения определяется разработчиком с учетом сведений, относящихся к конкретному приложению.

При реализации детального контроля доступа можно использоваться два следующих механизма:

1) выбор предикатов: разработчик может обратиться к контексту приложения внутри функции реализации правил защиты, для того чтобы определить корректный возвращаемый предикат;

2) связывание переменной в предикате: атрибут контекста может быть использован в самом предикате для предоставления связанной переменной. При этом создается один SQL-запрос, применяемый всеми пользователями, но для каждого из них он выполняется поразному.

Контекст приложения обеспечивает защиту благодаря своим свойствам:

- *уникальность контекста*. Oracle 9i обеспечивает уникальность имен контекстов во всей базе данных, гарантируя, что отдельные пользователи не смогут их дублировать ни случайно, ни преднамеренно. Определять контекст могут только пользователи с соответствующими полномочиями;
- *проверка атрибутов*. При переустановке атрибутов контекста гарантируется корректность значений этих атрибутов и наличие у пользователя права оперировать доступом с соответствующим набором атрибутов. С этой целью могут выполняться запросы к метаданным приложения;
- *установка атрибутов защиты*. База данных гарантирует, что атрибут контекста устанавливает заслуживающий доверия пользователь или процесс. В результате приложениям можно предоставить возможность принимать решения о защите на основе контекста приложения, поскольку контекст наверняка установлен корректно известным и заслуживающим доверия, а не злонамеренным пользователем или процессом.

Предопределенные атрибуты, называемые примитивами сеанса (session primitive), позволяют контексту приложения для осуществления контроля доступа использовать информацию, уже имеющуюся в базе данных относительно сеанса пользователя. СУБД Oracle поддерживает встроенное пространство имен для контекстов приложений USERENV, обеспечивающее доступ к примитивам сеанса, т.е. к информации, которую база данных получает о сеансе пользователя.

Обеспечением безопасности данных в продуктах Oracle заняты десятки различных механизмов, в том числе виртуальные частные

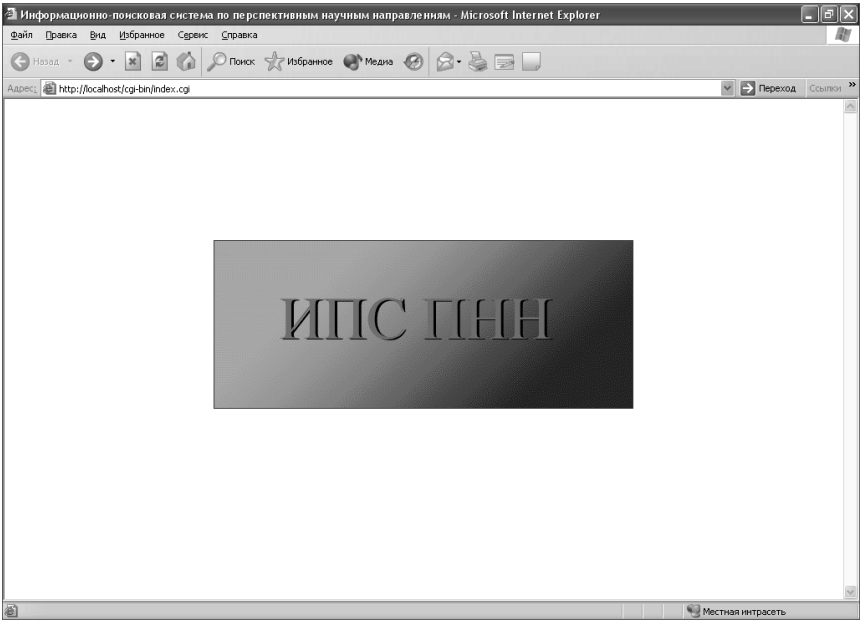


Рис. 7.9. Окно заставки системы

базы Virtual Private Database (VPD), Oracle 9i Label Security, селективная криптозащита и развитый аудит.

Организация интерфейса. Другим важным моментом при реализации систем ИАД является организация интерфейса для представления информации пользователям разного уровня.

Интерфейс конечного пользователя программной системы реализует Web-технологии и обеспечивает информационную поддержку пользователя на всех этапах технологии его работы. Он реализует сценарий технологии работы квалифицированного конечного пользователя, задаваемого рядом характерных этапов, которые выполняются в определенной последовательности и реализуются через систему вложенных окон:

- начальная настройка ИПС;
- мониторинг источников информации (ИИ);
- формирование массивов первичной информации;
- формирование массивов релевантной информации;
- аналитическая обработка данных (АОД);
- формирование отчета по предмету поиска.

Внешний вид начальной страницы, открывающейся в браузере, показан на рис. 7.9, главное меню (главная страница) с опциями — на рис. 7.10.

Полнотекстовый поиск — данная опция обеспечивает доступ к функциям системы, реализующим различные виды полнотекстового

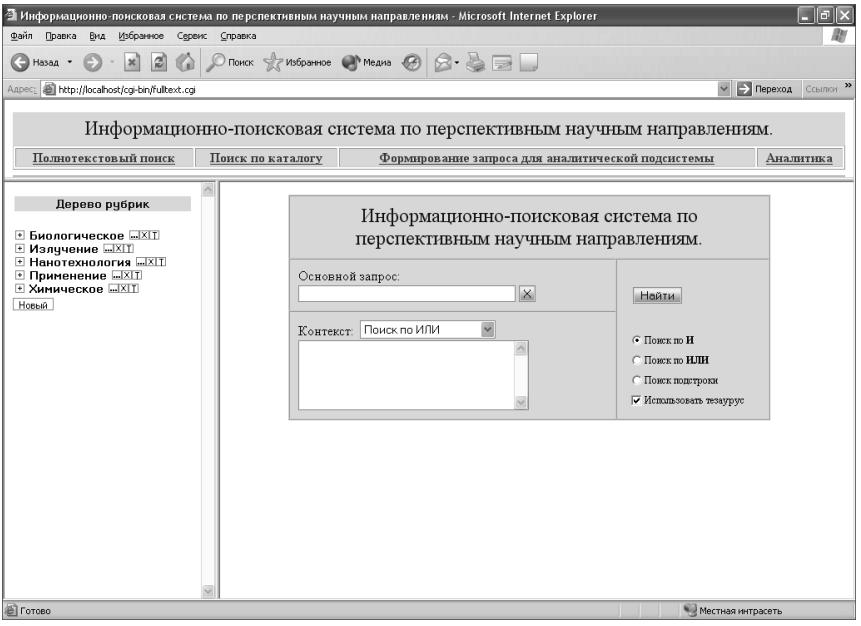


Рис. 7.10. Окно полнотекстового поиска

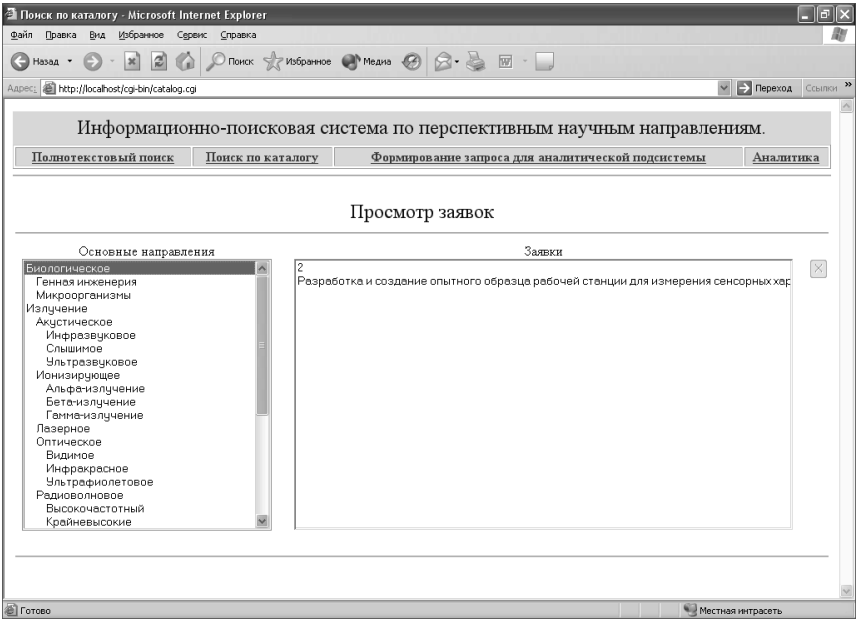


Рис. 7.11. Окно поиска по каталогу

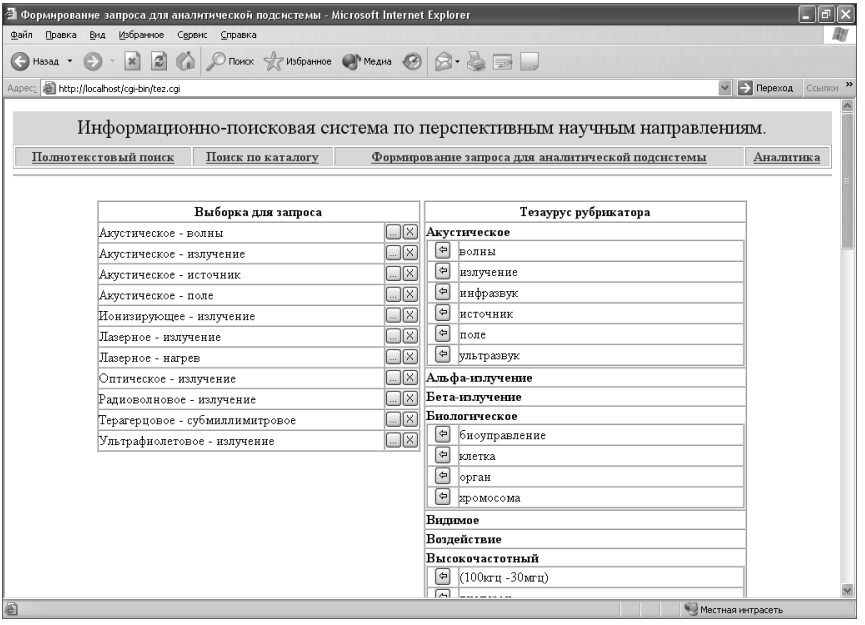


Рис. 7.12. Окно формирования запроса для аналитической подсистемы

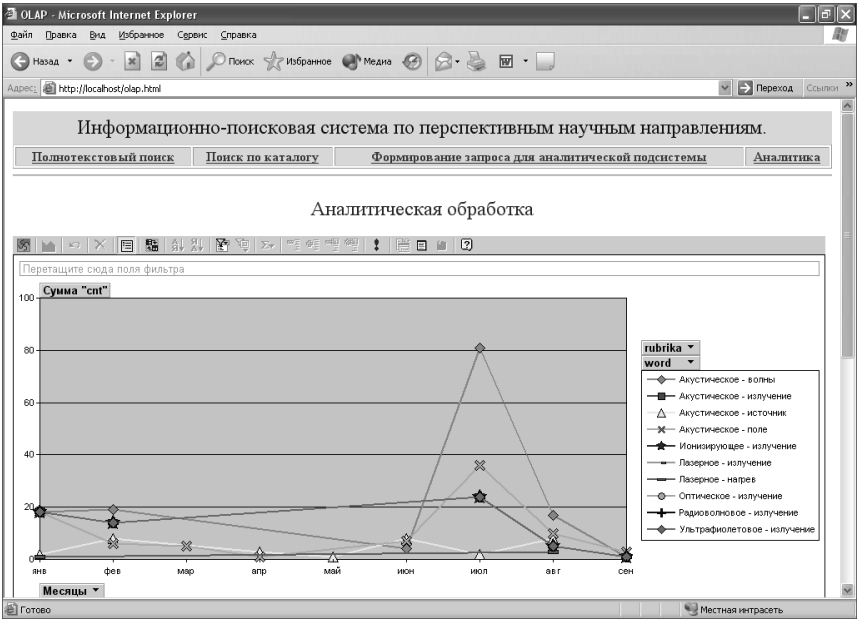


Рис. 7.13. Окно аналитической подсистемы (график)

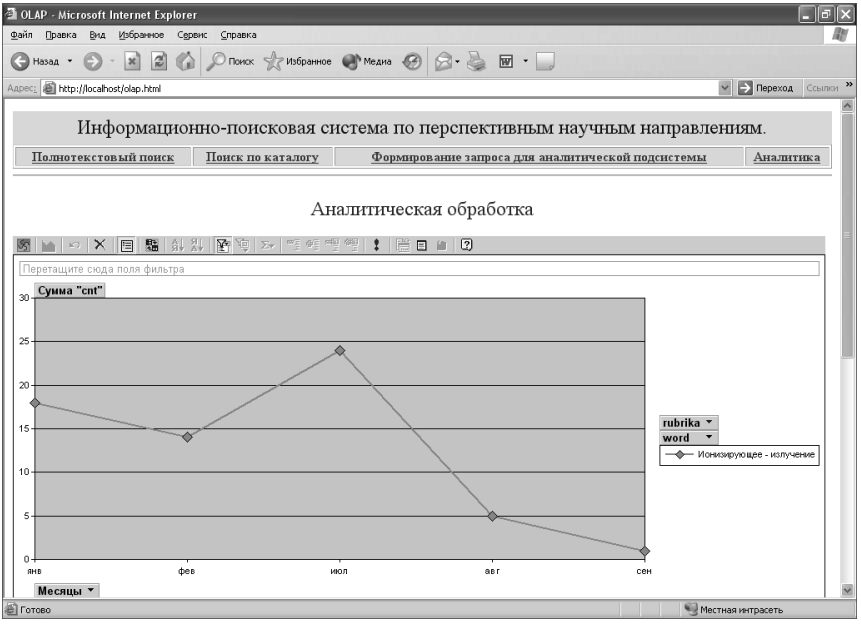


Рис. 7.14. Окно аналитической подсистемы (детализированный график)

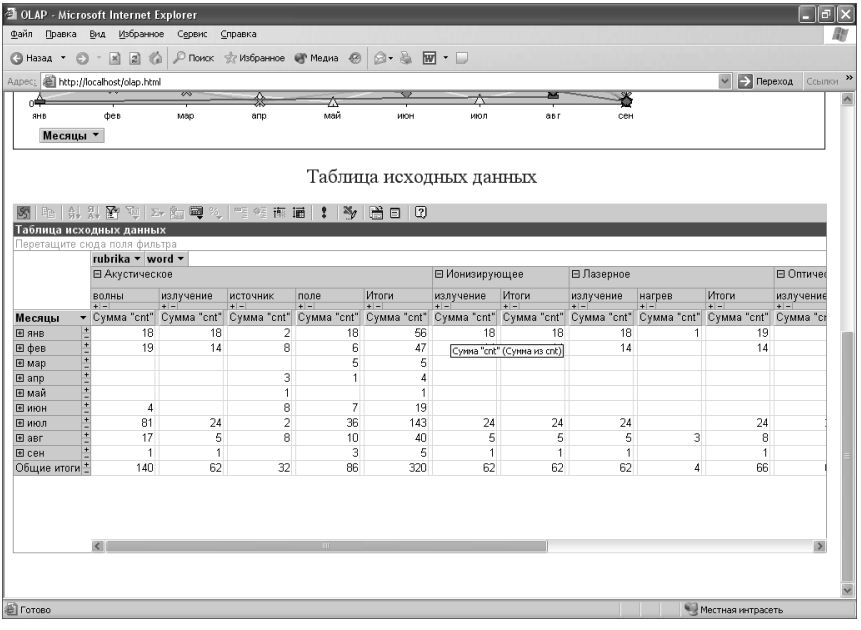


Рис. 7.15. Окно аналитической подсистемы (таблица)

поиска информации, создания и модифицирования рубрикатора, автоматического аннотирования выбранных документов, их классификации.

Поиск по каталогу — опция предоставляет возможность осуществлять поиск, просмотр и редактирование заявок в каталоге (рис. 7.11).

Формирование запроса для аналитической подсистемы — опция позволяет задать тематические рубрики и ключевые слова для проведения дальнейшего анализа изменения предметных областей, включая ретроспективный (рис. 7.12).

Аналитика — данная опция предоставляет справочную информацию, обеспечивает просмотр и анализ данных, опубликованных в виде микрокубов, построение графиков, гистограмм и выпуск отчетов (рис. 7.13 — 7.15). В этом режиме конечный пользователь может получать статистические данные разного уровня обобщения и форм представления, например статистические профили, касающиеся эффективности функционирования поискового механизма (анализ запросов, наиболее востребованных документов), аналитическую информацию общего вида, получаемую обработкой всего массива документов в целом.

Имеется возможность аналитического наращивания поискового механизма различными сервисными функциями (получение уведом-

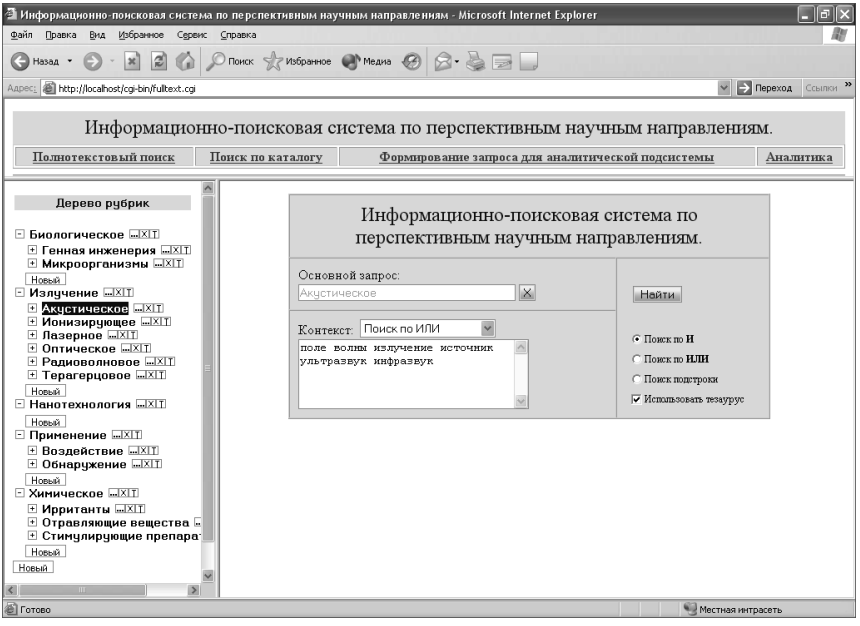


Рис. 7.16. Представление дерева рубрик (слева) и формирование запроса (справа)

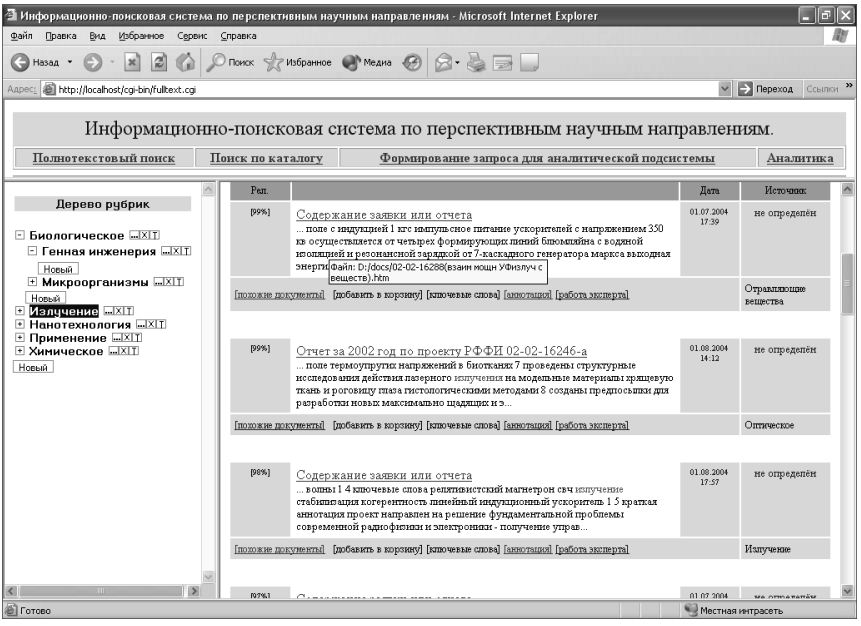


Рис. 7.17. Окно результатов поиска с автоматическим отнесением документов к рубрике

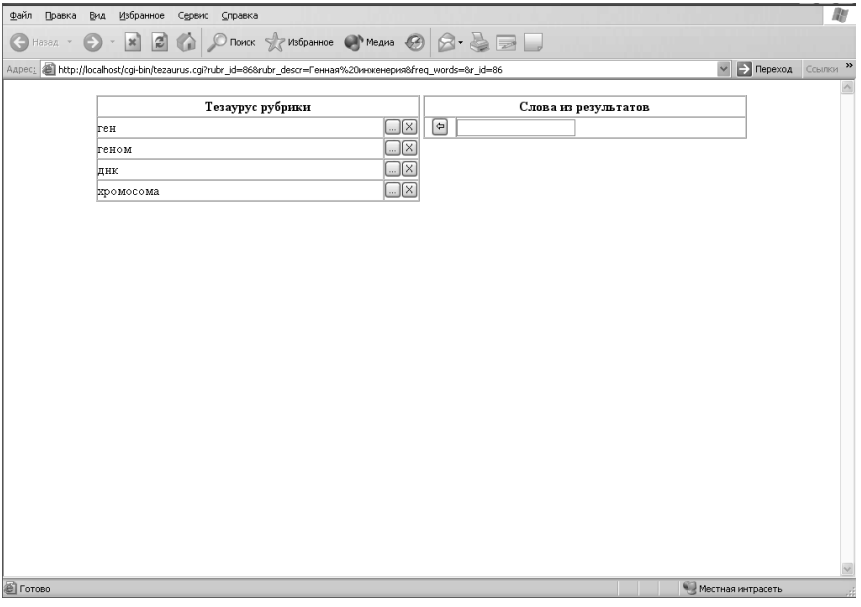


Рис. 7.18. Окно работы с Тезаурусом рубрики

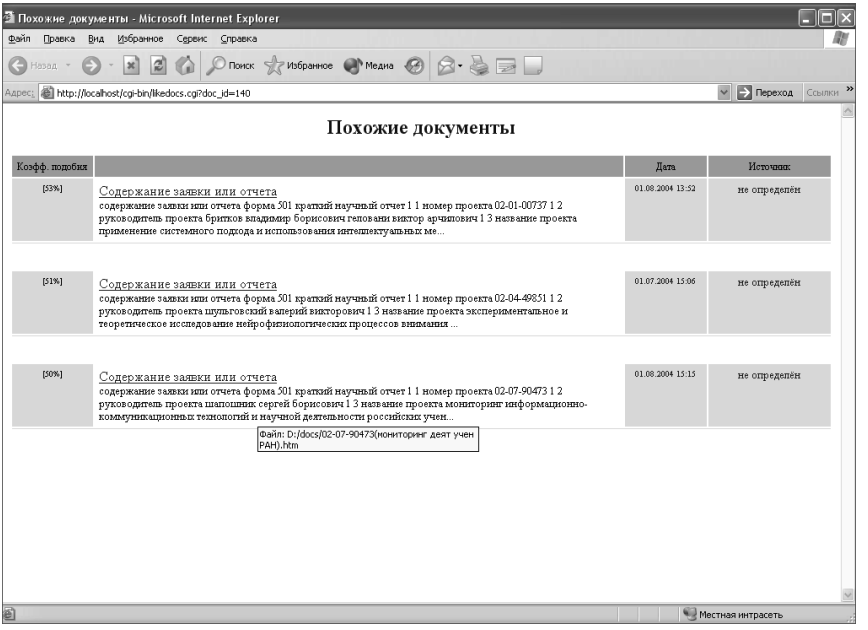


Рис. 7.19. Окно поиска похожих документов

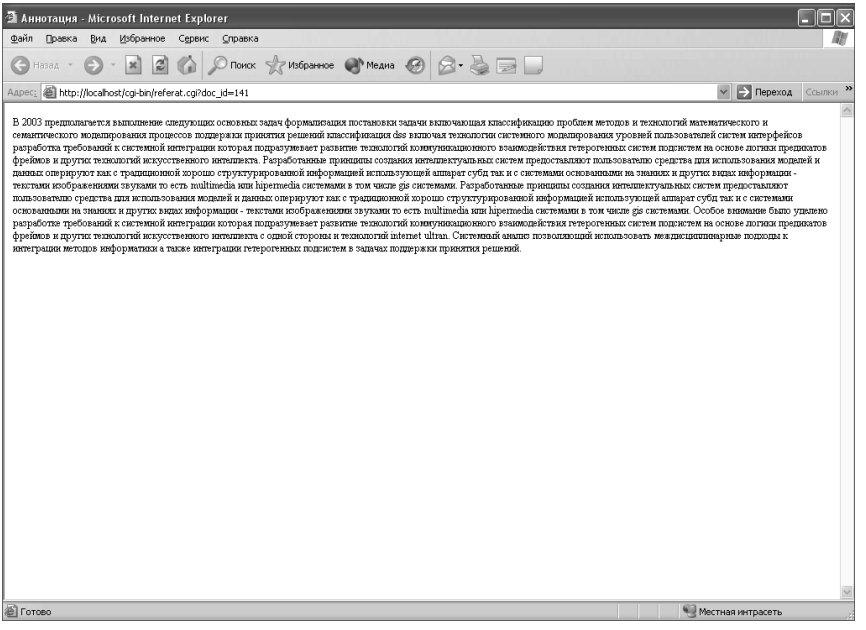


Рис. 7.20. Окно формирования аннотации документа

ления о появлении интересующего документа, автоматическое рубрицирование полнотекстовых массивов по формальным признакам, по тематике и др.).

Подсистема аналитической обработки накопленной информации, входящая в состав инструментальных средств макета, фактически решает задачи поиска ответов на вопросы типа «что мы имеем» (детальный и интегральный анализ текущего состояния, тенденции и закономерности), «что будет» (прогноз развития).

Каждая из представленных выше опций имеет собственную систему вложенных меню.

Страница Полнотекстовый поиск содержит такие пункты (рис. 7.16 — 7.20).

Дерево рубрик предоставляет возможности по формированию основных тем для классификации документов и автоматическому созданию запросов.

Тезаурус позволяет создавать и модифицировать описание предметной области.

Окно поиска дает возможность создавать различные запросы поисковой машины, осуществлять направленный и контекстный поиск и выводить результаты поиска.

Аннотация формирует краткое содержание найденного документа. Похожие документы дают возможность найти похожие докумен-

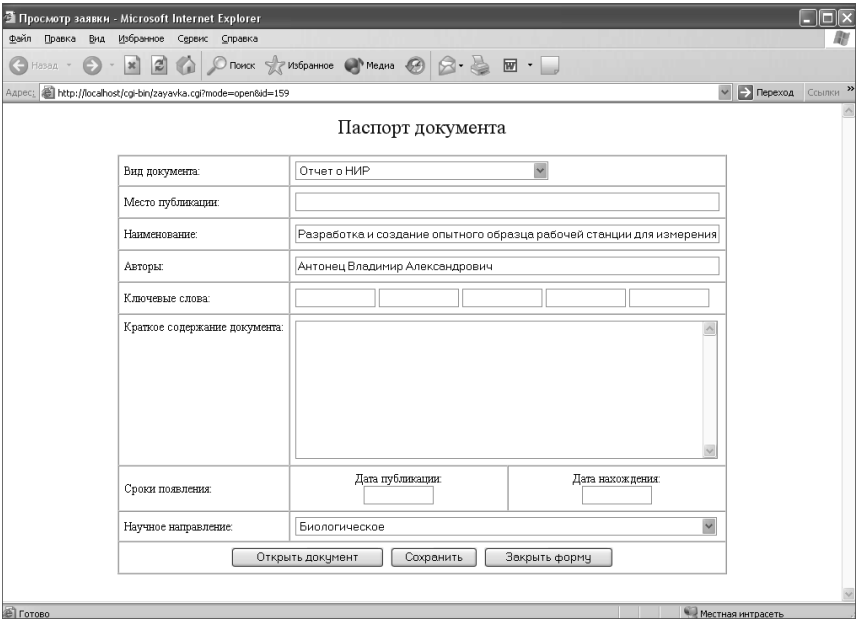


Рис. 7.21. Окно формирования паспорта документа

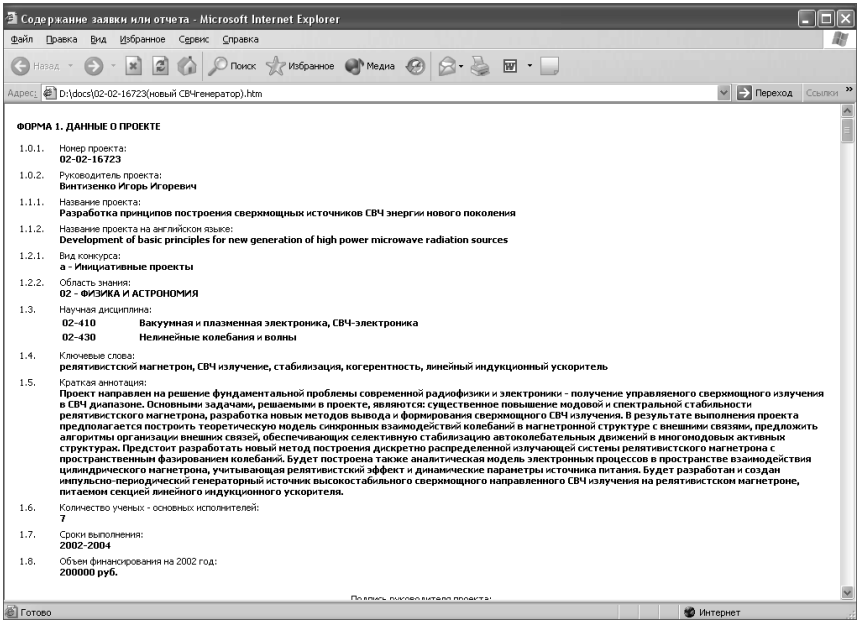


Рис. 7.22. Окно просмотра исходного документа

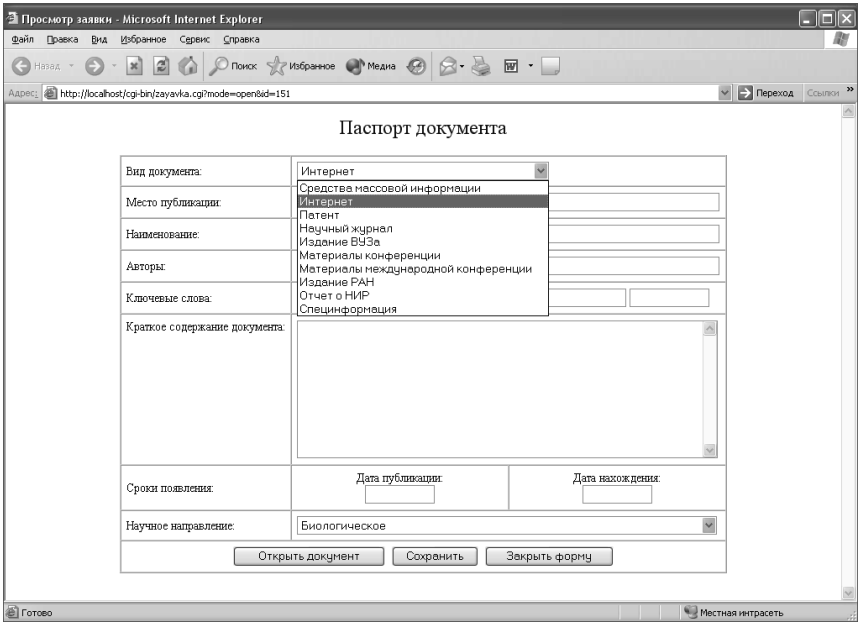


Рис. 7.23. Окно паспорта документа (место публикации)

ты. При формировании паспорта документа используется опция «Работа эксперта» (рис. 7.21).

Опция Поиск по каталогу позволяет определить набор полей, по которым возможен поиск в каталоге, и сохранить исходную версию документа (рис. 7.22, 7.23).

Таким образом, инструментальные средства ИС позволяют решать следующие задачи:

- осуществлять поиск научно-технической и патентной информации о новых достижениях науки и техники, организациях-разработчиках в существующих отечественных и зарубежных базах данных (РОСПАТЕНТ, РФФИ, сеть Интернет);
- на основе экспертных оценок и аналитической обработки определять перспективные направления науки и техники, способные оказать существенное влияние на появление новых и развитие имеющихся технологий;
- выдавать по новым технологиям перечень приоритетных направлений и научно-технических разработок, а также организаций, способных их реализовать;
- систематизировать и накапливать информацию в собственных базах данных и знаний.

Система работает следующим образом. Сначала вводятся исходные данные (тезаурусы, графовая модель представления текста произвольного содержания), конкретизирующие предметную область поиска. После ввода исходных данных пользователь работает через специализированный интерфейс, при этом выполняются такие процедуры:

1) эксперт (пользователь) задает либо документ-образец, либо ключевые слова с весами или слова-окрестности; выбирает кодовое расстояние, задающее окрестность вокруг ключевых слов; указывает источники информации. По умолчанию поиск осуществляется по всем источникам;

2) по выбранным источникам и ключевым словам формируется запрос к информационным источникам;

3) полученный от поисковых систем список документов проходит процедуру преобразования результатов к единой форме;

4) найденные документы могут иметь различный формат: doc, pdf, rar и т. д. Отфильтрованный список преобразуется к единому формату и сохраняется в базе данных промежуточных результатов.

Сохраненный список используется в дальнейшем при уточняющем поиске, где изменяется величина функции принадлежности и окружения;

5) для сохраненного списка документов строится графовая модель или шаблон документ;

6) вычисляются степени подобия документов по отношению к образцу (шаблону);

7) для всех документов (в зависимости от настроек сортировки) запускается процедура нахождения новых слов: в документах ищутся слова, являющиеся разностью между словами найденного документа и словами шаблона, которые затем сравниваются со словарем. Если в словаре такого слова нет, следовательно, оно новое;

8) релевантный список (относительно ИПС) подается эксперту в виде отсортированного списка по следующим параметрам: степени подобия по наличию и количеству новых слов. Эксперт может выбирать приоритетный параметр. Возможные варианты представлены в табл. 7.5.

Пользователь (эксперт) анализирует данный список документов. Релевантные по мнению эксперта документы он заносит в хранилище. Для этого эксперт указывает тематику, к которой они относятся. У эксперта есть возможность занести новое слово в словарь и создать новую рубрику.

Таблица 7.5. **Варианты параметров**

№ п/п	Параметр	Описание
1	Приоритет новых слов	Выводятся все документы, содержащие новые слова независимо от их функции принадлежности. Сортируются следующим образом: самым главным является документ с наибольшим количеством новых слов. В список также включаются документы, функция принадлежности которых ниже пороговой
2	Приоритет функции принадлежности	Сортировка по функции принадлежности: у кого она больше, тот и приоритетней. Новые слова не учитываются
3	Приоритет функции принадлежности и новых слов	Сортировка по функции принадлежности: у кого больше всех новых слов и функция принадлежности, тот и приоритетней

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Дайте определение понятия ИАД.
2. Что такое хранилище данных? Какие существуют виды структур хранилищ данных?
3. Каковы специфические требования к хранилищам данных?
4. Опишите методологию проектирования хранилищ данных.

5. Назовите разновидности многомерной модели данных.
6. Что такое многомерный куб? В чем достоинства и недостатки MOLAP?
7. Что такое киоск (магазин, витрина) данных?
8. Назовите основные задачи интеллектуального анализа данных.
9. Назовите основные этапы интеллектуального анализа данных.
10. Что такое «грязные» данные»?
11. Каким образом можно очистить данные?
12. Назовите основные этапы эволюции поисковых систем.
13. В чем отличие интеллектуальных агентов от Java-апплетов?
14. Какие типы интеллектуальных агентов используются в поисковых системах?
15. Как определяется частотность термина?
16. Как определяется частотность документа?
17. Что такое разрешающая способность термина?
18. Назовите логики похожих документов.
19. Как определяется сходство векторов документа и запроса?
20. Раскройте состав пакета DataMartSuite.

РАЗДЕЛ III

ПРИКЛАДНЫЕ ИНТЕЛЛЕКТУАЛЬНЫЕ ТЕХНОЛОГИИ

Глава 8

СИСТЕМЫ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ

8.1. Общие характеристики, области применения

Выработка решений — это неотъемлемая функция человека за все время его существования.

Под ***решением*** понимается выбор воздействия из множества возможных на основе поставленной цели и принятого критерия.

Цель — состояние, к которому стремится система.

Критерий — оценка способа достижения цели.

Первоначально решения вырабатывались человеком вручную. С развитием производства и расширением функций человека выработка решений без применения компьютеров стала затруднительной. В связи с этим были созданы системы поддержки принятия решений (СППР).

В настоящее время нет общепринятого, устоявшегося определения СППР, поскольку конструкция СППР существенно зависит от вида задач, для решения которых она разрабатывается, от доступных данных, информации и знаний, а также от пользователей системы.

СППР — интерактивные автоматизированные системы, используемые для различных видов деятельности при принятии решений (в том числе и многокритериальных) в ситуациях, когда невозможно или затруднительно иметь автоматическую систему, полностью выполняющую весь процесс решения вследствие слабой структурированности или неструктурированности решаемых задач.

Современные системы поддержки принятия решения представляют собой системы, максимально приспособленные к решению задач повседневной управленческой деятельности, являются инструментом, призванным оказать помощь ЛПР.

Цель разработки и внедрения СППР — информационная поддержка оперативных возможностей и комфортных условий для высшего руководства и ведущих специалистов при принятии обоснованных решений, соответствующих миссии предприятия, его стратегическим и тактическим целям. Основой функционирования таких систем являются два условия:

1) доставка статистических данных и информации аналитического и сводного характера как из внутренних, так и из внешних источников для экономических и финансовых оценок, сопоставление планов, разработка моделей и составление прогнозов в бизнесе;

2) формирование и эксплуатация во взаимодействии с руководством соответствующей системы информационных, финансовых, математических и эвристических моделей экономических и финансовых процессов.

Концептуальное решение поставленной проблемы должно базироваться на обеспечении доступа к данным и формировании адаптивной системы моделей бизнеса. При этом необходимо обеспечить следующее:

- доступ к данным внутренних и внешних источников информации, использующих серийно выпускаемые базы данных;
- управление данными в разнородных (многоплатформных) комплексах, что позволяет обеспечить их открытость (локализуемость, мобильность);
- хранение данных и информации в унифицированных форматах, пригодных для дальнейшего анализа, синтеза и представления, включая модели «что будет, если ...?»;
- представление информации в виде диаграмм, графиков географических карт в форме, интуитивно понятной и удобной руководству для выработки решений;
- анализ и синтез финансовой и экономической информации, моделирование состояний, процессов и условий.

Решение проблемы в рамках систем поддержки принятия решений отражает уровень ее понимания пользователем и его возможности получить и осмыслить решение. В ряде случаев системы способны пояснять свои рассуждения в процессе получения решения. Очень часто эти пояснения оказываются более важными для пользователя, чем само решение.

Идеальные СППР должны обладать следующими характеристиками:

- позволять человеку управлять процессом принятия решений с помощью компьютера;
- оперировать со слабоструктурированными решениями и быть предназначенными для ЛПР различного уровня с учетом их стиля и методов решения;
- быть адаптируемыми для индивидуального и группового использования;

Рис. 8.1. Классификация СППР

- поддерживать как взаимозависимые, так и последовательные решения;
- быть простыми в использовании и модификации, гибкими к изменениям как организации, так и ее окружения;
- улучшать эффективность процесса принятия решений.

Отличительной чертой современного процесса принятия решений является выработка решений несколькими ЛПР при взаимном влиянии решений отдельных руководителей. ЛПР в этом случае разделены территориально, а решения могут приниматься в разное время. Примером может служить взаимодействие решений диспетчера и руководителей цехов в процессе производства; ректора, деканов и заведующих кафедрами в учебном процессе.

Структура связи ЛПР при этом может быть одноранговой или иерархической.

В одноранговой структуре ЛПР не подчинены друг другу, однако решения одних ЛПР непосредственно и существенно влияют на решения других ЛПР.

В иерархической структуре решения ЛПР более высокого уровня служат ограничениями для решений нижележащих уровней.

СППР характеризуются большим многообразием, что вызывает необходимость их классификации.

Существует множество классификационных признаков, из которых рассмотрим наиболее важные.

С позиции пользователя СППР делят на пассивные, активные и кооперативные.

Пассивные СППР помогают процессу в принятии решения, но они не могут вынести предложение, какое решение принять.

Активные СППР могут сделать предложение, какое решение следует выбрать.

Кооперативные (*распределенные*) СППР позволяют ЛПР изменять, пополнять или улучшать решения, предлагаемые системой, посылая затем эти изменения в систему для проверки и возвращая их пользователю. Процесс продолжается до получения согласованного решения.

На концептуальном уровне выделяют СППР, управляемые сообщениями, данными, документами, знаниями и моделями.

СППР, *управляемые сообщениями* (Communication-Driven DSS), поддерживают группу пользователей, работающих над выполнением общей задачи.

СППР, *управляемые данными* (Data-Driven DSS), или СППР, ориентированные на работу с данными (Data-oriented DSS), в основном ориентируются на доступ к данным и манипуляции с ними.

СППР, *управляемые документами* (Document-Driven DSS), осуществляют поиск и манипулируют неструктурированной информацией, заданной в различных форматах.

СППР, *управляемые знаниями* (Knowledge-Driven DSS), обеспечивают решение задач в виде фактов, правил, процедур.

СППР, *управляемые моделями* (Model-Driven DSS), характеризуются в основном манипуляциями с математическими моделями (статистическими, финансовыми, оптимизационными, имитационными).

В зависимости от данных, с которыми эти системы работают, СППР условно можно разделить на оперативные и стратегические (рис. 8.1).

Оперативные СППР предназначены для немедленного реагирования на изменения текущего состояния системы управления финансово-хозяйственными процессами компании. СППР этого типа получили название информационных систем руководства (ИСР).

Для ИСР характерны следующие основные черты:

- отчеты базируются, как правило, на стандартных для организации запросах, число которых относительно невелико;
- отчеты предполагают достаточно глубокую оперативную проработку данных, предоставляются в максимально удобном виде

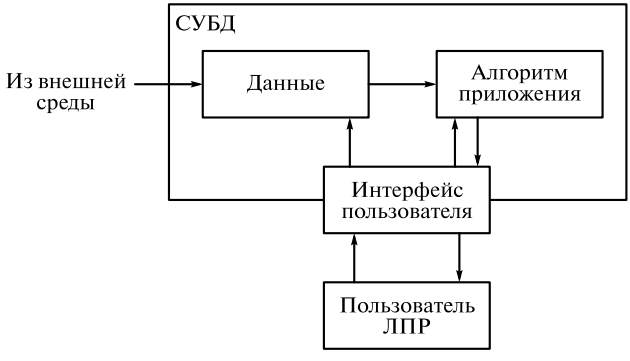


Рис. 8.2. Схема СППР

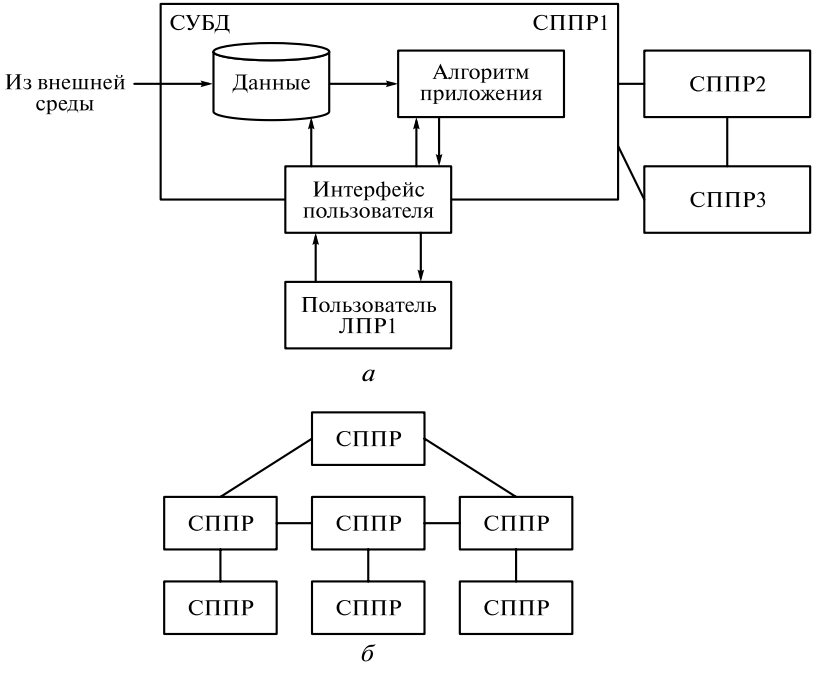


Рис. 8.3. Схема распределенной СППР

(включают в себя таблицы, деловую графику, мультимедийные возможности, правила принятия решений) и позволяют удобно использовать результаты в ходе процесса принятия решений.

ИСР, как правило, ориентированы на конкретный рынок, например, финансы, маркетинг, управление ресурсами.

Стратегические СППР предназначены для обоснования долгосрочных решений на основе агрегированных данных с использованием факторов устойчивого роста компании и снижения рисков. Важнейшей целью этих СППР является прогнозный поиск рациональных вариантов развития компании с учетом влияния различных факторов, таких как конъюнктура целевых рынков, изменения финансовых рынков и рынков капиталов, изменения в законодательстве. Стратегические СППР ориентированы на анализ значительных объемов разнородной ретроспективной информации, собираемой из различных источников.

Ограничимся рассмотрением наиболее распространенных оперативных СППР, которые разделяют на СППР *проектирования* и СППР *функционирования* (см. рис. 8.1). В СППР функционирования структура и функциональное наполнение структурных элементов уже существует и задача заключается в том, чтобы их изучить. В СППР проектирования следует сначала сформировать структуру,

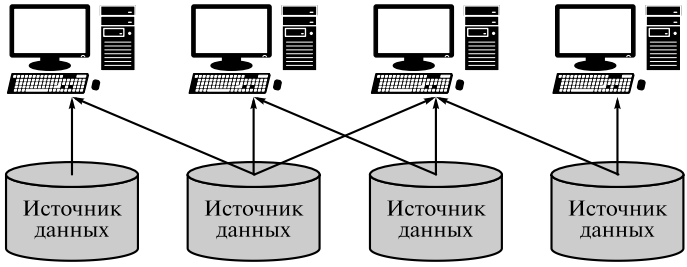


Рис. 8.4. Функциональная СППР

а затем выбрать методы ее функционального наполнения. В силу сказанного акцент сделаем на рассмотрении последней.

При принятии решений в общем случае выделяют следующие этапы:

- выбор цели;
- формирование вариантов (альтернатив) достижения цели;
- исследование ресурсов;
- составление модели;
- определение критериев;
- сравнение альтернатив по принятому критерию и принятие решения;
- реализация решения.

Как при проектировании, так и при функционировании выделяют процессы планирования и управления. Эти процессы различаются по структуре: разомкнутая структура в процессе планирования и, как правило, замкнутая структура в процессе управления. Часто названные процессы интегрируются.

Системы поддержки принятия решений — это компьютерные системы, почти всегда интерактивные (рис. 8.2).

Пользователь взаимодействует с СППР через пользовательский интерфейс, выбирая набор данных и используя для расчетов частную модель. Затем СППР представляет результаты пользователю через тот же самый интерфейс. СППР специализированы по специфическим решениям или классам решений типа маршрутизации, формирования очередей, оценки. На рис. 8.2 присутствует одно ЛПР. Назовем такую СППР *автономной*.

В современных системах все чаще взаимодействуют несколько ЛПР, образуя распределенную СППР (рис. 8.3).

Структура связей может быть одноранговой (рис. 8.3, *а*) или иерархической (рис. 8.3, *б*).

Основная концепция СППР — дать пользователям инструментальные средства, необходимые для анализа важных блоков данных, используя легкоуправляемые сложные модели, гибким способом.

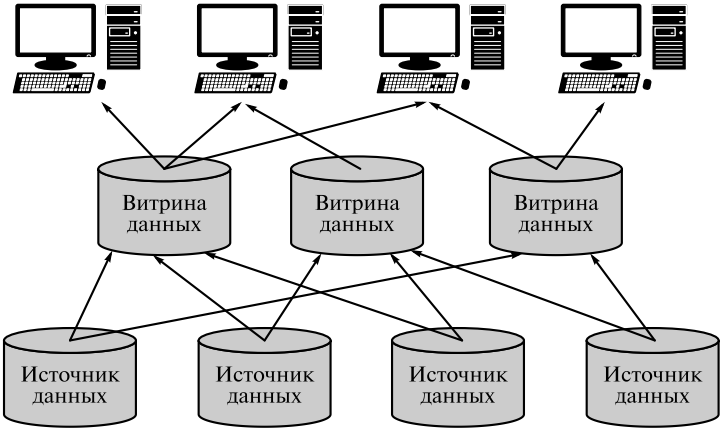


Рис. 8.5. Независимые витрины данных

Методы поддержки принятия решений, реализованные в системе, дают возможность решить следующие задачи:

- формализовать процесс нахождения решения на основе имеющихся данных (процесс порождения вариантов решения);
- ранжировать критерии и дать критериальные оценки физическим параметрам, влияющим на решаемую проблему (возможность оценить варианты решений);
- использовать формализованные процедуры согласования при принятии коллективных решений;
- выбрать вариант, приводящий к решению проблемы.

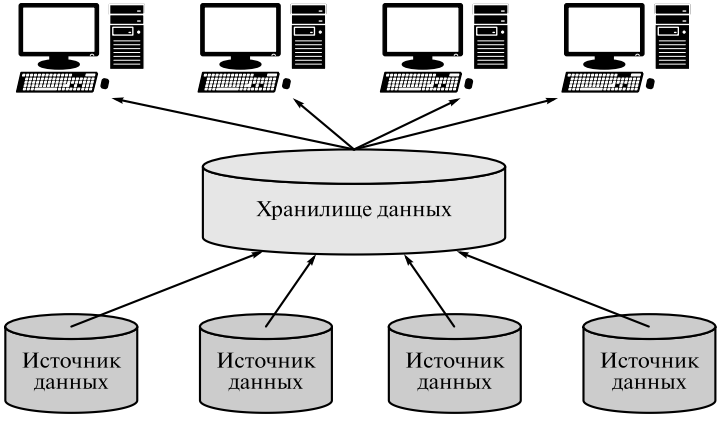


Рис. 8.6. Двухуровневое хранилище данных

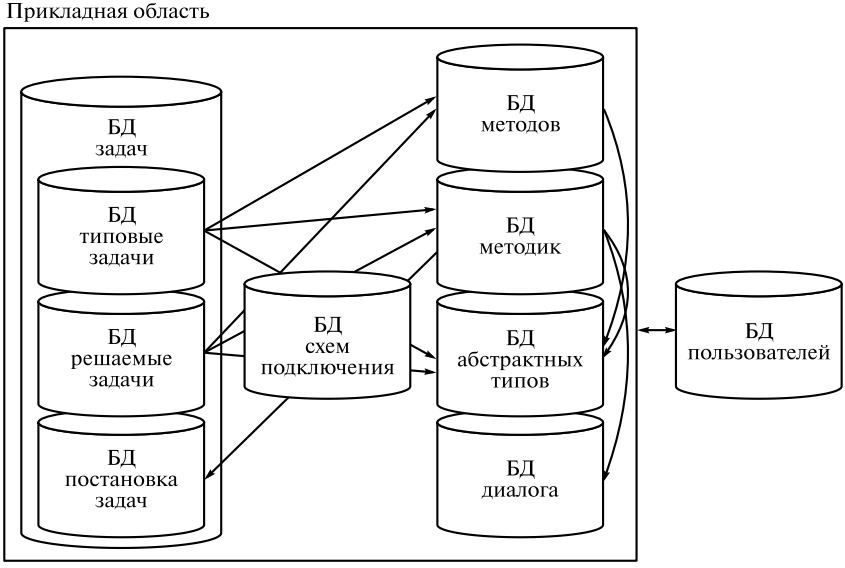


Рис. 8.7. Информационная структура СППР

Процесс принятия решения — это получение и выбор оптимальной альтернативы с учетом анализа последствий.

Алгоритм принятия решения отдельного ЛПР может быть многоуровневым или одноуровневым.

Алгоритм приложения может быть по своей структуре с разомкнутым (планирование) или с замкнутым циклом (управление).

Основу СППР составляет комплекс взаимосвязанных моделей с соответствующей информационной поддержкой исследования.

Можно выделить четыре наиболее популярных типа архитектур систем поддержки принятия решений: функциональная СППР; независимые витрины данных; двухуровневое хранилище данных; трехуровневое хранилище данных.

Функциональная СППР (рис. 8.4) является наиболее простой с архитектурной точки зрения. Такие системы часто встречаются на практике. Анализ в них осуществляется с использованием данных из оперативных систем.

Для архитектуры характерны быстрое внедрение из-за отсутствия этапа перегрузки данных в специализированную систему и минимальные затраты за счет использования одной платформы. В то же время единственный источник данных потенциально сужает круг вопросов, на которые может ответить система.

Отсутствие этапа очистки данных приводит к низкому качеству данных с точки зрения их роли в поддержке принятия стратегических решений. Большая нагрузка на оперативную систему при слож-

ных запросах может привести к остановке работы оперативной системы.

Независимые витрины данных (рис. 8.5) применяются в крупных организациях при работе со сложными системами с большим количеством независимых пользователей.

Затраты на разворачивание витрин данных незначительны, данные в витрине оптимизированы для использования определенными группами пользователей, что облегчает процедуры их наполнения и способствует повышению производительности.

Вместе с тем данные хранятся многократно в различных витринах, что приводит к дублированию данных, увеличению расходов на хранение и потенциальным проблемам, связанным с необходимостью поддержания непротиворечивости данных. Достаточно сложен процесс наполнения витрин данных при большом количестве источников данных, к тому же данные не консолидируются на уровне организации (или системы).

Двухуровневое хранилище данных (рис. 8.6) строится централизованно для предоставления информации в рамках системы в целом.

Здесь данные хранятся в единственном экземпляре с минимальными затратами на хранение данных при отсутствии проблемы синхронизации нескольких копий данных. Данные консолидируются на уровне системы.

Однако данные не структурируются для поддержки потребностей отдельных пользователей или групп пользователей; возможны проблемы с производительностью системы, трудности с разграничением прав пользователей на доступ к данным.

Общая структура информационного обеспечения подобной системы приведена на рис. 8.7.

8.2. Методология и этапность разработки систем

Концепция разработки автоматизированных информационных систем управления зародилась в 1950 — 1960-х гг. На первом этапе основным подходом в проектировании и разработке информационных систем (в том числе СППР) был метод «снизу-вверх», когда система создавалась как набор приложений, наиболее важных в данный момент для решения тех или иных производственных задач или задач автоматизированного управления.

Основной целью этих проектов было создание не тиражируемых продуктов, а средства, позволяющего решать текущие задачи. Такой подход отчасти сохраняется и в настоящее время. В рамках современных концепций автоматизации достаточно хорошо обеспечивается поддержка отдельных функций, но практически полностью от-

сутствует стратегия развития комплексной системы автоматизации. Объединение функциональных подсистем превращается в самостоятельную и достаточно сложную проблему.

Создаваемые в настоящее время СППР в большинстве своем становятся изолированными системами. Периодические изменения технологий разработки и структуры объектов управления, а также сложности, связанные с разными представлениями пользователей об одних и тех же данных, приводят к непрерывным доработкам программных продуктов для удовлетворения новых потребностей пользователей и разработчиков. В связи с этим и работа программистов, и создаваемые СППР (СПР, ИС, ЭС) в большинстве своем не удовлетворяют заказчиков.

Развитие современных технологий проектирования СППР связано и с тем, что в настоящее время сложилась ситуация, характеризующаяся выраженной потребностью в достаточно стандартных программных средствах автоматизации деятельности различных учреждений и предприятий. Из всего спектра проблем разработчики выделили наиболее заметные: автоматизацию ведения аналитического учета в различных областях и автоматизацию технологических процессов. Системы начали проектироваться «сверху-вниз», руководствуясь идеей, что одно программно-аппаратное средство должно удовлетворять потребности различных разработчиков и пользователей концептуально подобных систем.

Сама идея использования универсальных средств накладывает существенные ограничения на возможности разработчиков по формированию структуры баз данных и баз знаний, экранных форм, по выбору алгоритмов расчета. Заложенные при проектировании жесткие рамки не дают возможности гибко адаптировать систему к специфике конкретного применения, учитывать необходимую глубину аналитического и технологического процессов, включать необходимые процедуры обработки данных, обеспечивать интерфейс каждого рабочего места с учетом функций и технологии работы конкретного пользователя. Решение этих задач требует серьезных доработок системы.

Согласно статистическим данным, собранным Standish Group (США), из 8 380 проектов, обследованных в США в 1994 г., неудачными оказались более 30 % проектов, общая стоимость которых превышала 80 млрд долларов. При этом оказались выполненными в срок лишь 16 % общего числа проектов, а перерасход средств составил 189 % запланированного бюджета.

В то же время, современный этап развития СППР и ЭС характеризуется тем, что заказчики стали выдвигать все больше требований, направленных на обеспечение возможности комплексного использования готовых программно-аппаратных средств для решения сходных задач проектирования, анализа, управления.

Возникла насущная необходимость формирования новой методологии и технологий построения информационных систем в целом и СППР в частности. Цель современных методологии и технологий проектирования сложных программно-аппаратных систем должна заключаться в регламентации процесса проектирования СППР и обеспечении управления этим процессом с тем, чтобы гарантировать выполнение требований как к самой СППР, так и к характеристикам процесса разработки. Основными задачами, решению которых должна способствовать методология и технологии проектирования, могут быть следующие:

- обеспечение создания СППР, отвечающих целям и задачам организации-заказчика, а также предъявляемым требованиям по автоматизации процессов анализа и управления;
- гарантированное создание системы с заданным качеством в заданные сроки и в рамках установленного бюджета проекта;
- поддержка удобной дисциплины сопровождения, модификации и наращивания системы;
- обеспечение преемственности разработки, т. е. использование в разрабатываемой СППР существующей информационной инфраструктуры (накопленного опыта в области информационных технологий).

Внедрение технологий должно приводить к снижению сложности процесса создания СППР за счет полного и точного описания этого процесса применения современных методов и технологий создания информационных систем (СППР) на всем жизненном цикле — от замысла до реализации проекта.

Проектирование СППР чаще всего охватывает три основные области:

1) проектирование объектов данных, которые будут реализованы в базе данных (базе знаний);

2) проектирование программ, экранных форм, отчетов, которые будут обеспечивать выполнение запросов к данным;

3) учет конкретной среды или технологии, а именно: топологии сети, конфигурации аппаратных средств, используемой архитектуры (файл-сервер или клиент-сервер), параллельной обработки, совокупности управляющих и информационных сигналов, распределенной обработки данных и т. п.

Цель проекта СППР можно определить как решение ряда взаимосвязанных задач, включающих в себя обеспечение функциональности системы и уровня ее адаптивности к изменяющимся условиям функционирования; пропускной способности системы; времени реакции системы на запрос; безотказной работы системы; необходимого уровня безопасности; простоты эксплуатации и поддержки системы.

Создание СППР представляет собой процедуру построения и последовательного преобразования ряда согласованных моделей на всех

этапах жизненного цикла (ЖЦ) ИС. Современные технологии могут использоваться на каждом из этапов ЖЦ изделия — СППР. С их помощью становится возможной разработка специфичных моделей: модели требований к ИС, проекта ИС, требований к приложениям. Модели формируются рабочими группами команды проекта, сохраняются и накапливаются в репозитарии проекта. Создание моделей, их контроль, преобразование и предоставление в коллективное пользование осуществляется с использованием специальных программных инструментов — CASE-средств.

Процесс создания СППР может быть разделен на ряд этапов (стадий), ограниченных некоторыми временны́ми рамками и заканчивающихся выпуском конкретного продукта (моделей, программных продуктов, документации). Обычно выделяют следующие этапы создания вариантов СППР: формирование требований к системе; проектирование; программная реализация; тестирование; ввод в действие; эксплуатация; сопровождение.

Заметим, что это фактически те же этапы, которые рассматривались при изучении экспертных систем.

Начальный этап процесса создания СППР — моделирование процессов управления, протекающих в системе, частью которой является СППР, и реализующих ее цели и задачи. Модель системы (мультисервисной сети, системы транковой или сотовой связи), описанная в терминах и функциях СППР, позволяет сформулировать основные требования к разрабатываемой ИС. Такой подход обеспечивает объективность в выработке требований к проектированию системы. Множество моделей описания требований к ИС затем преобразуется в систему моделей, описывающих концептуальный проект ИС. Формируются модели архитектуры ИС, требований к программному обеспечению (ПО) и информационному обеспечению (ИО). Затем формируется архитектура ПО и ИО, выделяются структуры БД и БЗ и отдельных приложений, формируются модели требований к приложениям и проводятся их разработка, тестирование и интеграция.

Задача формирования требований к ИС является одной из наиболее ответственных, трудно формализуемых, наиболее дорогих и тяжелых для исправления в случае ошибки. Использование современных технологий проектирования и инструментальных средств позволит достаточно быстро создавать СППР по готовым требованиям. Однако неточное или неполное определение требований к СППР на этапе анализа может привести к многочисленным доработкам и резкому удорожанию фактической стоимости ИС в целом.

Таким образом, современные информационные технологии и технологии работы со знаниями (теория разработки ЭС, соответствующие инструментальные средства, средства OLAP, различные средства моделирования процессов и предметных областей — UML, OWL) позволят на этапе проектирования сформировать модели данных,

обеспечить проектировщиков результатами анализа предметной области в качестве исходной информации. Построение логической и физической моделей данных является основной частью проектирования базы данных. Полученная в процессе анализа информационная модель сначала преобразуется в логическую, а затем в физическую модель данных.

Главная цель проектирования процессов заключается в отображении функций, полученных на этапе анализа, в модули информационной системы. При проектировании модулей определяют интерфейсы программ: разметку меню, вид окон, горячие клавиши и связанные с ними вызовы. Конечные продукты этапа проектирования — схема базы данных (например, на основании ER-модели, разработанной на этапе анализа); набор спецификаций модулей системы, построенных на базе моделей функций.

На этапе проектирования может осуществляться разработка архитектуры СППР, включающая в себя выбор платформы (платформ) и операционной системы (операционных систем). В неоднородной СППР могут функционировать несколько компьютеров на разных аппаратных платформах и под управлением различных операционных систем.

Использование современных информационных технологий в составе технологии проектирования СППР позволит оптимизировать обычно распределенный во времени этап тестирования системы. Например, после завершения разработки отдельного модуля системы может быть сформирован автономный тест, реализующий две основные цели: обнаружение отказов модуля (жестких сбоев); соответствие модуля спецификации (наличие всех необходимых и отсутствие лишних функций). После успешного прохождения автономного теста модуль включается в состав разработанной части системы и группа сгенерированных модулей проходит тесты связей, которые должны отследить их взаимное влияние.

Далее группа модулей тестируется на надежность работы, проходя тесты имитации отказов системы и тесты наработки на отказ. Первая группа тестов показывает, насколько хорошо система восстанавливается после сбоев программного обеспечения, отказов аппаратного обеспечения. Вторая группа тестов определяет степень устойчивости системы при штатной работе и позволяет оценить время безотказной работы системы. В комплект тестов устойчивости могут входить тесты, имитирующие пиковую нагрузку на систему.

8.3. Использование онтологий при проектировании систем

Основная задача онтологий — определение значений терминов, используемых для описания состояния и возможных решений. Основ-

ные идеи использования онтологий основаны на следующей последовательности шагов:

- описание системы независимо от платформы, которая ее поддерживает;
- независимое описание платформы;
- выбор конкретной платформы для системы;
- трансформирование описания системы для удовлетворения требований конкретной платформы.

Для поддержки этого процесса используют четыре типа описаний, реализуемые через онтологии.

Первое описание строится на основании точки зрения, независящей от способов вычисления, используемых в модели. С этой позиции описывается среда, в которой система должна функционировать, и требования к системе. Это описание определяет условия, в которых система будет применяться. Впоследствии эта модель системы может применяться для принятия решения о необходимости использования данной системы в определенной ситуации.

Второе описание системы строится на основе моделей, не зависящих от конкретной платформы и описывающих функционирование системы на таком абстрактном уровне, на котором не требуется отражать особенности реализации системы на конкретной платформе.

Третье описание определяет модель платформы, содержащую множество технических понятий, используемых для описания реализаций систем на данной платформе. Оно учитывает технические особенности реализации абстрактных понятий, используемых в моделях второго описания, на конкретной платформе.

Четвертое описание специализирует второе описание с использованием описания конкретной платформы.

Необходимость онтологий в MDA особенно заметна при оперировании с такими абстрактными понятиями, как система, процесс или ресурс. Значение таких терминов может варьировать в широком диапазоне, и задачей онтологий — обеспечить единое понимание термина всеми участникам процесса разработки и пользователями системы.

Разработка онтологии предметной области требует проведения концептуального анализа. Результаты анализа можно использовать для организации взаимодействия между различными компонентами. Разработка на основании онтологий интерфейсов позволит избежать широко распространенных ошибок в иерархиях классов, когда иерархия классов не отражает реальных отношений «класс-подкласс» между элементами предметной области, а основана на отношении расширения функциональности для удобства программиста. Ошибки такого рода могут сделать невозможным развитие системы.

Особенно важно использовать онтологию при разработке баз знаний, которые предполагается расширять и развивать. Предикаты,

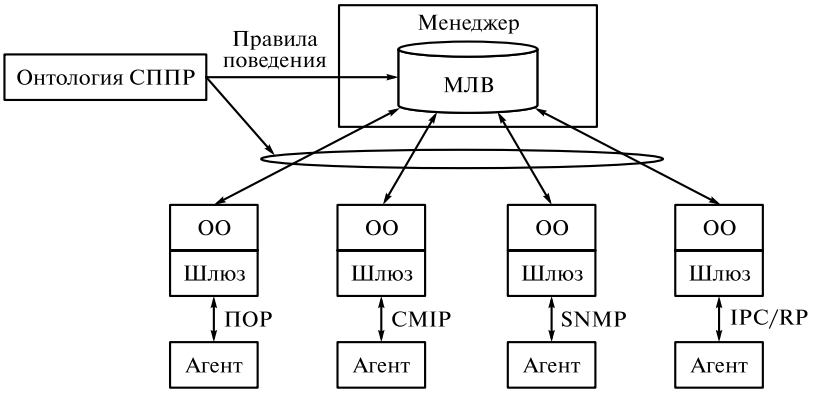


Рис. 8.8. Использование онтологий отображения

фреймы и другие понятия должны либо определяться в онтологии, либо быть встроенными в формат представления знаний.

Онтологии в БЗ дают возможность проводить семантическую трансляцию данных, полученных по разным протоколам, в общую информационную модель, которая не будет зависеть от конкретных протоколов. В результате архитектура может быть снабжена автоматическим построением описания системы.

Такое построение будет осуществляться простым объединением двух компонентов: модели, независимой от платформы (в нашем случае от протокола взаимодействия), и онтологии, описывающей информационную модель конкретных протоколов. Недостаток такого подхода — дополнительные требования к поддержке вычисления таких взаимосвязей на всех компонентах системы, так как одну и ту же информацию можно представить различными способами. В связи с этим заранее будет неизвестно, в каком подмножестве информационной модели будут представлены входные параметры или запрос, обрабатываемый данным компонентом.

Альтернативой является выделение наиболее сложных зависимостей в специальные отображающие онтологии, которые будут удалять избыточные свойства из входных данных, преобразуя их к свойствам, используемым внутри системы. Общая функциональная архитектура системы, построенная на отображающей онтологии, показана на рис. 8.8.

Такая архитектура позволяет менеджеру работать только с одной минимальной информационной моделью. При этом он сможет управлять элементами и пользоваться инструментами, основанными на других информационных моделях.

Система будет осуществлять семантический перевод на основе онтологии отображения (ОО), которая строится специально для каждой информационной модели. Шлюз выполняет задачи посредника

между системами, управляемыми разными информационными моделями, переводя сообщения и команды между ними.

Перспективным направлением является использование онтологий для управления графическим интерфейсом. Поскольку онтология определяет информационную модель, ее можно применять для организации семантически однозначного интерфейса с пользователем. При этом интерфейс сможет осуществлять проверку введенных данных, динамически реагировать на изменение состава и названий свойств.

Например, при описании сетевого коммутатора пользователю может предлагаться введение всех свойств, необходимых СППР для работы с коммутатором. Использование такого интерфейса на этапе функционирования системы приведет к описанию всех компонент в формате, заданном онтологией СППР.

На основе онтологии можно разрабатывать схему БД компонентов. Если такую БД заполнить стандартными компонентами, то эффективность работы с системой возрастет.

Возможные подходы к использованию онтологии зависят от технологии проектирования.

Конкретный состав и глубину описания терминов сетевой онтологии определяют на основании анализа потенциального использования разрабатываемой информационной системы и потенциально возможного взаимодействия с другими системами. В результате предварительного анализа были выделены наиболее важные компоненты предметной области (рис. 8.9).

Предметная область сетевых технологий содержит в себе десятки мелких областей, моделей и теорий. Чтобы создавать такие комплексные онтологии, их необходимо разбивать на модули. Критерий такого разбиения основан на степени связности терминов внутри онтологии. Все модули должны быть согласованы между собой и поэтому их нужно организовывать в иерархические структуры, возможно с множественным наследованием. Онтологии на верхнем уровне абстракции должны содержать только общие термины, связывающие всю онтологию в единое целое.

Онтологии на более низких уровнях должны уточнять и расширять понятия, определенные на разных уровнях абстракции.

Сложность перевода понятий между онтологиями в основном определяется подобием онтологий верхнего уровня, поэтому при создании онтологии требуется с особым вниманием учитывать все потенциально используемые модели данных.

Для рассматриваемой сетевой онтологии такой моделью является метамодель UML, так как другие сетевые технологии не имеют достаточно развитой концептуальной модели. Однако из-за нарушения критериев разработки онтологии и узкоспециализированных определений ее нельзя положить в основу иерархии. На рис. 8.10 представлена декомпозиция онтологии.

Рис. 8.9. Основные компоненты онтологии проблемной области «Управление сетями и сервисами»

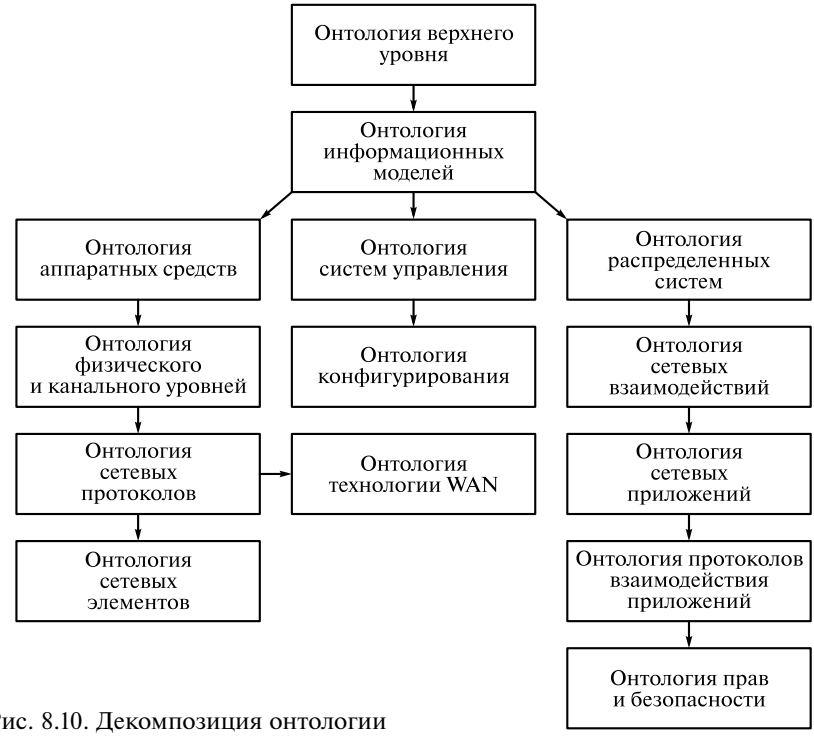


Рис. 8.10. Декомпозиция онтологии

Онтология верхнего уровня определяет такие понятия: момент времени, состояние, объект, группа, человек, событие, деятельность, участник деятельности, роль участника, метод, ресурс, параметр, результат.

Онтология информационных моделей должна быть полностью переводима в диаграммы UML. В силу того, что остальные онтологии будут основаны на данной онтологии, ее необходимо разрабатывать особенно тщательно, так как любая семантическая ошибка повлечет за собой проблемы во всех остальных модулях и может значительно увеличить сложность использования всей онтологии. Для избежания таких ошибок построение онтологии следует осуществлять с использованием специальных методологий построения онтологий. Приведем пример применения методологии.

На *первом шаге* составляют перечень всех терминов, которые необходимо определить в онтологии и дают их неформальное определение. Выделим следующие термины:

System — класс всех систем;

Information system — система, используемая пользователями для создания, обработки, хранения или получения информационных ресурсов;

Component — компонент, являющийся частью системы и использующий набор интерфейсов для взаимодействия с другими компонентами информационной системы. Примеры подклассов компонентов — ActiveX Component, Database, DNS server, Driver, Workstation и Hub;

SoftwareComponent — программный компонент;

SoftwareArchitecturalComponent — компонент в понимании UML;

Port — точка доступа к интерфейсам (используется UML);

Interface — класс объектов, используемых для организации взаимодействия (программный, аппаратный, графический интерфейс);

SoftwareInterface — интерфейс, определяющий группу программных методов;

SoftwareArchitecturalInterface — интерфейс в понимании UML;

SoftwareArchitecturalPort — порт SoftwareArchitecturalComponent в понимании UML;

Hardware Component — аппаратный компонент;

SoftwareArchitecturalNode — любой аппаратный компонент информационной системы, используемый программными компонентами системы (введен для совместимости с UML);

User — пользователь;

Information — абстрактный объект, который может кодироваться в виде наборов данных;

Data — информация, представленная в определенной форме;

Format — форма представления информации, ограниченная некоторыми правилами;

Authoring — деятельность, приводящая к созданию информации;

Processing — деятельность, преобразующая информацию;

Storing — деятельность по хранению данных;

Retrieving — подкласс деятельности по извлечению данных;

Information resource — ресурс, предоставляющий информацию для деятельности;

Representing — деятельность по представлению информации в виде данных;

File — класс всех файлов;

Application — подкласс SoftwareInformationSystem;

Database — база данных;

Interaction — подкласс событий;

Method — метод, используемый в основном для аннотирования деятельностей;

EncodingMethod — класс методов шифрования;

CompressionMethod — класс методов сжатия;

DecompressionMethod — класс методов извлечения информации;

ControlSumAndHashCodeComputationMethod — класс методов вычисления контрольных сумм и хэш-кодов;

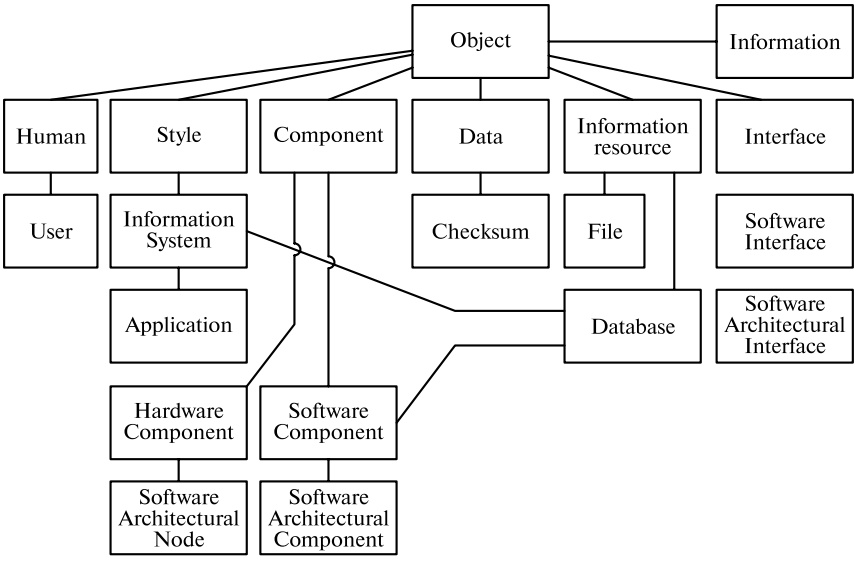


Рис. 8.11. Иерархия классов

Compression — деятельность по сжатию информации;

Decompression — деятельность по извлечению информации;

Encoding — подкласс Processing, реализующий некоторый метод EncodingMethod;

ControlSumAndHashCodeComputation — вычисление CRS и хэш-кодов;

Checksum — подкласс данных, являющихся контрольными суммами и вычисленных некоторым методом для конкретных данных.

На *втором этапе* строят иерархию классов (фрагмент показан на рис. 8.11).

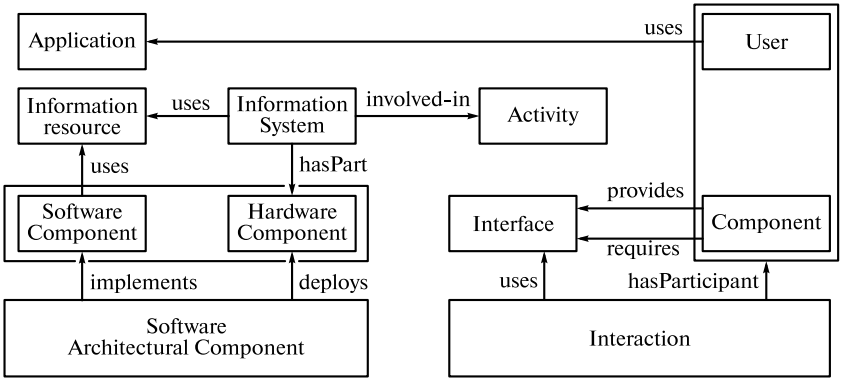


Рис. 8.12. Диаграмма отношений

Таблица 8.1. **Таблица классов**

Класс	Обозначение	Синоним	Объекты
Application	http://onto.nms.ru/nms.owl#Application	Program	IE6.0; Mozila

Окончание табл. 8.1

Класс	Атрибуты класса	Атрибуты объекта	Отношения
Application	—	installPath pluginPath requiredDisk Space	hasPart usesResource involved-in

На *третьем этапе* формируют диаграммы связей между объектами классов (рис. 8.12.).

На *четвертом шаге* строят таблицу классов, определяющую для каждого класса обозначение, список синонимов, полный или частичный перечень объектов, список атрибутов класса и его объектов, список отношений, определенных на классе (табл. 8.1).

На *пятом шаге* строят таблицу отношений и атрибутов, в которой для каждого отношения записаны обозначение, список синонимов, область определения, область значения (табл. 8.2).

На *шестом шаге* строят логические аксиомы и производят их запись в формате OWL.

Проведенный анализ показал, что онтология информационных систем в дальнейшем декомпозируется на четыре почти несвязанных модуля: онтология аппаратных средств, онтология распределенных систем, онтология систем управления, онтология конфигурирования.

Таблица 8.2. **Таблица отношений и атрибутов**

Свойства отношения	Значение
Название	hasParticipant
Обозначение	http://onto.nms.ru/nms.owl#hasParticipant
Область определения	Interaction
Область значения	User или Component
Количество значений	> = 2
Математические свойства	—
Обратное отношение	—

Онтология аппаратных средств определяет термины, необходимые для описания компьютера, состояния процессора, памяти, шин, аппаратных интерфейсов, кабелей, соединительных разъемов, аппаратных сигналов, методов аппаратного кодирования, анализа четности, процедур handshake. Эта онтология далее расширяется онтологиями физического и канального уровней. Полученные модули не используются напрямую в онтологиях распределенных, сетевых систем и в системах управления.

Онтологии сетевых протоколов содержат описание основных сетевых технологий, таких как Ethernet, FDDI, FastEthernet и т. п. В силу специфики технологий WAN их следует выделять в отдельный модуль.

Онтология сетевых элементов содержит описание основных типов оборудования для каждой технологии и библиотеку стандартных компонентов.

Онтология распределенных систем используется как основа для описания сетевых протоколов выше сетевого уровня, сервисов и технологий распределенных вычислений.

Онтология систем управления описывает общие термины: состояние, план, требования, цели, предположения, управляющие воздействия, методы моделирования, диагностики, планирования.

Онтология конфигурирования обеспечивает основную часть терминологии, используемой для описания задачи и библиотечных компонентов. Таким терминами являются переменная состояния, допустимые границы ее значений, динамика изменения переменной состояния, нарушение границ, критерий предпочтения, поправки, библиотечный компонент, конфигурационный параметр.

8.4. Методы описания процессов в системе

Спектр методов математического описания СППР достаточно широк (табл. 8.3), поскольку в системе может быть использован один из методов упомянутых ранее разновидностей интеллектуальных систем.

Методы первых четырех классов подробно описаны ранее, в связи с чем остановимся на применении в СППР методов МАС. Рассмотрим прикладные вопросы построения СППР этого вида.

Java Agent Development Framework (*JADE*) — программная среда, предназначенная для разработки МАС и приложений, соответствующих стандартам FIPA для интеллектуальных агентов. Она состоит из собственно FIPA-совместимой агентной платформы и пакета разработки агентов Java.

Таблица 8.3. **Методы, используемые в СППР**

Класс интеллектуальной системы	Методы
Экспертные системы	Продукции, предикаты первого порядка
Искусственные нейронные сети	Дифференциальные (разностные), алгебраические уравнения
Системы на естественном языке	Онтологии
Интеллектуальные системы управления	Однородный метод на базе динамического линейного программирования
Многоагентные системы (МАС)	Агентная логика

Среда JADE написана на языке Java и состоит из библиотеки Java-классов; она предоставляет разработчикам прикладных программ готовые фрагменты функциональных возможностей и абстрактных интерфейсов. JADE имеет набор инструментальных средств, которые упрощают администрирование и прикладную разработку.

Remote Management Agent (*RMA*) — графическая оболочка, предназначенная для управления и контроля агентной платформы.

Dummy Agent — инструмент контроля и отладки, представляющий собой графический интерфейс пользователя (GUI) и являющийся агентом среды JADE. GUI позволяет составлять сообщения на языке ACL и посылать их другим агентам, отображать список всех отправленных и/или полученных сообщений на языке ACL с указанием информации о метках времени для записи и повторения разговора агента.

Sniffer — агент, который может перехватывать сообщения на языке ACL в то время, когда они выполняются, и графически отображать их в виде последовательности направленных линий, что полезно при отладке.

SocketProxyAgent — простой агент, функционирующий в качестве двунаправленного шлюза между платформой JADE и обычным TCP/IP подключением, ACL-сообщения, передающиеся внутри JADE в собственном внутреннем формате, конвертирующиеся в простые ASCII-строки и отправляющиеся через сокет подключения.

DF GUI — законченный графический интерфейс пользователя. Может использоваться любым другим DF, который необходим пользователю. Таким способом пользователь может создавать сложную сеть областей (доменов) и подобластей «желтых страниц».

Основными особенностями JADE являются следующие:

1) JADE — распределенная агентная платформа, которая может быть распределена среди нескольких компьютеров. Только одно Java-

приложение и только одна виртуальная машина Java выполняются на каждом компьютере. Виртуальная машина Java — основной контейнер агентов, обеспечивающий полную среду выполнения агента; позволяет нескольким агентам одновременно быть выполненными на одном и том же компьютере. Агенты реализованы как потоки (thread) Java;

2) GUI позволяет управлять несколькими агентами и агентными контейнерами агента с удаленного хоста;

3) средства отладки облегчают разработку мультиагентных приложений;

4) поддержка внутриплатформенной мобильности агента, включая состояние и код агента;

5) поддержка одновременного выполнения множества действий через модель поведения агента;

6) FIPA-совместимая агентная платформа;

7) множество FIPA-совместимых DF могут быть запущены во время выполнения для реализации многодоменных приложений. Доменом является логический набор агентов, услуги которых «рекламируются» через общего DF;

8. Эффективная передача ACL-сообщений внутри одной и той же агентной платформы. Сообщения передаются в закодированном виде как объекты Java, что предпочтительнее символьных строк, так как позволяет избежать процесса маршаллинга/демаршаллинга — процесса упаковки/распаковки данных в необходимый формат представления данных процесса-получателя. При пересечении границ платформы сообщение автоматически конвертируется в/из FIPA-совместимый синтаксис, код и транспортный протокол. Преобразование является прозрачным для разработчика агента, который имеет дело только с Java-объектами;

9) библиотека протоколов взаимодействия FIPA, готовых к использованию;

10) автоматическая регистрация и дерегистрация агентов в AMS;

11) FIPA-совместимая служба имен. При запуске агент получает свой Глобальный Уникальный Идентификатор на платформе (Globally Unique Identifier, GUID);

12) поддержка языков содержания и онтологии для конкретных приложений.

Разработан ряд методологий проектирования мультиагентных систем: Multiagent System Engineering (MaSE), Process for Agent Societies Specification and Implementation (PASSI), ADELFE. Все они имеют много общего, поэтому более подробно рассмотрим методологию MaSE.

Методология Multiagent System Engineering (*MaSE*) представляет собой развитие объектно-ориентированного подхода. Агенты — это следующий уровень абстракции по отношению к объектам. Они координируют свое взаимодействие посредством общения.

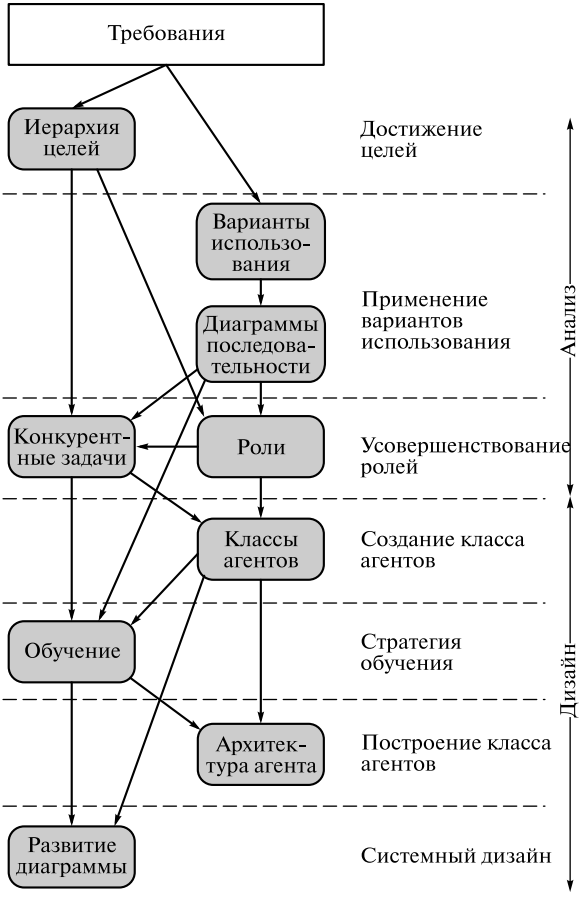


Рис. 8.13. Методология MaSE

Первый уровень описания — системные требования. В результате появляются описания уровня анализа (иерархия целей, диаграммы вариантов использования, описание ролей) и описания уровня дизайна (классы агентов, описание взаимодействий агентов, описание всей совокупности агентов и диаграмма развертывания). Последовательность и взаимосвязь этапов и артефактов отображена на рис. 8.13.

Цель *фазы анализа* — идентифицировать задачи, определяющие в совокупности системные требования и роли. Фаза состоит из трех шагов: определение целей, вариантов использования и ролей.

Для определения целей бизнес-функции системы раскладываются на функциональные требования. С точки зрения методологии MaSE цель — это абстракция, идентифицирующая совокупность не-

которых функциональных требований. Производится выстраивание целей в иерархию на основе отношения цель-подцель.

Описание вариантов использования дают в двух плоскостях: описание структуры (иерархия вариантов) и описание динамики (диаграммы последовательностей) для каждого варианта. Варианты использования указывают, что система должна делать. На диаграммах последовательностей отражаются роли и сообщения, которыми обмениваются роли в ходе взаимодействия.

Уточнение ролей: на этом шаге определяется функциональная декомпозиция системы. Это достигается выделением ролей и ассоциированных с ними задач. С каждой ролью связывается одна или несколько целей из числа выделенных на первом шаге и задачи, реализующие цели.

Назначение *фазы дизайна* — определить, каким образом система должна быть реализована и как должна достигать своих целей. Фаза состоит из следующих шагов: создание классов агентов, проектирование диалогов, сборка совокупности агентов и системный дизайн.

На шаге создания классов агентов определяется общая архитектура системы в терминах агентов и диалогов между ними. Классы агентов создаются для ролей, определенных во время фазы анализа. Каждый агент ассоциируется, как минимум, с одной ролью.

На шаге проектирования диалогов определяются протоколы взаимодействия между парами агентов. Каждое взаимодействие отражается парой диаграмм классов. Первая диаграмма отражает поведение инициатора диалога, а вторая — реакцию отвечающего.

Сборка совокупности агентов подразумевает моделирование внутренней архитектуры агентов.

На шаге системного дизайна проектируется физическая архитектура системы System Design.

С точки зрения удобства разработки для поддержания методологии MaSE разработана библиотека agentMOM, которая может быть использована для генерации кода.

Методология MaSE реализована в продукте Agent Tool, который предоставляет возможности для построения и анализа диаграмм MaSE и автоматического преобразования моделей с генерацией кода заглушек.

Вместе с тем методологии отличаются по назначению.

Методология MaSE предназначена для проектирования закрытых систем, в которых все внешние взаимодействия скрыты за специальными агентами, являющимися частью системы. Создание систем, в которых агенты рождаются и уничтожаются динамически, не предполагается. Межагентное взаимодействие осуществляется только по принципу «один к одному», без широковещательных сообщений.

Методология ADELFE нацелена на проектирование адаптивных МАС, и поэтому большое внимание уделяется изучению всех возможных ситуаций, которые вызывают взаимодействия между аген-

тами в системе, и ситуаций, которые препятствуют таким взаимодействиям. Агенты могут взаимодействовать как непосредственно, так и опосредованно через окружение. Познавательные и поведенческие характеристики агента описываются в терминах способностей (aptitudes), умений (skills) социальных и физических представлений (representations) агента.

Методология PASSI развивает классический подход к проектированию программных систем, основанный на построении некоторого отображения из области требований в область решений. PASSI предлагает еще одну промежуточную область — область агентов, определяющую структуру решения в терминах агентов. PASSI ориентирована на создание FIPA-совместимых решений на таких платформах, как FIPA-OS, Jade.

При обучении СППР могут использоваться и дополнительные методы.

Основное внимание в *статистических методах и пакетах* уделяется классическим методикам — корреляционному, регрессионному, факторному анализу и др. Большинство методов, входящих в состав пакетов, опираются на статистическую парадигму, в которой главными фигурантами служат усредненные характеристики выборки. При исследовании реальных сложных ситуаций они часто являются неадекватными величинами. В качестве примеров наиболее мощных и распространенных статистических пакетов можно назвать SAS (компания SAS Institute), SPSS (SPSS), STATGRAPICS (Manugistics), STATISTICA, STADIA и др.

Основной недостаток *нейросетевой парадигмы* — необходимость иметь большой объем обучающей выборки. Другой существенный недостаток заключается в том, что даже натренированная нейронная сеть представляет собой черный ящик. Знания, зафиксированные как веса нескольких сотен межнейронных связей, совершенно не поддаются анализу и интерпретации человеком. Попытки дать интерпретацию структуре настроенной нейросети (система KINOsuite-PR) выглядят неубедительными. Примеры нейросетевых систем — BrainMaker (CSS), NeuroShell (Ward Systems Group), OWL (HyperLogic).

Идея *систем рассуждений на основе аналогичных случае* такова. Чтобы осуществить прогноз или выбрать правильное решение, эти системы находят в прошлом близкие аналоги наличной ситуации и выбирают тот же ответ, который был для них правильным. В связи с этим метод называют методом ближайшего соседа (nearest neighbour).

Недостаток подобных систем в том, что вообще не создается каких-либо моделей или правил, обобщающих предыдущий опыт. Поэтому невозможно сказать, на основе каких конкретно факторов CBR-системы строят свои ответы. Для систем характерна нечеткость в выборе меры «близости» прецедентов. Системы, использующие

CBR, — KATE tools (Acknosoft, Франция), Pattern Recognition Workbench (Unica, США).

Деревья решений являются одним из наиболее популярных подходов. Популярность подхода связана с наглядностью и доступностью, однако деревья решений принципиально не способны находить «лучшие», наиболее полные и точные правила в данных. Самыми известными являются системы See5/C5.0 (RuleQuest, Австралия), Clementine (Integral Solutions, Великобритания), SIPINA (University of Lyon, Франция), IDIS (Information Discovery, США), KnowledgeSeeker (ANGOSS, Канада).

Эволюционное программирование можно рассмотреть на примере отечественной разработки системы PolyAnalyst. В ней гипотезы о виде зависимости целевой переменной от других переменных формулируются в виде программ на некотором внутреннем языке программирования. Процесс построения программ строится как эволюция в мире программ. Когда система находит программу, более или менее удовлетворительно выражающую искомую зависимость, она начинает вносить в нее небольшие модификации и отбирает среди построенных дочерних программ те, которые повышают точность. Таким образом, система «выращивает» несколько генетических линий программ, которые конкурируют между собой в точности выражения искомой зависимости. Специальный модуль системы PolyAnalyst переводит найденные зависимости с внутреннего языка системы на понятный пользователю язык (математические формулы, таблицы).

Другое направление эволюционного программирования связано с поиском зависимости целевых переменных от остальных в форме функций какого-то определенного вида. Например, в методе группового учета аргументов (МГУА) зависимость ищут в форме полиномов. В настоящее время в России МГУА реализован в системе NeuroShell компании Ward Systems Group.

Генетические алгоритмы (ГА) удобны тем, что их легко распараллеливать. Например, можно разбить поколение на несколько групп и работать с каждой из них независимо, обмениваясь время от времени несколькими хромосомами.

Критерий отбора хромосом и используемые процедуры являются эвристическими и далеко не гарантируют нахождения «лучшего» решения. Примером систем с ГА может служить GeneHunter фирмы Ward Systems Group.

Алгоритмы ограниченного перебора вычисляют частоты комбинаций простых логических событий в подгруппах данных. Примеры простых логических событий: $X = a$; $X < a$; $X > a$; $a < X < b$, где X — какой-либо параметр; a и b — константы. Ограничением служит длина комбинации простых логических событий. На основании анализа вычисленных частот делается заключение о полезности той или иной комбинации для установления ассоциации в данных, классификации, прогнозирования.

Наиболее ярким современным представителем этого подхода является система WizWhy предприятия WizSoft. Автор WizWhy утверждает, что система обнаруживает все логические if-then правила в приемлемое время только для сравнительно небольшой размерности данных.

Системы для визуализации многомерных данных специализируются исключительно на графическом отображении данных. Примером здесь может служить программа DataMiner 3D словацкой фирмы Dimension5 (5-е измерение). Основное внимание сконцентрировано на дружелюбности пользовательского интерфейса, позволяющего ассоциировать с анализируемыми показателями различные параметры диаграммы рассеивания объектов (записей) базы данных.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое СППР? Каковы их назначение и возможности?
2. Назовите разновидности структуры СППР.
3. Что такое распределенные СППР?
4. Каковы цели разработки СППР?
5. Какими характеристиками должны обладать СППР?
6. Назовите наиболее распространенные архитектуры СППР.
7. Опишите базовую конфигурацию СППР «КУРС».
8. Перечислите основные этапы создания СППР.
9. Дайте определения понятий «формальный процесс», «неформальная процедура».
10. Назовите четыре описания онтологий.
11. Что такое онтология отображений?
12. Зачем нужна декомпозиция онтологий?
13. Из чего состоит отология информационных моделей?
14. Зачем нужны иерархия классов и диаграмма отношений?
15. Перечислите методологии и методы создания СППР на основе МАС.

Глава 9

ТЕХНОЛОГИЯ РЕАЛИЗАЦИИ СИСТЕМ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ

9.1. Базовые технологии

Возрастающая сложность современного программного обеспечения (ПО) привела к созданию компьютерной инженерии (Software Engineering), основной задачей которой является создание эффективных методов разработки сложного ПО.

Известные средства разработки БД на основе структурного подхода разработки ПО к настоящему времени выделены в самостоятельное направление. Они предоставляют мосты в CASE-средства, реализующие UML. Основой для такой интеграции является тот факт, что диаграммы классов UML представляют собой расширение диаграмм сущность-связь, используемых для спецификации структуры базы данных.

Технология создания СППР должна быть ориентирована на использование стандартных методологий и может включать в себя следующие этапы.

1. Моделирование предметной области. Чтобы получить проект ИС, адекватный предметной области, в виде системы правильно работающих программ, необходимо иметь целостное, системное представление модели.

Существуют различные методологии структурного моделирования предметной области, среди которых следует выделить функционально-ориентированные и объектно-ориентированные методологии структурного моделирования предметной области (IDEF0, DFD, SADT-диаграммы, объектный подход с использованием унифицированного языка моделирования UML). Используемые программные средства должны реализовывать тот или иной подход к построению моделей.

2. Формирование перечня программных средств, необходимых для реализации задач СППР и целевой системы. Перечень задач определяется на основании данных, полученных на предыдущем этапе.

3. Разработка структуры базы знаний и баз данных подсистем, наполнение базы знаний прикладной системы и баз данных подсистем. База знаний может содержать информацию о составе аппаратно-программных средств комплекса, типовых схемах построения управляющих комплексов, информацию об алгоритмах функционирова-

ния данных средств, информацию о типовых решениях для блокирования аварийных или нештатных ситуаций, о политиках безопасности системы.

4. Формирование конечной структуры и архитектуры СППР, настройка и конфигурирование элементов. К инструментальным средствам создания прикладной СППР можно отнести любое программное средство, автоматизирующее процесс разработки и начальный этап эксплуатации программного комплекса.

В общем случае инструментарий должен содержать:

- графическую среду для описания и документирования фрагментов целевой системы, среду разработки интерфейсных модулей подсистем СППР;
- подсистему интеграции отдельных компонент комплекса разработки, обеспечивающую управляемость процесса разработки СППР;
- специальным образом организованное хранилище проектных данных, библиотеки типовых решений, аппаратных и программных средств;
- средства проектирования и моделирования, обеспечивающие построение моделей предметной области;
- средства, обеспечивающие построение модели информационной системы в целом;
- средства конфигурирования среды разработки;
- средства документирования процесса разработки;
- средства тестирования комплекса;
- средства управления текущим проектом СППР;
- средства обеспечения безопасности функционирования;
- средства модификации и модернизации проекта.

Существующие инструментальные средства могут быть использованы при разработке СППР и классифицированы следующим образом: CASE-средства; СУБД, серверы БД; генераторы отчетов; системы автоматизированного тестирования; средства разработки и контроля; средства устранения неполадок в системе и антивирусы; системы OLAP; ПО общего назначения; инструментальные средства ЭС.

Рассмотрим возможности некоторых из перечисленных классов инструментальных средств.

CASE-средства. *ErWin*. Разработанное компанией PLATINUM Technology (Logic Works) CASE-средство, позволяющее проектировать структуру базы данных, а затем создавать файлы данных (поддерживаются практически все SQL-серверы и персональные БД). Возможна поддержка связи с наиболее распространенными средствами разработки: Delphi, PowerBuilder, Visual Basic, PROGRESS, Oracle Designer.

BPWin. CASE-средство компании PLATINUM Technology (Logic Works). BPwin — инструментарий, обеспечивающий реинжиниринг бизнес-процессов и проектирование информационных систем в тер-

минах стандартов IDEF0, IDEF3, DFD. Встроенный механизм анализа позволяет оценивать и анализировать затраты на осуществление различных видов активности. Связь с ERwin позволяет сократить время проектирования и разработки сложных информационных систем.

S-Designer. CASE-продукт фирмы Sybase, обеспечивающий построение информационных моделей, интеграцию со средствами разработки и генерацию приложений, построение диаграмм процессов и потоков данных, организацию коллективной разработки.

Oracle Designer. Позволяет моделировать сложные системы средствами построения схем процессов, моделей отношений сущностей, иерархий функций, диаграмм потоков данных.

Rational Rose. CASE-система верхнего уровня, позволяющая моделировать как бизнес-процессы, так и различные компоненты программного обеспечения. Работает с методологиями UML, OMT, Booch'93. Поддерживает C++, Visual Basic, Java, PowerBuilder, Ada, Fort, Rational Rose/Smalltalk.

Arena. Система имитационного моделирования для среды Windows. Под имитационным моделированием понимают создание компьютерной модели реальной или предполагаемой системы и проведение на построенной модели экспериментов в целях описания наблюдаемых результатов и/или предсказания результатов.

Генераторы отчетов. *Oracle Reports*. Средство построения сложных высококачественных отчетов на основе данных сложной структуры. Oracle Reports поддерживает Web-технологии и промышленные стандарты HTML, Cascading Style Sheets и Adobe PDF (Portable Document Format); позволяет объединять данные различных форматов (динамические, табличные, графические), имеет графический интерфейс для построения SQL-запросов. Oracle Reports обеспечивает доступ к различным базам данных, включая Microsoft SQL Server, Sybase, Informix, DB2, Oracle и другие ODBC-совместимые.

Oracle Discoverer. Средство формирования неформализованных запросов и отчетов, представляющее быстрый доступ к информации из реляционных хранилищ данных, витрин данных или систем оперативной обработки транзакций.

Системы автоматизированного тестирования. *Rational Suite PerformanceStudio*. Позволяет выполнять тесты производительности для таких распределенных сетевых приложений, как клиент-серверные системы, ERP-системы, Web-сайты и т.п. Включает в себя средства планирования, записи и отладки тестов, создания и управления расписаниями тестирования, а также средства анализа полученных результатов и подготовки отчетов.

TESTBytes. Средство компании PLATINUM Technology (Logic Works) заполнения БД правдоподобными тестовыми данными. Этот продукт предназначен для разработчиков клиент-серверных приложений и используется на этапе отладки и тестирования систем.

TestFoundation. Rational Test Foundation для Windows 2000 представляет собой набор инструментов и документов, предназначенных для тестирования приложений на совместимость с Windows 2000.

Средства разработки и контроля. *JDataStore*. Хранилище данных, поддерживающее многопользовательские транзакции с реляционными объектами, графикой и потоками данных.

Delphi. Широко известное и очень популярное средство разработки клиентских частей приложений, серверов приложений и персональных систем, базирующееся на языке Pascal.

JBuilder. Средство разработки приложений, работающих с базами данных в среде Internet/Intranet. Основано на языке Java.

Eclipse. Кроссплатформенная среда разработки программного обеспечения, поддерживающая множество языков программирования (C/C + + , JAVA, PYTHON, COBOL). Развитие платформы поддерживается сообществом из более чем 100 фирм, работающих в сфере IT.

Системы поддержки принятия решений (OLAP). *Seagate Info*. Трехуровневая система «клиент-сервер». В Seagate Info реализовано централизованное создание и использование отчетов в любой сетевой среде. Seagate Info позволяет проводить стандартизацию документооборота и отчетности в масштабе предприятия, а модуль Seagate Info OLAP обеспечивает многомерный анализ данных и является инструментом поддержки принятия решений.

Oracle Express. Полный набор средств, предназначенный для построения интеллектуальных систем поддержки принятия решений в масштабе предприятия. Включает в себя сервер, среду разработки, приложения для поддержки OLAP.

Средства устранения неполадок в системе и антивирусы. *eTrust PKI*. Средство безопасного доступа к информации совмещает точное управление доступом, повышенную конфиденциальность и легкость использования.

eTrust Access Control. Защищает критически важные данных и приложений.

InoculateIT. Комплексная защита компьютеров в сети.

ClamAV. Оперативно обновляемый антивирусный пакет распространяется под лицензией GNU GPL.

Инструментальные средства ЭС. По своему назначению и функциональным возможностям инструментальные программы, применяемые при проектировании экспертных систем, можно разделить на четыре достаточно больших категории.

Оболочки экспертных систем (*expert system shells*). При создании оболочки из системы-прототипа удаляются компоненты, слишком специфичные для области ее непосредственного применения. Это упоминавшиеся GURU, G2.

Языки программирования высокого уровня. Одним из наиболее известных представителей таких языков является OPS5. Этот язык

прост в изучении и предоставляет программисту гораздо более широкие возможности, чем типичные специализированные оболочки.

Среда программирования, поддерживающая несколько парадигм (multiple-paradigm programming environment). Средства этой категории включают в себя несколько программных модулей, что дает возможность пользователю комбинировать в процессе разработки экспертной системы разные стили программирования. Среди первых проектов такого рода была исследовательская программа LOOP, которая допускала использование двух типов представления знаний: базирующегося на системе правил и объектно-ориентированного. Позднее наибольшую известность получили KEE, KnowledgeCraft и ART. Эти программы предусматривают множество опций и для последующих разработок, таких как KAPPA и CLIPS, и стали своего рода стандартом.

Дополнительные модули. Средства этой категории представляют собой автономные программные модули, предназначенные для выполнения специфических задач в рамках выбранной архитектуры системы решения проблем. Хорошим примером может служить модуль работы с семантической сетью, использованный в системе VT. Этот модуль позволяет отслеживать связи между значениями ранее установленных и новых параметров проектирования в процессе работы над проектом.

При разработке технологии проектирования СППР целесообразно ориентироваться на программные системы с открытыми лицензиями. Применение подобных программных продуктов, эквивалентных по функциональности коммерческим решениям, позволит снизить стоимость разрабатываемой системы, расширить при необходимости функциональность программных продуктов, обеспечить контроль исходных текстов ПО и переносимость программных средств на альтернативную платформу функционирования.

Если необходимо учесть переносимость и кроссплатформенность СППР, то в качестве основного средства разработки «оболочки» СППР может быть предложен язык программирования сверхвысокого уровня Python 2.3.

Язык *Python* — это свободный интерпретируемый объектно-ориентированный расширяемый встраиваемый язык программирования очень высокого уровня:

- свободный — все исходные тексты интерпретатора и библиотек доступны для любого использования;
- интерпретируемый — использует «позднее связывание»;
- объектно-ориентированный — классическая модель, включая множественное наследование;
- расширяемый — имеет строго определенные API для создания модулей, типов и классов на C и C++;
- встраиваемый — обладает строго определенными API для встраивания интерпретатора в другие программы;

- очень высокого уровня — динамическая типизация, встроенные типы данных высокого уровня, классы, модули, механизм исключений.

Интерпретатор Python реализован практически на всех платформах и операционных системах.

Расширяемость языка означает, что имеется возможность совершенствования языка всеми заинтересованными программистами. Интерпретатор написан на языке С и исходный код доступен для любых манипуляций. В случае необходимости можно вставить его в прикладную программу и использовать как встроенную оболочку.

В состав рабочей конфигурации вошли следующие программные средства:

- язык программирования сверхвысокого уровня Python 2.3,
- графические библиотеки WXWindows, Tcl/Tk, OGL — средства для создания интерфейсов АРМ;
- среда разработки экспертной системы CLIPS 6.20;
- средство хранения данных СУБД PostgreSQL 8.0.

Одно из основных достоинств языка Python — наличие большого числа подключаемых к программе модулей, которые обеспечивают различные дополнительные возможности. Дополнительные модули пишутся на языке С и на самом Python и не встречают существенных трудностей при разработке кода. Этот подход очень хорошо согласуется с заявленной модульной концепцией построения СППР.

К таким модулям относятся следующие:

- Numerical Python — расширенные математические возможности манипуляции с целыми векторами и матрицами;
- Tkinter — построение приложений с использованием графического пользовательского интерфейса (GUI) на основе широко распространенного на X-Windows Tk-интерфейса;
- OpenGL — применение обширной библиотеки графического моделирования двух- и трехмерных объектов Open Graphics Library фирмы Silicon Graphics Inc.

Можно использовать расширения и библиотек Numeric — 1.3.1, Numarray — 1.2.2, PIL — 1.1.4, PyXML — 0.8.4, PyGreSQL — 3.6.2, PyCLIPS — 1.0-R2, PySNMP — 2.0.8.

Одна из важнейших задач процесса проектирования СППР — формирование баз знаний (БЗ), описывающих эвристические стратегии принятия решений. Основные требования к графическому интерфейсу пользователя — дружественность и универсальность. Дружественность интерфейса означает интуитивную ясность, наглядность и простоту при редактировании БЗ. Универсальность предполагает возможность настройки на конкретную предметную область и функциональную полноту при описании различных свойств и отношений между сущностями рассматриваемой предметной области.

Задача редактирования БЗ обычно решается средствами инструментальных программных средств разработки интеллектуальных си-

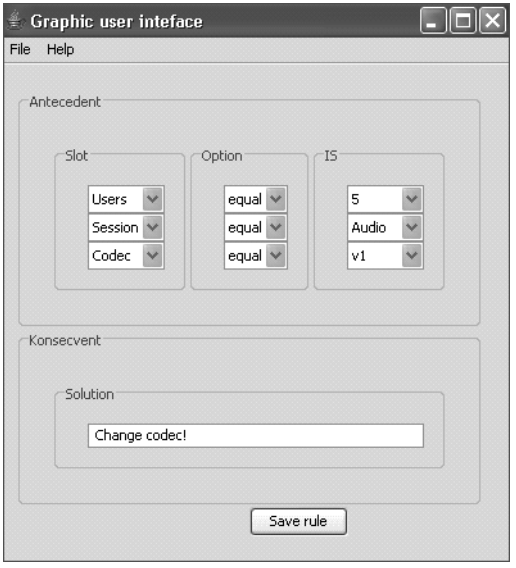


Рис. 9.1. Прототипная реализация пользовательского интерфейса редактора базы знаний СППР

стем. Существует большое число инструментальных средств для создания экспертных систем и СППР.

Среди коммерческих продуктов наиболее известными являются G2 (Gensym, США), RT Works (Talarian, США), ILOG Rules (ILOG, Франция), COMDALE/C (Comdale Techn, Канада), COGSYS (SC, США) .

Все более широкое применение находит подход к описанию общих, разделяемых и повторно используемых моделей предметных областей (онтологий). Однако существующие инструментальные средства, включая CLIPS, не предоставляют возможности формирования БЗ на основе онтологий. На рис. 9.1 представлен вариант редактора базы знаний на основе онтологий GUI.

Перед началом формирования необходимой базы знаний эксперт с помощью меню File должен подключить файл с онтологиями. Списки интересующих его параметров, которые будут использованы при составлении правил, станут доступны. Окно, с которым предлагается работать эксперту, состоит из двух частей. Верхняя часть окна предназначена для ввода левой части правила (антецедента), содержащей поля ввода Slot, Option, IS для ввода соответственно атрибута ситуации, отношения и значения. Эти поля соответствуют одному условному элементу в правиле и должны транслироваться в соответствующую синтаксису базу знаний CLIPS конструкцию defrule. Нижняя часть окна содержит поле Solution для ввода решения, которое должно помещаться в правую часть (консеквент) правила.

Формирование правила начинается с заполнения полей, входящих в антецедент. Для этого эксперт должен выбрать в поле Slot интересующий его параметр из предлагаемого списка, определенного онтологией. Заполнение начинается с первого предлагаемого списка, иначе правило не будет сохранено. Если значение параметра несущественно, эксперт имеет возможность выбрать в списке предлагаемых ему параметров пустое поле, что позволит сформировать меньшее по длине правило.

Выбрав необходимые параметры, эксперт переходит к заполнению поля Option. Выполняется выбор значения параметра в поле IS как допустимого для параметра, выбранного в поле Slot. Например, при управлении мультисервисной сетью для параметра Session (тип сеанса) эксперт будет иметь возможность выбрать из списка значения: Audio, Video, Data.

В нижней части графического интерфейса эксперт выбирает действие, которое необходимо выполнить в данной ситуации. В случае прямого управления задается управляющее воздействие, которое должно быть непосредственно выдано на сетевой элемент. При работе в режиме поддержки принятия решений консеквент правила может содержать ранжированное множество рекомендаций, выдаваемых на консоль оператора (АРМ ЛПР).

Кнопка «Save Rule» используется для добавления правила в БЗ.

Процесс установки и начальной настройки СППР проводится в несколько этапов. Основными из них являются:

- конфигурирование «оболочки» СППР в соответствии с требованиями конкретной прикладной области;
- адаптация СППР к конкретной предметной области;
- начальное формирование и пополнение баз данных и знаний прикладной СППР;
- тестирование и проверка адаптированной для конкретного применения СППР.

Процедура конфигурирования «оболочки» СППР в соответствии с требованиями конкретной прикладной области становится возможной благодаря модульному построению СППР.

Первоначально эта процедура осуществляется на этапе установки ПО СППР. Пользователь может самостоятельно выбрать те или иные компоненты для включения в рабочую конфигурацию и установку на соответствующие вычислительные средства. Это касается установки как отдельных функций АРМ, входящих в состав СППР, алгоритмов принятия решений, различных сервисных функций, так и некоторых АРМ целиком. Однако обязательной является установка, как минимум, двух АРМ: АРМ лица, принимающего решения, и АРМ эксперта.

В ходе адаптации СППР к конкретной предметной области осуществляется подключение необходимых библиотек объектов, используемых в управляемой системе, их свойств, отношений между ними,

правил взаимодействия. Производится подключение соответствующих разделов баз знаний и баз прецедентов, используемых при выработке решений.

Начальное формирование баз данных предполагает выполнение с помощью программного инструментария АРМ эксперта следующих действий:

- визуальное моделирование управляемого объекта;
- составление мнемосхемы, отображающей процессы, протекающие в объекте управления;
- задание свойств и параметров конкретных объектов;
- задание связей между конкретными объектами и их характеристик;
- формирование правил логического вывода и принятия решений;
- задание правил прогнозирования развития ситуаций.

В ходе работы СППР предусмотрена возможность внесения изменений, вызванных изменением топологии объекта управления и правил, в соответствующие разделы БД и базы знаний.

Заполнение и ведение базы прецедентов осуществляются автоматически на основании опыта самой СППР при штатном функционировании.

Сбор диагностической информации в разрабатываемой СППР можно реализовать программными средствами сторонних разработчиков: Nagios, MRTG (multi router traffic grapher), Ethereal, NMap, Nessus, IpTable (или ipfw), Snort.

9.2. Методы и средства обеспечения работоспособности систем

Современные СППР представляют собой органическое единство аппаратной составляющей и программного обеспечения, причем в последнее время четко прослеживается тенденция увеличения роли программной составляющей. Вопросы обеспечения работоспособности и надежности технического обеспечения (ТО) изучены достаточно полно. Этого нельзя сказать о программном обеспечении (ПО).

Перенести разработанные методы оценки ТО на ПО невозможно в силу того, что программное обеспечение более разнообразно и многогранно. В то же время требуется оценка и программного обеспечения, поскольку без него нельзя представить промышленность, систему здравоохранения, образовательные, финансовые и правительственные учреждения. Даже простые системы ПО обладают высокой степенью сложности.

В настоящее время практически во всех организациях обеспечивается контроль таких важнейших характеристик программных про-

дуктов, как время, финансовые средства, ресурсы. В большинстве случаев вне пределов сферы контроля наиболее важная характеристика программных продуктов — это качество продукта.

Проблема качества программного обеспечения имеет, по крайней мере, два аспекта: обеспечение и оценка (измерение) качества.

Все меры по обеспечению надежности программ направлены на то, чтобы свести к минимуму ошибки при разработке, как можно раньше выявить и устранить допущенные ошибки после изготовления программы.

Не существует устоявшегося определения качества ПО. За основу возьмем определение ISO.

Качество — это полнота свойств и характеристик продукта, процесса или услуги, которые обеспечивают способность удовлетворять заявленным или подразумеваемым потребностям.

Составляющими качества информационной системы являются следующие:

- качество инфраструктуры (infrastructure quality) — аппаратного и поддерживающего программного обеспечения (качество операционных систем, компьютерных сетей);
- качество программного обеспечения (software quality) — качество программного обеспечения информационной системы;
- качество данных (data quality), использующихся информационной системой на входе;
- качество информации (information quality), продуцируемое информационной системой;
- качество административного управления (administrative quality) — качество менеджмента, включая качество бюджетирования, планирования и календарного контроля;
- качество сервиса (service quality) — обучения, системной поддержки;
- качество программного обеспечения является составляющей качества информационной системы.

Процесс создания программного обеспечения — это множество взаимосвязанных процессов и результатов их выполнения.

Сравнительные характеристики существующих моделей создания ПО представлены в табл. 9.1.

Многообразие моделей создания ПО не отвергает существования фундаментальных базовых процессов, без реализации которых не может обойтись ни одна технология разработки программных продуктов, а именно разработка спецификации ПО, проектирование и реализация ПО, программирование, аттестация ПО, эволюция ПО.

Разработку спецификации ПО в настоящее время обычно называют разработкой требований. Она включает в себя следующие основные этапы:

- предварительные исследования (оценка степени удовлетворенности существующими системами ПО;

Таблица 9.1. **Сравнительные характеристики моделей создания ПО**

Модель	Достоинства	Недостатки
Каскадная модель: • анализ и формирование требований; • проектирование системы и программного обеспечения; • кодирование и тестирование программных модулей; • сборка и тестирование системы; • эксплуатация и сопровождение системы	Хорошо отражает практику создания ПО и повсеместно используется, когда требования к ПО формализованы достаточно четко и корректно	Негибкое разбиение процесса создания ПО на отдельные фиксированные этапы; трудно отменить или изменить решения более ранних этапов
Эволюционная модель: • разработка первоначальной версии программного продукта; • передача на испытание пользователям и доработка с учетом их мнения; • доработка промежуточной версии продукта (пробные разработки, прототипирование)	Совершенствование ПО в процессе разработки; эффективно учитывает изменения требований заказчика в процессе разработки; наиболее приемлем для разработки небольших программных систем и систем среднего размера	Многие этапы создания ПО не документируются; система часто получается плохо структурированной; часто требуются специальные средства и технологии разработки при быстрой разработке версий программного продукта
Модель формальной разработки: • формальное математическое представление системы; • последовательное и математически корректное трансфор-	Точное соответствие конечной программы спецификации, поскольку дистанция между последовательными преобразованиями значительно меньше дистанции между спецификацией	Сложность и нечеткость выбора для применения соответствующих формальных преобразований

Окончание табл. 9.1

Модель	Достоинства	Недостатки
мирование в программный детализированный код	и программой; метод, состоящий из последовательности небольших формальных шагов, очень удобен для доказательства корректности программного кода для больших масштабируемых систем; удовлетворяет строгим требованиям надежности, безотказности и безопасности	
Модель разработки ПО на основе ранее созданных компонентов: • спецификация требований; • анализ компонентов; • модификация требований; • проектирование системы; • разработка и сборка системы; • аттестация системы	Сокращается количество непосредственно разрабатываемых компонентов; уменьшается общая стоимость создаваемой системы	Отсутствие возможности влиять на появление новых версий компонентов при проведении модернизации системы; неизбежность компромиссов при определении требований

- экономическая эффективность создаваемой системы и бюджетные ограничения на ее разработку;
- формирование и анализ требований;
- специфицирование требований;
- документирование и утверждение требований.

На этапе проектирования определяется структура ПО, данные, которые являются частью системы, интерфейсы взаимодействия системных компонентов. Процесс проектирования можно представить в виде отдельных этапов:

- архитектурное проектирование;
- обобщенная спецификация;
- проектирование интерфейсов;

- компонентное проектирование;
- проектирование структур данных;
- разработка алгоритмов.

Реализация ПО — это процесс перевода системной спецификации в работоспособную систему.

Процесс программирования обычно следует непосредственно за процессом проектирования. Программирование — индивидуальный процесс, не подчиняющийся общим правилам, которым необходимо следовать при написании программного кода.

Аттестация ПО имеет цель показать соответствие системы ее спецификации и ожиданиям и требованиям заказчика и пользователей.

Эволюция программных систем предполагает, что любые программные системы должны модифицироваться в соответствии с изменениями требований заказчика.

Для сокращения времени производства качественного программного обеспечения необходимо оценивать качество программ на этапе разработки. Для этого существует несколько методов.

1. *Вероятностный подход к проблеме надежности*. Надежность, в конечном счете, — понятие статистическое, т. е. предполагается наличие некоторого (достаточно большого) количества одинаковых образцов, испытаний.

Надежность физического устройства меняется со временем: в начале эксплуатации она растет, затем некоторое время остается постоянной и, наконец, начинает уменьшаться из-за эффекта износа, или старения. Надежность аппаратуры определяют по средней фазе, на которой надежность постоянна.

Компьютерная программа не изнашивается, так что последней фазы для нее не существует, однако и первая фаза («приработки» программы) тоже отсутствует. Коррекция программы независимо от причин аналогична внесению изменений в конструкцию физического устройства. В результате получается новое устройство с другим показателем надежности.

2. *Метод «засорения» известными ошибками*. Количество ошибок в программе имеет косвенное отношение к ее качеству:

- число ошибок в программе — величина «ненаблюдаемая», так как наблюдаются не сами ошибки, а результат их проявления;
- неверное срабатывание программы может быть следствием не одной, а сразу нескольких ошибок;
- ошибки могут компенсировать друг друга, так что после исправления какой-то одной ошибки программа может начать «работать хуже»;
- надежность характеризует частоту проявления ошибок, но не их количество;
- известно, что ошибки проявляются с разной частотой и некоторые ошибки остаются невыявленными после многих месяцев и лет эксплуатации;

- нетрудно привести примеры, когда одна-единственная ошибка приводит к неверному срабатыванию программы при любых исходных данных.

Следует также отметить, что если число ошибок рассматривать как меру надежности, то в терминологии теории вероятностей это число есть случайная величина, однако самый главный вопрос (на каком пространстве элементарных событий она задана?) нигде не затрагивался.

Таким образом, число ошибок в программе характеризует скорее не программу, а ее изготовителей и используемый инструментарий.

3. *Следование стандартам ПО при его разработке*. Многие полагают, что если есть стандарт, то для получения качественного ПО достаточно просто четко ему следовать. Это заблуждение.

Основные сомнения, касающиеся стандартизации, таковы:

- стандарты редко появляются вовремя: процесс их создания, как правило, растягивается на столь долгое время, что к моменту публикации они теряют актуальность;
- бытует мнение, что иногда под предлогом борьбы за качество стандарты вводятся как средства конкурентной борьбы;
- стандарты не содержат методик количественной оценки выгод от их применения;
- взаимосвязь вводимого стандарта с лучшими образцами практики и с другими стандартами не всегда понятна.

Безусловно, следовать стандартам необходимо, но этого недостаточно.

Под качеством программного обеспечения следует понимать совокупность свойств, определяющих полезность этой программной системы для пользователей в соответствии с ее функциональным назначением и предъявленными требованиями.

Любая техническая или биологическая система должна включать в себя следующие компоненты: целевую подсистему, предназначенную для выполнения целевых функций; подсистему безопасности, защищающую всю систему от внутренних и внешних воздействий, препятствующих выполнению ее целевых функций; обеспечивающую подсистему, предоставляющую ресурсы для функционирования всей системы.

Отсюда следуют базовые качества любой системы:

- надежность — как свойство точного и своевременного выполнения целевых функций системы;
- безопасность — как свойство защищенности всей системы от внутренних и внешних воздействий, препятствующих выполнению целевых функций;
- обеспеченность — как свойство обеспеченности всей системы ресурсами, необходимыми для точного и своевременного выполнения ее целевых функций.

Таким образом, надежность представляет собой качество целевой подсистемы, безопасность — качество подсистемы безопасности, а обеспеченность — качество обеспечивающей подсистемы.

Существует множество различных подходов к определению показателей качества программных систем.

С одной стороны, оценить качество программного обеспечения может только пользователь, причем такая оценка трудно формализуется. С другой стороны, обеспечить необходимый уровень качества может только разработчик, добившись определенных характеристик конструкции и технологии создания программ. Эти характеристики также трудно формализуются и практически не соответствуют оценкам пользователей.

В большинстве случаев процесс определения показателей качества не связывают с комплексным тестированием несмотря на то, что эти два этапа неразрывны и их нельзя рассматривать обособленно.

Значительная часть методик оценки надежности программного обеспечения опирается на вероятностный подход, заимствованный из методов оценки качества аппаратных средств, что является в корне неверным. Поэтому, как в отечественных, так и в зарубежных стандартах и руководящих документах, вероятностный подход для оценки надежности и безопасности программных систем полностью исключен.

Учитывая все вышесказанное, можно сформулировать главные требования к методике определения качества СППР:

- результативность — методика должна не только давать всестороннюю и объективную оценку, но и использоваться в процессе определения характеристик качества, вырабатывать возможные рекомендации по доработке и совершенствованию программной системы;
- практичность — простота определения показателей качества;
- универсальность — применимость как для разработчика и заказчика программной системы, так и на всех стадиях жизненного цикла программных средств;
- система показателей надежности СППР должна представлять собой многоуровневую иерархию характеристик эксплуатационно-программных свойств программы;
- получение численных оценок должно обеспечиваться на всех уровнях;
- единство методического подхода к оценке для показателей всех уровней;
- высокая степень детализации и глубина проработки;
- гибкость в выборе показателей качества, т. е. возможность введения новых, замены и расширения старых показателей.

В соответствии с этими требованиями и на основе анализа существующих моделей и методов оценивания программных средств разрабатывается методика оценки качества СППР.

Разработка моделей ПО для этапов проектирования и производства потребовала привлечения методов, используемых в различных отраслях знаний, в том числе и в медицине (психологии). Методология моделирования базируется на интуитивном предположении о том, что чем сложнее ПО, тем труднее его спроектировать и создать. Возможное количество дефектов ПО и ряд других параметров ставится в прямую зависимость от логической сложности будущего ПО.

В свою очередь, оценка сложности заказываемого ПО может быть сделана уже на начальной стадии жизненного цикла. Исходными данными для расчетов могут служить требования, содержащиеся в техническом задании на разработку ПО.

Модели рассматриваемой группы базируются на метрической теории программ, разработанной М.Х.Холстедом. Идеи Холстеда основываются на учете особенностей мыслительной деятельности программистов при разработке программ. В рамках указанной теории разработано большое количество метрических моделей, позволяющих оценить ряд свойств ПО АСУ.

Исходными данными для расчетов являются параметры создаваемого ПО, которые можно задать заранее. К ним относятся число входных и выходных переменных, число различных конструкций языка программирования, число требуемых констант. Учет квалификации программистов и способности человека совершать ошибки при мыслительной деятельности проводится путем использования в расчетах психофизиологических параметров (параметр Страуда, психофизиологическая гипотеза Джорджа Миллера «7±2»).

Параметр Страуда характеризует время, необходимое человеческому мозгу для выполнения элементарной мыслительной операции («различения» по Холстеду). Численные значения этой константы находятся в пределах от 5 до 20 различений в секунду.

По гипотезе Миллера мозг человека может обрабатывать в своей «сверхбыстрой» памяти одновременно и безошибочно лишь 7 ± 2 объекта. С использованием данной гипотезы Холстедом показано, что в среднем после обработки 24 бит абстрактной информации на языке высокого уровня человек совершает ошибку (появляется дефект в программе). Точность прогноза, получаемого с помощью моделей, базирующихся на теории Холстеда, находится в пределах 10—12 %. Для этапа проектирования это удовлетворительная величина.

Необходимо отметить, что к настоящему времени метрические модели — единственный класс моделей, с помощью которых оценивают свойства программных средств на этапе проектирования.

Назначение и сравнительные характеристики существующих математических моделей приведены в табл. 9.2.

В последнее время плодотворно развиваются направления по разработке моделей ПО, базирующиеся на анализе текстов программ, их структуры, взаимодействии с техническими средствами. С учетом

Таблица 9.2. **Модели оценивания надежности ПО**

Модели	Достоинства	Недостатки
Учитывающие интервалы времени между моментами проявления ошибок: • случайность интервала времени между проявлениями *i*-й и (*i* – 1)-й ошибками ПО; • параметры распределения случайной величины зависят от числа ошибок, которые присутствуют в ПО в течение этого интервала времени; • оценка неизвестных параметров получается на основе наблюдения реальных временны́х интервалов между последовательными ошибками ПО	Простота определения	Независимость интервалов времени между моментами выявления ошибок. Обнаруженные ошибки мгновенно диагностируются и устраняются. Новые ошибки не вводятся в ПО во время диагностирования и устранения выявленных ошибок. На данном интервале тестирования все ошибки имеют равную вероятность
Учитывающие число выявленных ошибок: • число ошибок ПО, выявленных за определенные временны́е интервалы, как реализация случайного процесса; • оценка интенсивности на основе наблюдения выпавших на интервалах ошибок или на основе самих моментов проявления ошибок; • получение оценки надежности ПО, среднего времени до следующей ошибки после оценки соответствующих интенсивностей	Случайные величины подчиняются распределению Пуассона	Ожидается, что после устранения обнаруженных ошибок интенсивность ошибок снизится. Общее число выявленных ошибок будет расти в зависимости от времени. Интервалы тестирования не зависят друг от друга. Тестирование внутри всех интервалов в приемлемых масштабах однородно и равномерно. Число ошибок, которые выявляются в непересекающихся интервалах, независимо
С «посевом» ошибок: • тестирование ПО на содержание неизвестного числа начальных ошибок;	Используя комбинаторные методы и оценку максимального правдо-	Сложно оценить достоверность результата

• фиксирование выявленных «посеянных» и начальных ошибок; • определение относительного числа выявленных и возможного числа невыявленных начальных ошибок; • получение оценки надежности ПО	подобия, определяют число оставшихся необнаруженными начальных ошибок в ПО	
Основанные на использовании «входной» области: • генерирование слукчайной выборки тестов из заведомо достаточной для отработки «входной области»	Предположение о некоторой входной области исходных данных, которая покрывает все возможные реальные ситуации функционирования ПО	Трудно определима входная область исходных данных. Трудно оценить распределение ограниченной области их использования. Хорошее совпадение теоретических прогнозов с практическими результатами для одной разработки нельзя распространять на другие разработки. Неучет структурных особенностей построения и испытаний ПО. Низкая достоверность получаемых оценок показателей надежности ПО. Применимость для частных специфических систем и невозможность их распространения для широкого класса объектов. Неучет специфики ПО. Использование многочисленных и разнородных оцениваемых параметров

данных факторов имеет место лингвистическая неопределенность, для представления которой применяется нечеткая логика, которая позволяет представить процессы принятия решений и оценки ситуаций человеком в некоторой алгоритмической форме.

В рамках данной методики введена иерархия показателей надежности ПО:

- показатель каждого вышестоящего уровня содержит в качестве составляющих показатели нижестоящего. Интегральный показатель ПО характеризует ее общую полезность;
- эксплуатационные характеристики определяют группы показателей надежности ПО, характеризующие потребительские свойства, которые соответствуют требованиям конечных пользователей;
- показатели последнего уровня характеризуют элементарные программные свойства, подлежащие непосредственной оценке в процессе комплексного тестирования;
- метрические характеристики каждого элементарного свойства определяются на основе одной или нескольких метрик, под каждой из которых подразумевается система определения значений заданного метрического показателя;
- для определения показателей надежности используются экспертные оценки. Для высокой объективности и достоверности оценки необходимо учитывать обобщенные статистические данные по отладке, испытанию и эксплуатации программных средств, подобных оцениваемому программному средству.

9.3. Оболочки и техническая реализация систем

В качестве оболочки СППР на основе алгоритмов экспертных систем предпочтительнее выбирать оболочку G2 (см. гл. 4).

Среди свободно распространяемых инструментов следует выделить среду CLIPS (C Language Integrated Production System), созданную Национальным аэрокосмическим агентством США (NASA) для разработки экспертных систем реального времени. CLIPS реализована на языке C, имеет открытый код и допускает интеграцию с другими программными средствами, может вызываться из других языков, выполнять свои функции и возвращать управление вызвавшей программе.

CLIPS имеет развитую поддержку фрейм-продукционных моделей; поддержку моделей нечеткого вывода и интеграции со средствами разработки онтологий (Protege); многоплатформенность.

Существует Java-реализация среды CLIPS под именем JESS, которая обеспечивает простую интеграцию с различными агентными платформами и средствами разработки.

Своеобразен синтаксис основных конструкций CLIPS.

Одной из основных форм представления информации в CLIPS являются факты. Все текущие факты в CLIPS помещаются в список фактов (fact-list), которые делятся на упорядоченные и неупорядоченные.

Упорядоченный факт состоит из заключенной в скобки последовательности одного поля или нескольких полей, разделенных пробелами.

Неупорядоченные факты представляют собой список взаимосвязанных именованных полей (слотов). Поля в неупорядоченном факте могут быть любыми простейшими типами данных, за исключением первого символьного типа. Первое поле упорядоченного факта специфицирует отношение (свойство), которое применяется к остальным полям факта.

Наличие имен полей позволяет осуществлять доступ к полям по именам, в отличие от упорядоченных фактов, в которых поля специфицируются своим местоположением в факте. Существуют одиночные слоты и мультислоты. Одиночный слот содержит единственное поле, тогда как мультислот может содержать любое число полей.

CLIPS следует рассматривать как наиболее предпочтительный вариант при реализации интеллектуального компонента СППР. Вместе с тем необходимо отметить отсутствие в CLIPS удобного пользовательского интерфейса для наполнения и редактирования БЗ. Тогда эксперту необходимо знать синтаксис основных конструкций, что повышает трудоемкость процесса разработки.

9.4. Построение СППР на основе многоагентного подхода

В промышленных приложениях агентные технологии применяются для решения задач планирования производственных процессов и управления ими на разных уровнях кооперативного проектирования, управления транспортными и телекоммуникационными системами и т.д. В системе YAMS производственное предприятие моделируется как иерархия рабочих ячеек. Ячейки группируются в гибкие производственные системы (ГПС), каждая из которых выполняет обеспечивает определенные функции, а совокупность ГПС группируется в фабрику. Одна компания может иметь много различных фабрик, причем эти фабрики могут дублировать друг друга по своим функциям и возможностям.

Каждая фабрика и каждый компонент фабрики представлены агентами, использующими протокол контрактной сети. Каждый агент обладает набором планов, отражающих его возможности. Протокол контрактной сети позволяет делегировать задачи (производственные задания) индивидуальным фабрикам, а от них — ГПС и затем индивидуальным рабочим ячейкам.

Для управления передачей электроэнергии и управления ускорителем частиц использовались *МАС, построенные на базе системы ARCHON*. Агент в ARCHON представляет собой сложную вычислительную сущность, включающую в себя четыре основных компонента:

1) модуль высокоуровневой связи (МВУС), управляющий межагентными коммуникациями;

2) модуль планирования и координации (МПК), ответственный за стратегию взаимодействия агента с другими агентами;

3) модуль поддержания информации агента (МПИ), ответственный за поддержание внутренней модели мира;

4) базовую интеллектуальную систему (ИнС), представляющую агента — предметного эксперта.

МВУС, МПК и МПИ в совокупности образуют своего рода оболочку агента, которая может использоваться для инкапсуляции существующих интеллектуальных систем и превращения их в агентов.

Например, в разработанной *МАС управления воздушным движением* (*OASIS*) и используемой в аэропорту г. Сиднея, агенты применяют для представления как самолетов, так и различных систем управления воздушным движением. Метафора агентов обеспечивает полезный и естественный способ моделирования автономных компонентов реального мира. Как только самолет попадает в воздушное пространство г. Сиднея, ему назначается программный агент с информацией и целями, соответствующими этому самолету (например, выполнить посадку на определенную полосу в определенное время). Агенты управления воздушным движением ответственны за управление системой.

Коммерческие приложения МАС включают в себя задачи управления информацией, электронную коммерцию, управление бизнес-процессами. Необходимость управления информацией обусловлена стремительным ростом объемов информации и возникающей вследствие этого проблемой информационной перегрузки. Данная проблема имеет два основных аспекта: сбор информации и ее фильтрацию.

Система WARREN представляет собой *МАС управления ценными бумагами*. Она интегрирует информацию, найденную и отфильтрованную в интересах формирования пользователем его фондовых ресурсов (пакета ценных бумаг). Система состоит из агентов, которые кооперативно самоорганизуются для отслеживания имеющихся в наличии квот, финансовых новостей, отчетов финансовых аналитиков и отчетов о прибылях компаний, для того чтобы оперативно оценивать развивающуюся финансовую ситуацию в интересах владельца пакета ценных бумаг.

Разработан электронный рынок (Kasbah), состоящий из покупающих и продающих агентов соответственно для каждого покупаемого или продаваемого товара. Все коммерческие операции выполняются путем взаимодействия этих агентов.

Получение соответствующей согласованной и свежей информации от отделений в рамках большой компании является сложным и требующим времени процессом. В связи с этим организации разрабатывают множество информационных систем, помогающих в управления бизнес-процессами.

Проект ADEPT рассматривает бизнес-процесс как сообщество ведущих переговоры и предоставляющих определенные сервисы агентов. Каждый агент играет отдельную роль в предприятии и способен обеспечивать один или несколько сервисов. Агенты, которым требуются услуги других агентов, вступают в переговоры о получении этой услуги по взаимоприемлемым цене, времени и качеству. Успешные переговоры завершаются заключением контракта между агентами.

Военные приложения явились одной из первых и важнейших областей применения МАС, где их используют при построении систем командования, связи и управления [37], виртуальных систем моделирования поля боя [6], создании автономных беспилотных боевых средств [62] и др. Крупнейшая программа по созданию МАС военного назначения — проект CoABS (Control of Agent-Based Systems), выполняемый в настоящее время рядом ведущих в этой области исследовательских коллективов. Работы в рамках данного проекта структурированы по четырем направлениям:

1) стратегии кооперативного управления;

2) принципы построения систем, методы и алгоритмы управления поведением;

3) архитектуры вычислительных сред для мультиагентных приложений;

4) поддерживающие технологии.

Работы первого направления включают в себя проекты, направленные на разработку эффективных методов координации множества интеллектуальных агентов для систем командования и управления, механизмов многоуровневой координации интеллектуальных агентов реального времени, быстро расширяемых и конструируемых агентов для построения робастных адаптивных коллективов, управление планами и др.

В рамках второго направления ведутся работы, направленные на разработку среды мультиагентного планирования и обучения, методов и средств управления ресурсами в крупномасштабных системах мобильных агентов, службы обработки исключений для ансамблей программных агентов, среды рефлексивных агентов.

Третье направление включает в себя разработку общей операционной среды мультиагентных систем и проведение экспериментальных исследований с различными протоколами межагентного общения.

В рамках четвертого направления создают программные среды, поддерживающие разработку и тестирование мультиагентных систем,

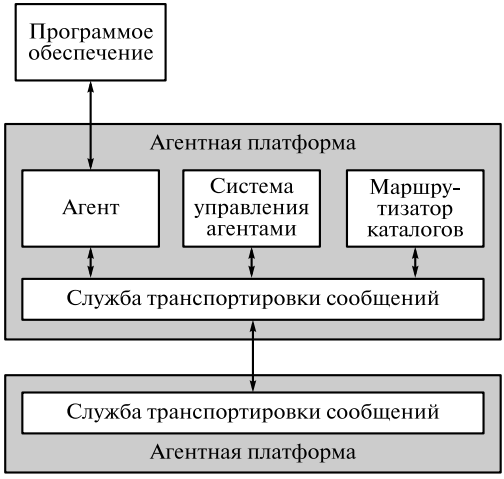


Рис. 9.2. Эталонная модель управления агентами

включая принципы ведения сложных переговоров между разнородными агентами.

Для стандартизации агентных технологий в 1996 г. была создана международная организация Foundation for Intelligent Physical Agents (FIPA), цель которой — разработка открытых спецификаций, поддерживающих взаимодействие агентов и агентных сервисов. Согласно спецификации FIPA агент обладает способностью предоставлять в рамках унифицированной и интегрированной исполнительной модели один или несколько сервисов, которые могут включать доступ к внешнему программному обеспечению, пользователям и коммуникационным возможностям.

FIPA разработала эталонную модель управления агентами, согласно которой управление агентами осуществляется благодаря обязательной инфраструктуре, в рамках которой действуют FIPA-агенты.

Сущности, содержащиеся в эталонной модели (рис. 9.2), представляют собой множество логических возможностей (сервисов) и не накладывают никаких ограничений на физические конфигурации. Детали реализации конкретных агентных платформ (АП) и агентов являются проектным решением разработчиков конкретных агентных систем.

Агент может иметь определенные способности брокера ресурсов для доступа к программному обеспечению. Агент должен иметь хотя бы одного владельца, например на основе организационной принадлежности или принадлежности пользователю, и может поддерживать несколько понятий идентичности.

Идентификатор агента (Agent Identifier — AID) помечает агента уникальным образом, чтобы он мог однозначно отличаться внутри

универсума агентов. Агент может быть зарегистрирован с множеством транспортных адресов, по которым с ним можно контактировать, и он может иметь определенные способности брокера ресурсов.

Служба каталога (Directory Facilitator — DF). Агенты могут регистрировать свои сервисы у DF или обращаться к DF с запросом на поиск сервисов, предоставляемых другими агентами. В рамках одной АП может существовать множество DF, а DF могут объединяться в федерации.

Система управления агентами (Agent Management System — AMS) — обязательный компонент АП, осуществляющий супервизорное управление доступом к агентной платформе и ее использованием. Только одна AMS может существовать на каждой АП. AMS поддерживает каталог идентификаторов агентов (AIDs), содержащих транспортные адреса агентов, зарегистрированных на данной АП. AMS предоставляет другим агентам сервис «белых» страниц. Каждый агент должен зарегистрироваться у своей AMS, для того чтобы получить свой AID.

Служба транспортировки сообщений (Message Transport System — MTS) является определенным по умолчанию методом коммуникации между агентами на различных АП.

Эталонная модель именования FIPA-агентов идентифицирует агента посредством расширяемой совокупности пар «параметр-значение», называемой идентификатором агента (Agent Identifier — AID). AID включает в себя имя и другие параметры (транспортные адреса, адреса сервиса разрешения имени).

AIDs, в первую очередь, предназначены для идентификации агентов внутри конверта (envelope) сообщения, конкретно в параметрах «:to» и «:from».

Значения параметров AID могут редактироваться или модифицироваться агентом, например для обновления последовательности серверов разрешения имени или транспортных адресов в AID. Однако обязательные параметры могут изменяться только агентом, к которому относится данный AID.

Параметр «:name AID» является глобально уникальным идентификатором, который можно использовать как уникальное ссылочное выражение для агента. Одним из простейших механизмов является его построение из актуального имени агента и адреса его домашней агентной платформы, разделенных символом '@'.

Транспортный адрес представляет собой физический адрес, по которому можно контактировать с агентом. Он обычно специфичен для протокола транспортировки сообщений. Конкретный агент может поддерживать несколько методов коммуникации и помещать множество значений транспортных адресов в параметр «:addresses» своего AID.

Разрешение имени — сервис, предоставляемый AMS посредством функции «search». Параметр «:resolvers» AID содержит последователь-

ность AID, в которой AID агента может, в первую очередь, разрешаться в транспортный адрес или множество транспортных адресов.

Каждый агент, желающий опубликовать свои сервисы для других агентов, должен найти подходящий DF и потребовать регистрации своего агентного описания. Должно поддерживаться описание объекта, содержащее значения всех обязательных параметров описания. Оно может также поддерживать необязательные и защищенные параметры, содержащие не стандартизованную FIPA-информацию, которую разработчик агента может включить в каталог. В любое время и по любой причине агент может потребовать у DF модифицировать его агентное описание.

Для описания предметных областей в форме, понимаемой всеми участниками сообщества, используются онтологии. Они определяют общий словарь рассматриваемой области, смысл терминов и отношений между ними. Возможность совместного и многократного использования онтологий обеспечивает открытость информационной среды на семантическом уровне.

Статус официальных рекомендаций W3C имеет два языка (RDF и OWL), используемых для хранения онтологий.

Язык RDF. Язык Resource Description Framework (RDF) представляет собой язык для описания ресурсов на семантическом, «понятном» компьютеру уровне. Под ресурсом понимается любая (физическая или абстрактная) сущность, имеющая уникальный идентификатор — URI. В качестве ресурса могут выступать:

- доступные через сеть ресурсы (электронные документы, изображения, сервисы или группы ресурсов);
- объекты, недоступные непосредственно по сети (люди, корпорации, книги в переплетах);
- абстрактные, не существующие в физическом мире понятия («создатель» (creator)).

Спецификации консорциума W3C определяют для RDF модель данных и XML-синтаксис RDF.

Для именования субъектов, предикатов и объектов в RDF-утверждениях RDF использует URI-ссылки. URI-ссылка представляет собой уникальный идентификатор ресурса (URI) вместе с необязательным *идентификатором фрагмента* в конце, отделенным символом «#». Например, URI-ссылка http://www.example.org/index.html#section2 состоит из URI http://www.example.org/index.html и идентификатора фрагмента section2. RDF URI-ссылки могут содержать символы Unicode, позволяющие отображать в них много языков.

URI-ссылки могут быть абсолютными или относительными. Абсолютная URI-ссылка ссылается на ресурс независимо от контекста, в котором она появляется (например, http://www.example.org/index.html). Относительная URI-ссылка — сокращенная форма абсолютной, в которой отсутствует некоторый префикс и для восполнения недостающей информации требуется знать контекст, в котором по-

является URI-ссылка. Например, относительная URI-ссылка otherpage.html при появлении в ресурсе http://www.example.org/ index.html должна дополняться до абсолютной http://www.example.org/otherpage.html. URI-ссылка без части URI рассматривается как ссылка на текущий документ (документ, в котором она появилась). Пустая URI-ссылка в документе рассматривается как ссылка на сам этот документ.

URI-ссылка, состоящая только из идентификатора фрагмента, рассматривается как эквивалент URI-ссылки на документ, в котором она появляется, с добавленным к ней идентификатором фрагмента. Например, если внутри документа http://www.example.org/index.html появляется относительная URI-ссылка #section2, она будет рассматриваться как эквивалент абсолютной ссылки http://www.example.org/index.html#section2.

Модель данных RDF обеспечивает абстрактный каркас описания ресурсов. Для машинного представления и обработки RDF-модели необходим конкретный *синтаксис*, позволяющий записывать утверждения в виде некоторого текста (выполнять сериализацию модели). Для этой цели используется XML.

Элемент RDF используется в качестве оболочки, отмечающей в XML-документе границы сериализованной RDF-модели. Атрибутами элемента RDF являются объявления пространств имен. Объявление собственного пространства имен RDF записывается так:

```
<rdf:RDF
xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-
syntax-ns#">
/RDF-модель /
</rdf:RDF>
```

В RDF-модели для каждого ресурса, как правило, приводится не одно RDF-утверждение, а описывается сразу несколько свойств. Множество утверждений, относящихся к одному и тому же ресурсу, группируются с помощью элемента Description. Для указания идентификатора ресурса, к которому относится каждое из утверждений, элемент Description использует атрибут about, значением которого является URI-ссылка.

Каждому свойству ресурса сопоставляется свой элемент (тег). Имена элементов (свойств) ассоциируются с пространствами имен с помощью префиксов. Пусть свойство Creator взято из некоторого пространства имен *s*, имеющего URI-ссылку http://description.org/schema/. Тогда простое утверждение, RDF-модель которого представлена на рис. 9.3, запишется с помощью XML синтаксиса так:

```
<?xml version = "1.0"?>
<rdf:RDF
      xmlns:rdf = "http://www.w3.org/1999/02/22-
```

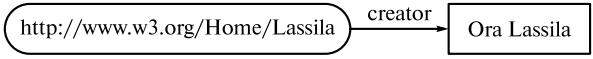


Рис. 9.3. RDF-граф для простого утверждения

```
rdf-syntax-ns#"
   xmlns:s = "http://description.org/schema/">
<rdf:Description rdf:about = "http://www.w3.org/
Home/Lassila">
<s:Creator>Ora Lassila</s:Creator>
</rdf:Description>
</rdf:RDF>
```

Одно из пространств имен можно указать пространством имен по умолчанию, имена из этого пространства будут записываться без префикса. Если в рассматриваемом примере пространством имен по умолчанию указать пространство имен RDF, модель запишется следующим образом:

```
<?xml version = "1.0"?>
<RDF
    xmlns = "http://www.w3.org/1999/02/22-rdf-
    syntax-ns#"
  xmlns:s = "http://description.org/schema/">
<Description about = "http://www.w3.org/Home/
Lassila">
<s:Creator>Ora Lassila</s:Creator>
</Description>
</RDF>
```

Вместо rdf:about в элементе Description может использоваться атрибут rdf:ID. Этот атрибут указывает относительную RDF URI-ссылку, эквивалентную символу "#" с последующим значением атрибута rdf:ID. Например, rdf:ID = "name" будет эквивалентно rdf:about = "#name".

В утверждениях часто возникает необходимость описывать совокупности значений. Например, если работа выполнена коллективом авторов или группа состоит из нескольких студентов, значением свойства является множество значений. Для поддержки таких совокупностей ресурсов или литералов используются RDF-контейнеры (Containers). RDF определяет три типа контейнерных объектов:

- Bag — неупорядоченный список значений;
- Seq — упорядоченный список значений;
- Alt — список альтернативных значений элемента.

Для спецификации элементов, входящих в совокупности rdf:Bag, rdf:Alt и rdf:Seq используется свойство rdf:li ("list item"). В следующем примере свойство <cd:artist> содержит неупорядоченную совокупность значений артистов.

```
<?xml version = "1.0"?>
<rdf:RDF
xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-
syntax-ns#"
xmlns:cd = "http://www.mus-cd.org/cd">
<rdf:Description
rdf:about = "http://www.mus-cd.org/cd/Beatles">
<cd:artist>
<rdf:Bag>
<rdf:li>John</rdf:li>
<rdf:li>Paul</rdf:li>
<rdf:li>George</rdf:li>
<rdf:li>Ringo</rdf:li>
</rdf:Bag>
</cd:artist>
```

В RDF свойства рассматриваются как атрибуты ресурсов или как отношения между ресурсами. Однако сам по себе RDF не предоставляет никаких механизмов для описания этих свойств и их отношений между собой и с другими ресурсами. Эту роль выполняет RDF-схема (RDF Schema, RDFS) — язык описания словаря RDF.

RDFS расширяет RDF *словарем схемы*, включающим в себя такие элементы, как Resource, Class, subClassOf, subPropertyOf и др. RDFS позволяет определять словарь терминов и отношений между ними.

Все описываемые с помощью RDF сущности считаются ресурсами и являются экземплярами класса rdfs:Resource. Таким образом, этот класс — наиболее общий класс в RDF(S). Все другие классы являются его подклассами.

Два основных подкласса rdfs:Resource — rdfs:Class и rdf:Property. Все классы и свойства, определяемые при специфицировании схемы конкретной предметной области, являются экземплярами этих двух подклассов. rdfs:Resource, в свою очередь, является экземпляром rdfs:Class.

Ресурс rdfs:Class означает множество всех классов в объектно-ориентированном смысле. Например, определяемые в схеме некоторого конкретного приложения классы appl:Person или appl:Organisation представляют собой экземпляры метакласса rdfs:Class. Аналогичным образом каждое определяемое в конкретной схеме свойство является экземпляром класса rdf:Property.

rdfs:Literal — класс литеральных значений, таких как строки и целые числа. Литералы могут быть простыми или типизированными. Типизированный литерал представляет собой экземпляр класса типов данных. rdfs:Literal является экземпляром rdfs:Class и подклассом rdfs:Resource.

rdfs:Datatype — класс типов данных, экземпляры которого соответствуют RDF-модели. rdfs:Datatype — одновременно и экземпляр, и подкласс rdfs:Class. Каждый экземпляр rdfs:Datatype является подклассом rdfs:Literal.

RDFS дает возможность задавать ограничения для области определения и области значений свойств. Например, такие ограничения позволяют определить, что только люди могут находиться в отношении marriedWith с другими людьми.

rdfs:range является экземпляром rdf:Property и используется для утверждения, что значения свойства являются экземплярами одного или более классов.

Тройка вида P rdfs:range C «утверждает», что P — экземпляр класса rdf:Property, C — экземпляр класса rdfs:Class. Значением свойства P могут быть экземпляры класса C. Если P имеет более одного свойства rdfs:range, множество его значений является объединением всех классов C, указанных в различных свойствах rdfs:range.

Свойство rdfs:range может быть применено к самому себе. Утверждение rdfs:range rdfs:range rdfs:Class «говорит», что любой ресурс, являющийся значением свойства rdfs:range, является экземпляром rdfs:Class.

rdfs:domain является экземпляром rdf:Property и используется для утверждения, что любой ресурс, имеющий это свойство, является экземпляром одного или более классов. Тройка вида P rdfs:domain C «говорит», что P является экземпляром класса rdf:Property, C является экземпляром класса rdfs:Class и ресурсы, обозначающие субъекты троек, предикатами которых является P, являются экземплярами класса C. Если P имеет более одного свойства rdfs:domain, то ресурсы, обозначенные субъектами троек с предикатами P, являются экземплярами всех классов, с помощью свойств rdfs:domain.

Утверждение rdfs:domain rdfs:domain rdf:Property «говорит», что любой ресурс со свойством rdfs:domain является экземпляром rdf:Property.

Утверждение rdfs:range rdfs:domain rdfs:Class «говорит», что любой ресурс, который является значением свойства rdfs:domain, является экземпляром rdfs:Class.

Свойство rdfs:subClassOf определяет отношение подклассов между классами. Тройка вида C1 rdfs:subClassOf C2 «утверждает», что C1 и C2 являются экземпляром rdfs:Class и C1 является подклассом C2.

Свойство rdfs:subPropertyOf определяет иерархию свойств и используется для утверждения, что все ресурсы, связанные одним свой-

ством, связаны также и другим. Тройка вида P1 rdfs:subPropertyOf P2 «утверждает», что P1 и P2 — экземпляры rdf:Property и P1 является подсвойством P2. Например, свойство fatherOf является подсвойством parentOf.

Свойства rdfs:subClassOf и rdfs:subPropertyOf транзитивны.

Спецификация RDFS определяет следующие элементы: rdfs:label, rdfs:comment, rdfs:Container, rdfs:member, rdfs:ContainerMembershipProperty, rdfs:seeAlso и rdfs:isDefinedBy.

Язык OWL. Язык RDFS имеет достаточно ограниченные выразительные возможности. Более развитым языком описания онтологий для семантического Web является Web Ontology Language (OWL). Язык OWL предоставляет словарь для определения классов, их свойств и отношений между ними. Он позволяет выразить значительно более богатые отношения и поддерживает более развитые возможности вывода, чем RDFS.

Язык OWL включает в себя три подъязыка с различными выразительными возможностями, разработанных для использования разными сообществами разработчиков и пользователей:

1) OWL Lite (облегченный OWL) поддерживает классификационные иерархии и простые ограничения;

2) OWL DL (OWL с дескриптивной логикой) предоставляет выразительные возможности, максимально допустимые при сохранении вычислительной полноты и разрешимости (все вычисления закончатся за конечное время) систем логического вывода. Дескриптивная логика изучает конкретные разрешимые фрагменты логики первого порядка;

3) OWL Full (полный OWL) обеспечивает максимальные выразительные возможности и синтаксическую свободу языка RDF, однако не гарантирует вычислимости.

OWL-документ состоит из необязательных заголовков онтологии и произвольного числа *аксиом классов*, *аксиом свойств* и *фактов* об индивидах, задающих определения соответствующих ресурсов. OWL-код записывается в виде подэлементов элемента RDF, объединяющего элементы с общими пространствами имен XML и базовыми объявлениями.

Пространства имен используются для указания конкретных словарей, из которых будут взяты термины.

При наличии данных URL удобно объявлять сущности в объявлении типа XML-документа (DOCTYPE), предшествующем объявлению онтологии. Имена, пространства имен имеют значения только как части XML-тегов, значения атрибутов не чувствительны к пространствам имен. Однако в OWL идентификаторы часто упоминаются как значения атрибутов и могут записываться в полной расширенной форме.

Элементы OWL-онтологии определяют классы, свойства, экземпляры классов и отношения между ними. Иногда важно различать

класс как объект и класс как множество содержащихся в нем элементов. Классы предоставляют механизм абстракции для группирования ресурсов со сходными характеристиками.

Каждый класс OWL ассоциируется с множеством индивидов (экземпляров), называемым *объемом класса*. Класс имеет и интенсиональное значение (соответствующее понятие), которое связано с объемом класса, но не эквивалентно ему. Таким образом, два класса могут иметь один и тот же объем, но все же быть разными.

Класс OWL можно описать либо посредством имени, либо специфицируя объем неименованного (анонимного) класса. В OWL существует шесть типов описаний классов:

1) идентификатор класса (URI-ссылка);
2) полное перечисление индивидов — экземпляров класса;
3) ограничение свойства;
4) пересечение описаний двух или более классов;
5) объединение описаний двух или более классов;
6) дополнение описания класса.

Все способы, кроме первого, описывают анонимный класс путем наложения ограничений на объем класса.

Для создания и работы с онтологиями необходимы специальные инструментальные программные средства. Анализ существующих редакторов онтологий показал, что самым перспективным из них является Protégé-2000, разработанный в Стэндфордском университете.

Protégé-2000 является Java-инструментом с открытым кодом, предоставляющим расширяемую архитектуру для создания и пользовательской настройки основанных на знаниях приложений. Расширяемость обеспечивается открытой модульной архитектурой системы, которая дает возможность сторонним разработчикам добавлять новую функциональность в систему путем создания своих модулей. Накоплена значительная библиотека, содержащая более 80 дополнительных модулей. Большинство этих модулей можно отнести к одному из следующих типов: модули поддержки различных форматов записи онтологий; модули визуализации информации; модули, добавляющие новую функциональность в работе с некоторыми специфичными базами знаний.

Среди модулей первого типа есть модули поддержки работы с языками онтологий: RDF(S), OIL, DAML + OIL и OWL.

Модель знаний в Protégé-2000 включает в себя поддержку создания классов и иерархии классов с множественным наследованием, создание различных типов слотов, определение различных свойств слотов. Поддерживаются метаклассы и иерархии метаклассов.

Главными достоинствами Protégé-2000, которые выделяют его среди других редакторов онтологий, являются масштабируемость и расширяемость. Масштабируемость обеспечивается продуманной системой хранения и позволяет создавать онтологии, содержащие в себе сотни тысяч понятий. Расширяемость обеспечивается открытой мо-

дульной архитектурой системы, которая позволяет сторонним разработчикам добавлять новую функциональность в систему путем создания своих модулей. К настоящему моменту уже накоплена значительная библиотека, содержащая около 80 дополнительных модулей.

Существует сервер Protégé, который позволяет работать с удаленной онтологией нескольким пользователям одновременно. Изменения, внесенные в исходную онтологию, отображаются у всех просматривающих ее пользователей. В качестве клиента для данного сервера выступает редактор онтологий Protégé-2000. Для клиент-серверного взаимодействия используется механизм RMI (вызов удаленных процедур).

Современная СППР должна быть распределенной и способной взаимодействовать с гетерогенными системами. Интеграция ИА в состав СППР предоставит высокоуровневый внешний управляющий интерфейс, основанный на стандартных языках общения агентов и онтологиях, способный решить обе задачи.

Для решения этих задач ИА должен быть способен обрабатывать входящие сообщения и отправлять сообщения агентам, как показано на рис. 9.4.

Интерфейс ИА в системе можно использовать для взаимодействия с агентами:

- может осуществляться через установление общих стратегий управления (функций предпочтения) и текущих целей. В процессе такого управлении СППР верхнего уровня может потребоваться

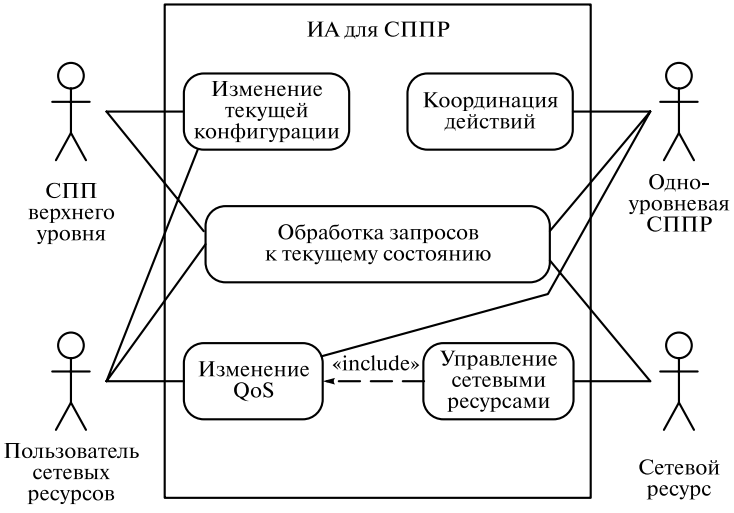


Рис. 9.4. Основные применения агентного интерфейса

информация о текущем состоянии объекта управления данной СППР;

- одноуровневой СППР может потребоваться информация о текущем состоянии соседних СППР для более эффективного использования ресурсов, например для организации маршрутов или для преодоления нештатных ситуаций. Взаимодействия на этом уровне могут использовать весь спектр терминов и обобщенных действий;
- пользователи сетевых ресурсов в общем случае могут запрашивать незначительные изменения конфигурации сети и определенное качество обслуживания в данный момент или в будущем;
- управляемый сетевой ресурс в общем случае может либо самостоятельно запрашивать значения основных управляющих параметров, либо ИА СППР может явно управлять им через заранее определенный набор действий и выполнять мониторинг его состояния.

Существует два возможных подхода для интеграции ИА в СППР.

В соответствии с первым подходом все компоненты СППР реализуются в виде интеллектуальных агентов, работающих на общей агентной платформе. В таком подходе каждый ресурс (рабочая станция, управляемое оборудование или сетевой сервис) будет снабжен ИА, отвечающим за управление ресурсом и мониторинг. Взаимодействие таких агентов с управляемыми ресурсами может происходить по произвольным протоколам, начиная с SNMP и заканчивая выдачей рекомендаций пользователю.

Эти агенты будут являться источниками информации для ИА, выполняющими анализ состояния и предлагающими возможные действия для повышения эффективности работы системы, ее надежности или функционирования в нештатных ситуациях. Построенные ИА планы и некоторые параметры от ИА, выполняющих мониторинг, передаются на АРМ администраторов, которые в данном подходе также являются особым видом агентов, отвечающих за визуализацию текущего состояния и предлагаемого плана действий.

Основное достоинство такой архитектуры — ее открытость. Ресурсы могут быть добавлены и удалены из системы в процессе работы СППР. При этом в систему могут быть добавлены новые агенты для управления новыми типами ресурсов и услуг, понимающие онтологию СППР.

Подход имеет два существенных недостатка: высокие затраты ресурсов на передачу и обработку простейших агентных сообщений и высокую сложность централизованного управления всеми ресурсами.

Первый недостаток связан с необходимостью использования высокоуровневых средств общения при решении даже незначительных задач, таких как подсчет переданных и принятых пакетов. Второй недостаток тесно связан с первым. Из-за неэффективности комму-

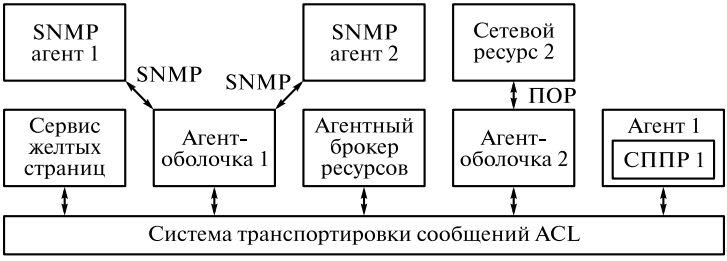


Рис. 9.5. Модель интеграции ИА и традиционного ПО

никаций ИА централизованное управление также становится неэффективным.

Второй подход основан на использовании специализированного агента оболочки, являющегося посредником между МАС и СППР, реализованной по традиционной технологии. Архитектура системы, разработанная на основе спецификации FIPA XC00079B, показана на рис. 9.5.

Агент пользователя сетевых ресурсов может находиться в зоне действия нескольких СППР разного уровня, поэтому ему требуется специальный сервисный агент, который позволит найти имя агента, предоставляющего требуемую услугу.

Таким агентом является агентный брокер ресурсов (Agent Resource Broker), отвечающий за регистрацию предоставляемых услуг и хранение их подробного описания. Описание связывает с услугой набор свойств, определенных в онтологии СППР. Для поиска ресурсного брокера используется специальный сервис желтых страниц (Directory Facilitator).

В соответствии со спецификацией FIPA XC00079B агентом-оболочкой является агент, который способен динамически взаимодействовать с одной или несколькими системами, имеющими уникальные программные описания. Под динамическим взаимодействием понимается, в первую очередь, способность находить новые программные системы по описанию.

Агент-оболочка может использоваться для реализации трех вариантов взаимодействия: для уведомления о наступлении события, для мониторинга за изменением параметра и для формирования управляющих воздействий. Такой подход позволяет агенту-оболочке взаимодействовать с ресурсом, используя расширяемое множество параметров, зависящее только от функциональности, реализуемой ресурсом. В задачи агента-оболочки входит только подстановка параметров, полученных в сообщении на ACL в соответствии с протоколом взаимодействия с ресурсом.

Большинство сетевых ресурсов, работающих под управлением СППР, не реализуют интерфейс ИА, и наиболее эффективным решением будет управление ими напрямую из СППР, не затрачивая

дополнительные ресурсы на построение и разбор агентных сообщений. Доступ к ним из МАС также останется возможным и будет осуществляться через интерфейсы СППР.

Одним из важнейших требований является реализация агентом фиксированной политики безопасности. Спецификацией FIPA PC00089D определяется два класса политик безопасности: политики, управляющие доступом к ресурсу или услуге в данный момент, и политики, управляющие обещанием оказать ту или иную услугу или предоставить доступ к ресурсу в будущем.

Механизм безопасности, предлагаемый FIPA, основан на доменах безопасности, определяющих множество агентов с одинаковыми правами на работу с ресурсами и услугами. Решение FIPA, основанное на выделении множества специализированных агентов (менеджер домена, сервис контроля репутации, библиотеки политик и машина, интерпретирующая политики), не отражает существенных особенностей СППР. Основная задача политик безопасности СППР — обеспечение согласованного использования управляемых ресурсов.

Достоинство предлагаемого подхода к интеграции ИА с СППР — простота реализации даже сложных политик безопасности, реализующих планирование предоставления ресурсов. По сравнению с первым подходом значительно проще осуществляется взаимодействие с СППР: за все сервисы, предоставляемые сетью, отвечает один агент.

Основные недостатки данного подхода — высокая сложность агента по сравнению с агентами в первом подходе, увеличение задержек обработки запросов при возрастании количества взаимодействий с ИА по сравнению с децентрализованным подходом. Устранения второго недостатка можно добиться, выделив наиболее ресурсоемкие диалоги в отдельные ИА. Архитектура СППР показана на рис. 9.6.

По функциональному назначению все модули СППР можно разделить на три подсистемы:

1) подсистему восприятия, содержащую модули, отвечающие за сбор исходных данных от датчиков, SNMP-агентов, CMIP-агентов, и модули, выполняющие взаимодействие с пользователем;

2) подсистему интерпретации, содержащую модули, выполняющие предварительную обработку, преобразование и унификацию данных;

3) подсистему выбора решений, содержащую модули, отвечающие за принятие решения.

Все модули передают данные через БД, хранящую данные в модели представления знаний RDF и использующую модель предметной области, определенную во внутренней онтологии СППР. Для обработки запроса к текущему состоянию ИА достаточно только преобразовать входное сообщение в запрос к базе данных.

Параметр MIB ipInDiscards накапливает значения. При его изменении СППР получает уведомление о новом значении, сохраняемом

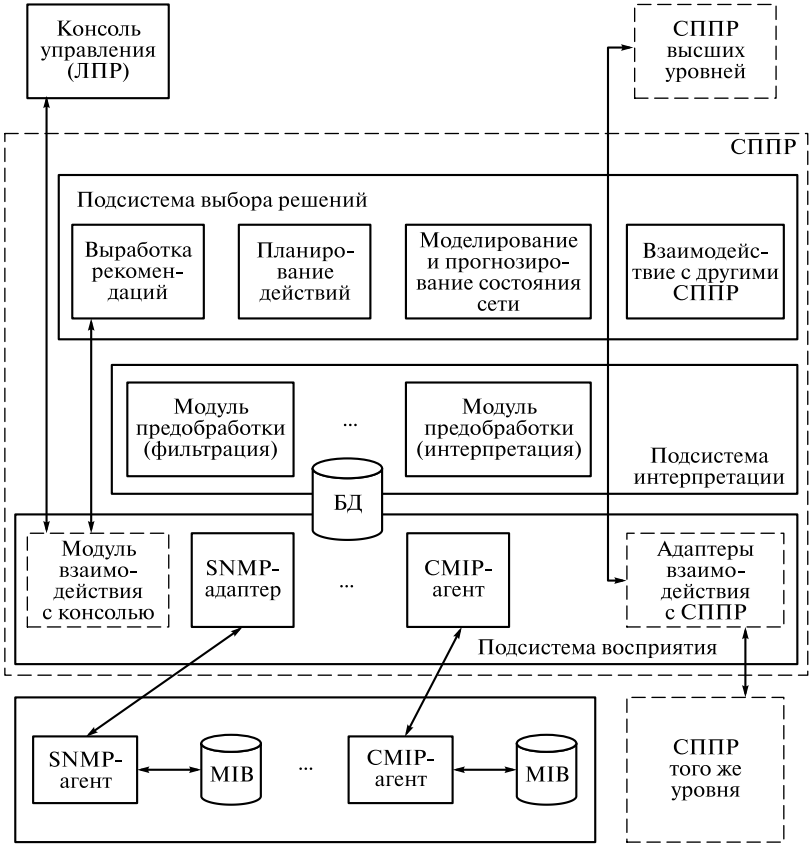


Рис. 9.6. Архитектура СППР

в БД с меткой времени. Потому для обработки запроса необходимо интерпретировать функцию (during "1h") и получить интервал времени, например, с 12:48 до 13:48, запросить в БД все ресурсы, управляемые данной СППР, для которых зарегистрировано событие ipInDiscards, и вычесть из него значение ipInDiscards, зарегистрированное перед 12:48.

Описанную функциональность можно реализовать тремя способами: всю обработку делает сам агент; агент взаимодействует с БД напрямую; агент взаимодействует со специализированным модулем интерпретации.

В соответствии с первым способом агент сможет использовать самое выразительное множество языка SL2. Такая модель имеет два недостатка: необходимость переписывания агента при изменении онтологии и невозможность использовать его аналитические возможности внутри неагентной части системы.

В соответствии со вторым подходом запрос на SL будет напрямую конвертироваться в серию запросов SQL, например, в следующей форме:

```
SELECT object FROM ComponentTable
    WHERE subject = "http://some.url.net/dss.
rdf#DSS17"
          AND property = "http://some.url.net/dss-
ontology.owl#manages";
```

Предположим, в результате выполнения запроса получим hub1,r7, r12,r17,switch32,srv54. Тогда требуемый результат может быть получен следующим запросом:

```
SELECT subject, object FROM IntervalTable
      WHERE (subject = "hub1" OR subject = "r7" OR
subject = "r10" OR
            subject = "r17" OR subject = "switch32"
OR subject = "srv54")
            AND property = "http://some.url.net/dss-
ontology.owl#ipInDiscards"
            AND interval = "1h";
```

При этом предполагается наличие в подсистеме интерпретации некоторого специализированного модуля, который может определять значение изменений счетчиков в заранее заданных интервалах и хранить их в таблице IntervalTable.

Третий подход основан на непосредственном обращении ИА напрямую к модулям подсистемы интерпретации через специализированные интерфейсы. Могут использоваться различные подходы и технологии. Поскольку центральным элементом СППР является БД, в ней может существовать специальная таблица с необработанными запросами и именем модуля интерпретации. Результат обработки может записываться в другую таблицу. Достоинство такого подхода — универсальность и близость к базовой архитектуре системы. Недостаток — все модули интерпретации должны тратить ресурсы на опрос БД даже в то время, когда нет запросов к данному модулю.

Альтернативой данному подходу является написание интерфейсов с фиксированным набором методов. Этот подход потенциально может использоваться с технологиями удаленного вызова методов, такими как RMI, CORBA или COM. В этом случае СППР может быть распределенной по нескольким хостам.

Основное достоинство подхода — эффективное использование ресурсов. В любом случае для третьего подхода необходим специальный язык запросов с достаточно большими выразительными воз-

можностями. Этот язык должен представлять следующие конструкции:

- классы и отношения между понятиями, определяемые именами из онтологии СППР;
- аргументы отношений должны иметь имена, определенные в онтологии СППР;
- должна существовать возможность определять значения некоторых аргументов отношений;
- значениями аргументов могут быть функции, отображающие набор собственных аргументов в некоторый объект предметной области.

В результате вычисления запроса должны определяться комбинации значений для полей, чьи значения не были определены в запросе.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Перечислите этапы стандартной технологии создания СППР.
2. Назовите состав (классы) инструментария.
3. Охарактеризуйте средства разных классов.
4. Назовите классы инструментальных средств экспертных систем.
5. Охарактеризуйте модели создания ПО.
6. Назовите составляющие качества информационных систем.
7. Назовите достоинства и недостатки среды CLIPS.
8. Какими методами оценивают надежность ПО?
9. В чем суть эталонной модели управления агентами?
10. Назовите основные языковые средства онтологий.
11. Каковы основные компоненты языка OWL?
12. В чем состоят особенности управляющего интерфейса в СППР?
13. Укажите основные модули СППР.

Приложение 1

Программы в оболочке GURU

Гораздо большими возможностями обладает пакет GURU. Опишем взаимодействие экспертной системы и встроенной базы данных.

Общая программа состоит из двух частей с расширением *.rss (ЭС) и расширением *.ipf (программа взаимодействия).

```
/*otkad.rss*/

GOAL: POS

INITIAL:
e.lstr = 50
pos = unknown
degree = unknown
qualify = unknown
discov = unknown
grade = unknown
exper = unknown
fam = unknown

fam = ar1 (A,1);
degree = ar1 (A,2);
discover1 (A,3);
grade = ar1 (A,4);
exper = ar1 (A,5);

output ""
input degree with "Введите фамилию принимаемого"

D0:
OUTPUT FAM, POS

RULE: R1
PRIORITY
COST
IF: degree = "НЕТ"
THEN: POS = "ОТКАЗАТЬ"
REASON: ЕСЛИ СТЕПЕНИ НЕТ, ТО РЕШЕНИЕ — ОТКАЗАТЬ

RULE: R2
PRIORITY
```

```
COST
IF: degree = "ДА"
THEN: qualify = "ДА"
REASON: ЕСЛИ СТЕПЕНЬ ЕСТЬ, ТО ЕСТЬ КВАЛИФИКАЦИЯ

RULE: R3
PRIORITY
COST
IF: (qualify = "ДА") AND (discov = "ДА")
THEN: POS = "НАУЧНЫЙ СОТРУДНИК"
REASON: ЕСЛИ КВАЛИФИКАЦИЯ ЕСТЬ И ОТКРЫТИЕ ЕСТЬ, ТО
РЕШЕНИЕ — НАУЧНЫЙ СОТРУДНИК

RULE: R4
PRIORITY
COST
IF: (qualify = "ДА") AND (discov = "НЕТ") AND (grade> =
3,5)
THEN: POS = "ИНЖЕНЕР-КОНСТРУКТОР"
REASON: ЕСЛИ КВАЛИФИКАЦИЯ ЕСТЬ И ОТКРЫТИЯ НЕТ И БАЛЛ НЕ
МЕНЕЕ 3,5, ТО РЕШЕНИЕ — ИНЖЕНЕР-КОНСТРУКТОР

RULE: R5
PRIORITY
COST
IF: (qualify = "ДА") AND (discov = "НЕТ") AND (grade<3,5)
AND (exper>2)
THEN: POS = "ИНЖЕНЕР ПО ЭКСПЛУАТАЦИИ"
REASON: ЕСЛИ КВАЛИФИКАЦИЯ ЕСТЬ И ОТКРЫТИЯ НЕТ И СРЕДНИЙ
БАЛЛ МЕНЕЕ 3,5 И СТАЖ БОЛЕЕ 2 ЛЕТ, ТО РЕШЕНИЕ — ИНЖЕНЕР
ПО ЭКСПЛУАТАЦИИ

RULE: R6
PRIORITY
COST
IF: (qualify = "ДА") AND (discov = "НЕТ") AND (grade<3,5)
AND (exper<2)
THEN: POS = "ОТКАЗАТЬ"
REASON: ЕСЛИ КВАЛИФИКАЦИЯ ЕСТЬ И ОТКРЫТИЯ НЕТ И СРЕДНИЙ
БАЛЛ МЕНЕЕ 3,5 И СТАЖ МЕНЕЕ 2 ЛЕТ, ТО РЕШЕНИЕ — ОТКАЗАТЬ

VAR: FAM

VAR: DEGREE

VAR: QUALIFY

VAR: DISCOV
```

```
VAR: GRADE

VAR: EXPER
END:
```

Взаимодействие БД и ЭС (после компиляции программы ЭС), осуществляется программа с расширением *.ipf.

```
/*swr5.ipf,guru*/
e.lstr = 60
e.stat = false
e.step = false
bet = unknown
agfa = unknown
agfb = unknown
A = unknown
D = unknown
LDA = unknown
LDB = unknown
A1 = unknown
A2 = unknown
I = unknown
FKT = unknown
WAL = unknown
WAR = unknown
WAC = unknown
X = unknown
Z = unknown
F = unknown

/*Осуществляется выбор диалогового режима или БД*/
input bet num with "РАБОТА: С ТАБЛИЦЕЙ – 1, ДИАЛОГ – 0?";
/*При работе в режиме диалога запускается программа
otkad2 и осуществляется работа, описанная в предыдущем
параграфе*/
if bet = 0 then consult "otkad2";
wait;endif
/*При работе с БД первоначально формируется имя
таблицы*/
input WAL with "Введите имя таблицы";
WAM = WAL + ".itb";
WAC = "WAM";
/*Если таблица уже создана, то выводится метка «5»,
поставленная в последнее поле таблицы, и присваивается
переменной X. Символ ^ означает замену */
if bet = 1 and filex(^WAC) = true then
use ^WAL;
```

```
select * from ^WAL;
output "";
WAR = WAL + ".exper";
output label(^WAR);
X = label(^WAR);
endif
/*Если таблица отсутствует, она создается программным
путем с указанием метки в последнем столбце*/
if bet = 1 and filex(^WAC) = false then
output "СОЗДАЙТЕ НОВУЮ ТАБЛИЦУ moj.itb";
wait;
define moj with ^WAC; field fam str 15, field degree str 5;\
field disco str 5;field grade num;field exper num labeled
"5";
enddef
/* Заполнение таблицы и ее вывод на экран для проверки*/
create record for ^WAL;
output " ЗАПОМНИТЕ ЧИСЛО СТРОК И СТОЛБЦОВ ТАБЛИЦЫ
";wait;
select * from ^WAL;
wait;endif
/* Формирование двумерного массива, при этом число
строк задается переменной lasrrec, а число столбцов — с
помощью метки последнего столбца таблицы*/
if bet = 1 then
dim ar1(lastrec(^WAL),^X);
convert from ^WAL to array ar1(1,1);
show ar1;
input LDA with "БУДЕТЕ ПРОСМАТРИВАТЬ ВСЕ СТРОКИ
ТАБЛИЦЫ?";
if LDA = "ДА" then G = 1;A = 1
else input LDB with "УКАЖИТЕ НОМЕРА СТРОК";
Z = "{" + LDB + "}";
G = 0;F = 1;A = VALN(^Z,F);
endif
/*Идет построчная консультация с программой otkad6*/
while A< = #ROW do
D = 1;
while D< = #COL do
ar1(A,D);
D = D + 1;
endwhile
consult "otkad6";
endwhile
/*Начинается процедура объяснения, ее порядок
соответствует процедуре объяснений предыдущего параграфа
с той лишь разницей, что может выдаваться на экран
```

```
несколько строк*/
input FKT with "ОБЪЯСНЕНИЯ НУЖНЫ(ДА/НЕТ)?";
if FKT = "ДА" then
output "КАК ПОЛУЧЕНО РЕШЕНИЕ?";
how #goal;
OUTPUT "ПОЧЕМУ ПРАВИЛО N?";
input A1 with "Введите номер правила";
why A1;
output "ПОЧЕМУ ПЕРЕМЕННАЯ X";
release A1;
input A2 with "Введите имя переменной";
how A2;
output "ЦЕПЬ ПРАВИЛ";
release A2;
I = 1;
while I< = #HCNT do;
why #HOW(I);
?" Нажмите ввод";
wait;
I = I + 1;
endwhile
endif
release FKT;
?" Нажмите ввод";
wait;
if G = 1 then A = A + 1;
endif
if G = 0 and F< = NUMVAL(^Z) then F = F + 1;
A = VALN(^Z,F);
endif
if G = 0 and F>NUMVAL(^Z) then A = #ROW + 1;
endif
endwhile
endif
```

Программа может формироваться через меню GURU или с помощью любого текстового редактора. Программу otkad.rss следует откомпилировать.

Замечание. Одним из входов в ЭС является так называемый естественный язык. С его помощью программу можно запустить через меню.

Иллюстрируем работу на естественном языке с таблицей МАГАЗИН, полями которой являются город, название магазина, имена руководителей.

Через меню GURU на экран вызывается сообщение:

Ваш запрос _____.

Пусть он имеет вид: Show me the stories in Albuquerque (показать магазины в городе Альбукерк). Компьютер выдает результаты запроса.

Вопрос может быть и с ответами «да», «нет». Компьютер может исправить неверно введенное слово. Так, если вместо SALES введено SALEG, то компьютер спрашивает, не является ли слово SALEG словом SALES, и, по-

лучив положительный ответ, работает с верным словом. Так же компьютер реагирует, если вводится синоним. Например, проведен запрос: Which workers make more then 300 (сколько рабочих получает более 300 долларов), а вместо слова workers в словаре присутствует синоним employees.

При обнаружении незнакомого слова компьютер выдает контекстное меню:

Временное изменение
Постоянное определение
Более длинная фраза
Игнорировать слово
Снять запрос

При ответе на второй и третий элементы меню компьютер вводит слово workers в словарь синонимов.

После успешной компиляции возможен процесс консультации. Запустить его можно командой или через меню. В последнем случае на экране появляются соответствующие команды, что ускоряет процесс освоения GURU. После консультации могут потребоваться объяснения. Рассмотрим их разновидности.

Таблица П1.1

```
HOW #GOAL (правило, в котором выводится цель)
       POS
             R6
       ЕСЛИ КВАЛИФИКАЦИЯ ЕСТЬ И ОТКРЫТИЯ НЕТ И БАЛЛ НЕ
МЕНЕЕ 3,5, ТО РЕШЕНИЕ — ИНЖЕНЕР-КОНСТРУКТОР
```

Таблица П1.2

```
WHY R6 (почему правило R6)
         Rule R6(fired) — используется
              (3) DISCOV НЕТ cf 100
              (4) GRADE 4 cf 100
              (6) POS ИНЖЕНЕР-КОНСТРУКТОР cf 100
```

Таблица П1.3

```
HOW "grade"(как использовалась переменная)
              R5          4          cf 100
```

Таблица П1.4

```
LINE OF RESONING"(цепочка правил вывода результата)
          Rule R2(fired) — используется
              (1) DEGREE ДА cf 100
              (2) QUALIFY ДА cf 100
ЕСЛИ СТЕПЕНЬ ЕСТЬ, ТО ЕСТЬ КВАЛИФИКАЦИЯ
          Rule R3(fired) — используется
              (2) QUALIFY ДА cf 100
              (3) DISCOV НЕТ cf 100
```

```
ЕСЛИ КВАЛИФИКАЦИЯ ЕСТЬ, ТО ИМЕЕТЕ ЛИ ВЫ ОТКРЫТИЕ
          Rule R5(fired) — используется
               (2) QUALIFY ДА cf 100
               (3) DISCOV НЕТ cf 100
               (4) GRADE 4 cf 100
ЕСЛИ КВАЛИФИКАЦИЯ ЕСТЬ И ОТКРЫТИЯ НЕТ, ТО ВВЕДИТЕ
СРЕДНИЙ БАЛЛ
          Rule R6(fired) — используется
               (3) DISCOV НЕТ cf 100
               (4) GRADE 4 cf 100
               (6) POS ИНЖЕНЕР-КОНСТРУКТОР cf 100
ЕСЛИ КВАЛИФИКАЦИЯ ЕСТЬ И ОТКРЫТИЯ НЕТ И БАЛЛ НЕ МЕНЕЕ
3,5, ТО РЕШЕНИЕ — ИНЖЕНЕР-КОНСТРУКТОР
```

Отдельно следует обсудить оценку в оболочке GURU достоверности результатов с помощью фактора уверенности и нечетких переменных.

Покажем работу с фактором уверенности на примерах.

Пример П1.1. Пусть имеются предикаты

```
число_мес = 09.03      cf 40
год = 2005             cf 60
```

Необходимо определить достоверность предиката

```
a1 = год + число_мес   cf 100,
```

т.е. операции конкатенации.

В оболочке GURU для факторов уверенности имеются три настройки внешней среды:

e.cfjo — для арифметических операций, конкатенации и логической операции И;

e.cfco — для логических операций типа ИЛИ

e.cfva — представляет собой объединение факторов уверенности как операции И, так и операции ИЛИ.

Для e.cfjo возможны следующие правила (табл. П2.5).

Таблица П1.5. **Настройка e.cfjo**

Алгебра	Алгоритм	Результат 1	Результат 2
1. Минимум (m)	$\min(cf1, cf2)$	40	40
2. Произведение (p)	$(cf1 * cf2)/100$	24	19
3. Среднее (a)	$\{\min(cf1, cf2) + (cf1 * cf2)/100\}/2$	32	30
4. Баланс (b)	$\frac{cf1 \cdot cf2}{100} \cdot \left(2 - \frac{\max(cf1, cf2)}{100}\right)$	34	33

Для примера П1.1 результат помещен в столбце Результат 1.

Пример П1.2. Изменим результирующий предикат:

```
число_мес = 09.03      cf 40,
год = 2005             cf 60.
a1 = год + число_мес   cf 80.
```

Достоверность этого примера приведена в столбце Результат 2.

Данная настройка непригодна для логических операций ИЛИ. Если взять две операции И и ИЛИ, то результаты должны быть разными. Однако данная настройка дает одинаковые результаты.

Чтобы результаты были разными для случая $A \cap B$, используют настройку e.cfjo, для случая $A \cup B$ — настройку e.cfco.

В последней вместо вычисления min используют max, вместо произведения — вероятностную сумму:

$$cf = \frac{\{cf1 \cdot cf2 + (1 - cf1) \cdot cf2 + cf1(1 - cf2)\}}{100}.$$

Далее среднее от двух первых правил и баланс.

Сравнительные числовые данные выполним для такого примера.

Пример П1.3.

```
A cf 40,

B cf 80.
```

Результат сведем в табл. П1.6.

Таблица П1.6. **Сравнение правил**

Операция	Настройка	*m*	*p*	*a*	*B*
A ∩ B	e.cfjo	40	32	36	38
A ∪ B	e.cfco	80	88	84	82

Настройка среды e.cfva характеризуется двумя параметрами (e.cfjo и e.cfco соответственно) и насчитывает 16 алгебр.

Нечеткие переменные в GURU записываются следующим образом:

$$X = \{1,\ 2\ cf\ 40,\ 3\ cf\ 80\}.$$

Если после значения переменной ничего не указывается, фактор уверенности равен 100. GURU позволяет иметь до 255 значений нечеткой переменной. Для нечетких переменных устанавливаются две основных операции (сложение и вычитание):

1) сложение:

$$X = \{1\ cf\ 50,\ 2\ cf\ 50\};$$

$$X = + \{1\ cf\ 25\}.$$

Результат: $X = \{1\ cf\ 75,\ 2\ cf\ 50\}$;
2) вычитание:

$$X = \{1\ cf\ 50,\ 2\ cf\ 50\};$$

$$X = -\{1\ cf\ 25\}.$$

Результат: $X = \{1\ cf\ 25,\ 2\ cf\ 50\}$.
Подобные операции могут быть записаны и следующим образом:

$$1.\ X = \{1,\ 2\ cf\ 50,\ 3\}.$$

$$Y = X.$$

$$Y = \{1,\ 2\ cf\ 50,\ 3\}.$$

$$2.\ X = \{1,\ 2\ cf\ 50,\ 3\}.$$

$$Y = X\ cf\ 50.$$

$$Y = \{1\ cf\ 50,\ 2\ cf\ 25,\ 3\ cf\ 50\}.$$

Обычно нечеткие переменные используют 10 — 20 значений.

Программы на языке ПРОЛОГ

Здесь возможна реализация на правилах и логике. Рассмотрим реализацию на правилах.

Данные можно вводить в диалоге и через базу данных. Диалог используется все реже в силу резкого снижения производительности программного средства, поэтому рассмотрим второй вариант ввода.

Покажем программу с внутренней базой данных, представленной предикатами spis.

```
/*recepb.pro*/

predicates
spis(symbol, symbol, symbol, real, integer)

goal
          resh.

clauses

spis("Иванов", "да", "нет", 4,3).
spis("Петров", "нет", "да", 4,3).

    resh:-
    spis(F, _, _, _, _),
    A = "нет",
    write(F, "в приеме отказать"),
    !,fail.

    resh:-
    spis(F, A, B, _, _),
    A = "да",
    B = "да",
    write(F, " принять научным сотрудником"),
    !,fail.

    resh:-
    spis(F, A, B, C, _),
    A = "да",
    B = "нет",
    C>3.5,
    write(F, "принять инженером-конструктором"),
    !,fail.

    resh:-
```

```
    spis(F, A, B, C, D),
    A = "да",
    B = "нет",
    C<3.5,
    D> = 2,
    write(F, " принять инженером по эксплуатации"),
    !,fail.

    resh:-
    spis(F, A, B, C, D),
    A = "да",
    B = "нет",
    C<3.5,
    D<2,
    write(F, "в приеме отказать"),
    !,fail.
```

Приведем пример программы с внешней базой данных. Заметим, что все программы имеют внутреннюю цель. База данных содержится в файле inf. pro и имеет вид

```
spis("Иванов", "да", "нет", 4,3)
spis("Петров", "нет", "да", 4,3)

/*recepc.pro*/

predicates
sp(symbol, symbol, symbol, real, integer)

database
spis(symbol, symbol, symbol, real, integer)

goal
          resh.

clauses

    sp(F, A, B, C, D):-
    consult("inf.pro"),
    spis(F, A, B, C, D),
    !.

    resh:-
    sp(F, _, _, _, _),
    A = "нет",
    write(F, "в приеме отказать"),
    !,fail.

    resh:-
```

```
sp(F, A, B, _, _),
A = "да",
B = "да",
write(F, " принять научным сотрудником"),
!,fail.

resh:-
sp(F, A, B, C, _),
A = "да",
B = "нет",
C>3.5,
write(F, "принять инженером-конструктором"),
!,fail.

resh:-
sp(F, A, B, C, D),
A = "да",
B = "нет",
C<3.5,
D> = 2,
write(F, " принять инженером по эксплуатации"),
!,fail.

resh:-
sp(F, A, B, C, D),
A = "да",
B = "нет",
C<3.5,
D<2,
write(F, "в приеме отказать"),
!,fail.
```

Нетрудно видеть, что «база данных» является понятием условным, поскольку состоит из одной таблицы. Обратим внимание, что в примерах только получался результат и далее в информационном плане он нигде не использовался.

До сих пор ничего не говорилось о процедуре объяснений. Дело в том, что здесь возникает потребность изменений правил, т.е. вмешательства не только в секцию clause, но и в секцию predicates, а иногда — и в секцию domains. В связи с этим рассмотрим использование оболочки BASHELL, реализованной на языке ПРОЛОГ. Процедура ее создания и особенно отладки чрезвычайно трудоемка.

Оболочка работает следующим образом. Цель программы-оболочки предусмотрена внутри, поэтому по команде «run» программа автоматически запускается и на экране появляется меню (рис. П1.1).

Первоначально необходимо в окне РЕДАКТОР раскрыть содержание работы.

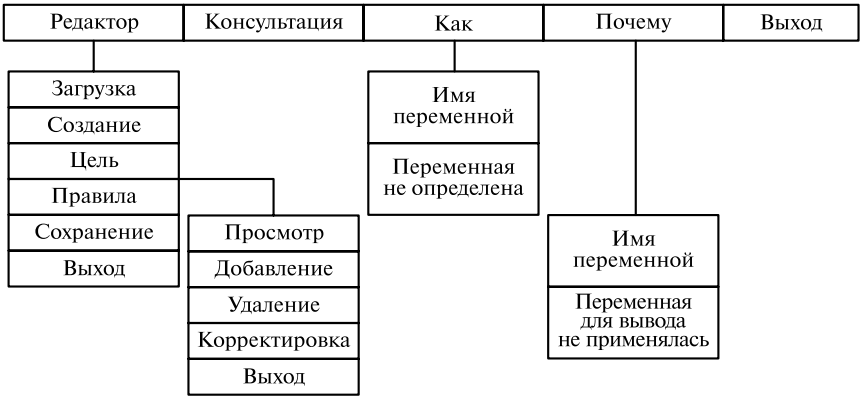


Рис. П2.1. Интерфейс пользователя (оконная система) оболочки BASHELL

Если база правил уже создана, соответствующий файл следует загрузить в элемент меню Загрузка.

Если база правил отсутствует, ее следует построить, перейдя в элемент меню Создание.

Используем следующие обозначения: rule — правило; quest — вопрос; proc — процедура; eq — равно; ne — не равно; gt — больше; lt — меньше; le — меньше или равно; ge — больше или равно.

База правил может быть создана в любом текстовом редакторе. Рассмотрим систему правил (табл. П2.1). В предикате «rule» указаны номер правила, его тип, список условий и цель.

Таблица П2.1. **Система правил**

```
/*ch.pro*/

rule(1, quest, [], input("Введите A (1 — "есть", 0 —
"нет"), "A")
rule(2, proc, [eq("A","0")], [eq("X", "отказать")])
rule(3, quest, [eq("A","1")], input("Введите B (1 —
"lf", 0 — "нет"), "B")
rule(4, proc, [eq("B","1")], [eq("X", "научный
сотрудник")])
rule(5, quest, [eq("B","0")], input("Введите C, балл",
"C")
rule(6, proc, [gt("C","3.5")], [eq("X", "инженер-
конструктор")])
rule(7, quest, [le("C","3.5"], input("Введите D, лет"),
"D")
rule(8, proc, [ge("D","2")], [eq("X", "инженер по
эксплуатации")])
rule(9, proc, [lt("D","2")], [eq("X", " отказать")])
```

Как видно, форма представления правил здесь своеобразна и отличается от предыдущего варианта. Такая форма позволяет при модификации правил не вмешиваться в программу, однако менее наглядна, что затрудняет контроль правильности реализации алгоритма.

После загрузки базы правил можно выполнить консультацию (элемент меню КОНСУЛЬТАЦИЯ) и задать вопросы компьютеру (элементы меню КАК и ПОЧЕМУ).

Приведем примеры сообщений-ответов на указанные вопросы, полагая, что при консультации имело место A = 1, B = 1, получен ответ "X" = "научный сотрудник" и сработали правила 1, 3, 4.

На вопрос КАК переменной X получен ответ:

X = научный сотрудник выведено в правиле 4 из

A = 1, B = 1

B = 1 введено

A = 1 введено.

На вопрос ПОЧЕМУ после окончания консультации для переменной B будет получен ответ:

A используется для вывода B в правиле 3

B используется для вывода X в правиле 4.

Правила полезно сопровождать комментариями для пользователя.

Программа базы знаний «Прием на работу» на основе СУБД InterBase

Программа экспертной системы реального времени (базы знаний) «Прием на работу» на основе СУБД InterBase в среде Delphi достаточно объемна, поэтому опишем основные положения ее формирования[1]. Полный текст программы приведен в [55].

При программировании с использованием базы данных выделяют такие составляющие:

1) собственно БД – система таблиц;

2) интерфейс пользователя;

3) алгоритм приложения.

Собственно БД. В структуре БД выделяются три основные таблицы (Kadry, Rule, Stat) и три дополнительные таблицы (Kadry_ish, Rule_ish, Stat_ish), предназначенные для восстановления данных при начале новой сессии. Все таблицы строятся по одному шаблону, поэтому приведем программу только для таблицы Rule.

```
/* Table: RULE, Owner: SYSDBA */
CREATE TABLE RULE (NBR_RULE INTEGER,
   STEPEN CHAR(5),
   OTKRITIE CHAR(5),
   ZNAK_BAL CHAR(5),
   AVG_BAL NUMERIC(15, 2),
   ZNAK_STAZ CHAR(5),
   STAG INTEGER,
   DOLGNOST CHAR(20),
   INFORM CHAR(255),
   UNIQUE(NBR_RULE);
```

Связи между таблицами в данной БД отсутствуют.

Производится заполнение таблиц данными.

Интерфейс пользователя. Пользователь будет взаимодействовать с построенной системой таблиц (собственно БД) через меню, которое следует установить на главной форме Form1 среды Delphi. Эта форма открывается при запуске приложения (проекта).

Используем 15 форм Delphi, а на главной форме Form1 оставим меню и компоненты Edit, указывающие текущий цикл и количество циклов.

Формы, кроме первой, главной, лучше сделать модальными, что позволит пользователю не забывать закрывать ненужные таблицы и избежать открытия одной и той же таблицы в разных формах. Одновременно в интерфейсе можно избавиться от избытка кнопок закрытия вызываемых на экран таблиц.

---

[1] См.: *Чертовской В.Д.* Базы данных: современный подход. — СПб. : Петерб. ин-т печати, 2003. — 212 с.

Применим следующий вид меню.

Главная	*Показать*	*Обработать*
Открыть	Штатное расписание	Начать
Завершить	Кадры	Запуск правил
Выход	Претенденты	Количественный результат
	Работающие	Объяснения
	Принимаемые	Корректировка правил
	Принятые	Корректировка результата
	Уволившиеся	Принять всех
		Продолжить

Примем следующее закрепление таблиц:

Form1 — Главная форма;
Form2 — Штатное расписание;
Form3 — Кадры;
Form4 — Работающие;
Form5 — Претенденты;
Form6 — Принимаемые;
Form7 — Количественный результат;
Form8 — Объяснения;
Form9 — Принимаемые, Правила;
Form10 — Принимаемые;
Form11 — Принятые;
Form12 — Штатное расписание-результат;
Form13 — Кадры-результат;
Form14 — Уволившиеся;
Form15 — Работающие + Принятые.

Алгоритм приложения. Рассмотрим основные типовые составляющие алгоритма приложения. Для каждого элемента меню выделяются серверная (хранимые процедуры) и клиентская составляющие алгоритма приложения.

1. Начать.

Серверная часть

```
CREATE PROCEDURE NACHAT
AS
BEGIN
DELETE FROM KADRY;
INSERT INTO KADRY
  SELECT * FROM KADRY_ISH;
END
```

Клиентская часть

```
procedure TForm1.N4Click(Sender: TObject);
{Начать}
begin
StoredProc1.UnPrepare;
{свойства DatabaseMame – priem_ss, StoredProcName - NACHAT}
StoredProc1.Prepare;
StoredProc1.ExecProc;
N2.Enabled:=False;//ОБРАБОТАТЬ
N3.Enabled:=True;//ПОКАЗАТЬ
N4.Enabled:=False;//Начать
N16.Enabled:=True;//Штатное расписание
end;
```

II. Запуск правил.

Серверная часть

```
CREATE PROCEDURE PRP_ZAPRULE (IN_VREM INTEGER)
AS
DECLARE VARIABLE DSTEPEN CHAR(10);
DECLARE VARIABLE DDOLGNOST CHAR(20);
DECLARE VARIABLE DOTKRITIE CHAR(5);
DECLARE VARIABLE DSTAG INTEGER;
DECLARE VARIABLE DAVG_BAL NUMERIC(15, 2);
DECLARE VARIABLE OLD_STATUS CHAR(5);
BEGIN
/* Pravilo1*/
    FOR SELECT
     STEPEN,
     DOLGNOST
      FROM RULE
        WHERE NBR_RULE=1
  INTO:DSTEPEN, :DDOLGNOST
    DO
     BEGIN
      UPDATE KADRY
        SET NBR_RULE=1,
            DOLGNOST=:DDOLGNOST
             WHERE STATUS='прете'
               AND STEPEN=:DSTEPEN
               AND VREM=:IN_VREM;
            END
END
```

Клиентская часть.

```
procedure TForm1.N6Click(Sender: TObject);
```

```
{Запуск правил}
begin
StoredProc2.UnPrepare;
StoredProc2.ParamByName('IN_VREM').Value:=StrToInt(Edit1.
Text);
StoredProc2.Prepare;
StoredProc2.ExecProc;
N2.Enabled:=False;//ОБРАБОТАТЬ
N3.Enabled:=True;//ПОКАЗАТЬ
N22.Enabled:=True;//Принимаемые
N6.Enabled:=False;//Запуск правил
end;
```

III. Количественный результат.

Серверная часть.

```
CREATE PROCEDURE PRO_KOLRES (IN_VREM INTEGER)
RETURNS (OUT_VREM INTEGER,
OUT_DOLGN CHAR(20),
OUT_PLAN_R INTEGER,
OUT_FACT_R INTEGER,
OUT_VAKANS INTEGER,
OUT_PRINIM INTEGER,
OUT_NEDOBOR INTEGER)
AS
   BEGIN
   FOR SELECT
      VREM,
   DOLGN,
   PLAN_R,
   FACT_R,
   VAKANS,
   PRINIM,
   NEDOBOR
      FROM STAT
   WHERE VREM = :IN_VREM
   INTO :OUT_VREM,
   :OUT_DOLGN,
   :OUT_PLAN_R,
   :OUT_FACT_R,
   :OUT_VAKANS,
   :OUT_PRINIM,
   :OUT_NEDOBOR
   DO
   BEGIN
   SELECT COUNT(*)
      FROM KADRY
```

```
      WHERE DOLGNOST =:OUT_DOLGN
      AND VREM=:IN_VREM
      AND STATUS='прним'
      INTO :OUT_PRINIM;
      OUT_NEDOBOR=:OUT_VAKANS -:OUT_PRINIM;
      UPDATE STAT
       SET PRINIM=:OUT_PRINIM,
           NEDOBOR=:OUT_NEDOBOR
             WHERE VREM = :IN_VREM
             AND DOLGN=:OUT_DOLGN;
      SUSPEND;
      END
END
```

Клиентская часть.

```
procedure TForm1.N8Click(Sender: TObject);
{Количественный результат}
begin
Form7.ShowModal;
if ib1=0 then
begin
if MessageDlg('Объяснения нужны?',mtInformation, [mbYes,
mbNo],0)=mrYes then
begin
N9.Enabled:=True;//Объяснение
N8.Enabled:=False//Количественный результат
end
else
begin
if ie1>=0 then
begin
N8.Enabled:=False;//Количественный результат
N13.Enabled:=True;//Принять всех
end;
if ie1<0 then
begin
ShowMessage('Число принимаемых больше количества
вакансий. Измените правила или результаты');
N8.Enabled:=False;//Количественный результат
N10.Enabled:=True;//Изменение правил
N11.Enabled:=True;//Изменение результата
end;
end;
end;
if ib1=1 then
begin
if ie1>=0 then
```

```
begin
N8.Enabled:=False;//Количественный результат
N13.Enabled:=True;//Принять всех
end;
if ie1<0 then
begin
ShowMessage('Число принимаемых больше количества
вакансий. Измените правила или результаты');
N8.Enabled:=False;//Количественный результат
N10.Enabled:=True;//Изменение правил
N11.Enabled:=True;//Изменение результата
end;
end;
end;

procedure TForm7.FormActivate(Sender: TObject);
{Количественный результат}
begin
with Query1 do
begin
Close;
{SQL-свойство select * pro_kolres(:param1)}
ParamByName('param1').Value:=StrToInt(Form1.Edit1.Text);
Open;
end;
with Query2 do
begin
Close;
Open;
end;
ie1:=Query2.FieldByName('out_min').Value;
end;
```

СПИСОК ЛИТЕРАТУРЫ

1. *Алексеев П. В.* Теория познания и диалектика / П. В. Алексеев, А. В. Панин. — М. : Высшая школа, 1991. — 383 с.

2. *Альсевич В. В.* Оптимизация линейных экономических моделей. Статические задачи / В. В. Альсевич, Р. Габасов, В. С. Глушенков. — Мн. : БГУ, 2000. — 210 с.

3. *Андрейчиков А. В.* Интеллектуальные информационные системы / А. В. Андрейчиков, О. Н. Андрейчикова. — М. : Финансы и статистика, 2004. — 256 с.

4. *Ашиенц Р. А.* Логические методы в искусственном интеллекте / Р. А. Ашиенц. — М. : МГАПИ, 2001. — 224 с.

5. Базы данных. Интеллектуальная обработка данных / В. В. Корнеев [и др.]. — М. : Нолидж, 2000. — 352 с.

6. *Балобанов В. С.* Искусственный интеллект и искусственное сознание. Тенденция развития и перспективы реализации / В. С. Балобанов, А. В. Серегин // Информация и космос — 2004. — № 3. — С. 105 — 112.

7. *Балобанов В. С.* Как определять функции, не указывая их области определения? / В. С. Балобанов // Современная логика : проблемы теории, истории и применения в науке: тез. докладов VII Общероссийской научной конференции, г. Санкт-Петербург 20 — 22 июня 2002 г. — СПб., 2002. — С. 131 — 132.

8. *Балобанов В. С.* Логика функций с неопределенными аргументами: дис. на соиск. уч. степени канд. философ. наук: 09.00.13: защищена 24.03.91: утв. 15.09.91 / Балобанов Виталий Серафимович. — Л. : 1991. — 220 с.

9. *Барский А. Б.* Нейронные сети : распознавание, управление, принятие решений / А. Б. Барский. — М. : Финансы и статистика, 2004. — 176 с.

10. *Башмаков А. И.* Интеллектуальные информационные технологии / А. И. Башмаков, И. А. Башмаков. — М. : Изд-во МГТУ им. Н. Э. Баумана, 2005. — 304 с.

11. *Братко И.* Программирование на языке Prolog для искусственного интеллекта / И. М. Братко: — Мир, 1990. — 420 с.

12. *Брокгауз Ф. А.* Энциклопедический словарь. В 86 т. / Ф. А. Брокгауз, И. А. Ефрон. — СПб. : Полрадис, 1993 — 1998.

13. *Гаврилова Т. А.* Базы знаний интеллектуальных систем : учеб. пособие / Т. А. Гаврилова, В. Ф. Хорошевский. — СПб. : Питер, 2000. — 384 с.

14. *Герман О. В.* Введение в теорию экспертных систем и обработку знаний / О. В Герман. — Минск. : ДизайнПРО, 1995. — 255 с.

15. *Гладков Л. А.* Генетические алгоритмы / Л. А. Гладков, В. В. Курейчик, В. М. Курейчик. — М. : Физматлит, 2006. — 320 с.

16. *Городецкий В. И.* Информационные технологии и многоагентные системы / В. И. Городецкий // Проблемы информатизации. — 1998. — Вып. 1. — С. 3 — 14.

17. *Девятков В. В.* Системы искусственного интеллекта / В. В. Девятков. — М. : МГТУ, 2001. — 352 с.

18. *Джексон П.* Введение в экспертные системы / П. Джексон. — М. : Вильямс, 2001.— 624 с.

19. *Жданов А. А.* Автономный искусственный интеллект / А. А. Жданов. — М. : Бином, 2005. — 359 с.

20. *Ин Ц.* Использование Турбо-Пролога / Ц. Ин, Д. Соломон. — М. : Мир, 1993. — 608 с.

21. Искусственный интеллект : справочник. В 3 кн. Кн. 1. Системы общения и экспертные системы / под ред. Э. В. Попова. — М. : Радио и связь, 1990. — 464 с. ; Кн. 2. Модели и методы / под ред. Д. А. Поспелова — М. : Радио и связь, 1990. — 304 с. ; Кн. 3. Программные и аппаратные средства / под ред. В. Н. Захарова, В. Ф. Хорошевского. — М. : Радио и связь, 1990. — 368 с.

22. *Кандрашина Е. Ю.* Представления знаний о времени и пространстве в интеллектуальных системах / Е. Ю. Кандрашина, Л. В. Литвинцева, Д. А. Поспелов. — М. : Наука, 1989. — 328 с.

23. *Карпенко А. С.* Паранепротиворечивая структура внутри многозначной логики / А. С. Карпенко // Многозначные, релевантные и паранепротиворечивые логики / Труды научно.-исслед. семинара по логике Института философии АН СССР. — М. : ИФАН, 1984. — С. 34 — 39.

24. *Карминский А. М.* Информационные системы в экономике: В 2 ч. Ч. 1. Методология создания / А. М. Карминский, Б. В. Черников. — М. : Финансы и статистика, 2006. — 336 с.

25. *Комашинский В. И.* Нейронные сети и их применение в системах управления и связи / В. И. Комашинский, Д. А. Смирнов. — М. : Горячая линия-Телеком, 2002. — 94 с.

26. *Кричевский М. Л.* Интеллектуальные методы в менеджменте / М. Л. Кричевский. — СПб. : Питер, 2005. — 304 с.

27. *Леоненков А.* Нечеткое моделирование в среде MatLab и fuzzyТЕСН / А. Леоненков. — СПб. : БХВ, 2003. — 736 с.

28. *Лимантов С. Ф.* Лекции по логике вопросов / С. Ф. Лимантов. — Л. : ЛГПИ, 1975. — 112 с.

29. *Лисс А. А.* Искусственные нейронные сети / А. А. Лисс. — СПб. : Изд-во СПбГЭТУ «ЛЭТИ», 2003. — 96 с.

30. *Ломазова И. А.* Моделирование многоагентных сетей вложенными сетями Петри / И. А. Ломазова // Изв. РАН. Теория и системы управления, — 2000. — № 6. — С. 965 — 974.

31. *Лорьер Ж. Л.* Системы искусственного интеллекта / Ж. Лорьер. — М. : Мир, 1991. — 568 с.

32. *Люгер Д. Ф.* Искусственный интеллект : стратегии и методы решения сложных проблем / Д. Ф. Люггер. — М. : Издательский дом «Вильямс», 2003. — 864 с.

33. *Мелихов А. Н.* Ситуационные советующие системы с нечеткой логикой / А. Н. Мелихов, Л. С. Бернштейн, С. Я. Коровин. — М. : Наука, 1990. — 272 с.

34. Методы и средства построения интеллектуальных агентов реального времени / В. В. Денисов [и др.]. // Труды 7-й нац. конф. по искусственному интеллекту КИИ, 2000. Т. 1—2. — М. : Физматлит, 2000. — С. 805—813.

35. Нейронные сети. STATISTICA Neural Networks. Методология и технология современного анализа данных / пер. с англ. — М. : Горячая линия — Телеком, 2008. — 392 с.

36. Нейросетевые системы управления / В. А. Терехов [и др]. — СПб. : СПбГУ, 1999. — 265 с.

37. *Острейковский В. А.* Информатика / В. А. Острейковский. — М. : Высшая школа, 2004. — 511 с.

38. *Панченко Т. В.* Генетические алгоритмы / Т. В. Панченко — Астрахань : Астраханский университет, 2007. — 87 с.

39. Перспективы развития вычислительной техники : В 11 кн. / под ред. Ю. М. Смирнова. Кн. 1: Информационные семантические системы / Н. М. Соломатин [и др.]. — М. : Высшая школа, 1989. — 127 с.

40. *Першиков В. И.* Толковый словарь по информатике / В. И. Першиков, В. М. Савинков. — М. : Финансы м статистика, 1991. — 543 с.

41. *Полещук О. М.* Методы и модели обработки нечеткой информации / О. М. Полещук. — М. : Энергоатомиздат, 2007. — 287 с.

42. *Поспелов Г. С.* Искусственный интеллект — основа новой информационной технологии / Г. С. Поспелов. — М. : Наука, 1988. — 280 с.

43. *Поспелов Д. А.* От коллектива автоматов к мультиагентным системам / Д. А. Поспелов // Proc. of the International Workshop «Distributed Artificial Intelligence and Multi-Agent Systems» DAIMAS`97. — June 15—18, 1997. St. Peterburg, Russia. — P. 319—325.

44. *Пупков К. А.* Интеллектуальные системы / К. А. Пупков, В. Г. Коньков. — М. : Издательство МГТУ им. Н. Э. Баумана, 2003. — 348 с.

45. *Романов В. П.* Интеллектуальные информационные системы в экономике / В. П. Романов. — М. : Экзамен, 2003. — 496 с.

46. *Роганов В. Р.* Методы искусственного интеллекта для машинного перевода текстов / В. Р. Роганов, С. М. Роганова, М. Е. Новосельцева. — Пенза : Изд-во Пенз. гос. ун-та, 2007. — 127 с.

47. *Рутковская Д.* Нейронные сети, генетические алгоритмы и нечеткие системы / Д. Рутковская, М. Пилиньский, Л. Рутковский. — М. : Горячая линия, 2006. — 452 с.

48. *Смирнова Г. Н.* Проектирование экономических информационных систем / Г. Н. Смирнова, А. А. Сорокин, Ю. Ф. Тельнов. — М. : Финансы и статистика, 2003. — 512 с.

49. *Смолин Д. В.* Введение в искусственный интеллект / Д. В. Смолин. — М. : Физматлит, 2004. — 356 с.

50. Статические и динамические экспертные системы / Э. В. Попов [и др.]. — М. : Радио и связь, 1996. — 352 с.

51. *Стефанюк В. Л.* Поведение многоагентных систем : парадигма координации / В. Л. Стефанюк. // Новости искусственного интеллекта. — 1997. — № 4. — С. 92—104.

52. *Тарасов В. Б.* От многоагентных систем к интеллектуальным организациям : философия, психология, информатика / В. Б. Тарасов. — М. : Эдиториал УРСС, 2002. — 352 с.

53. *Чень Ч.* Математическая логика и автоматическое доказательство теорем / Ч. Чень, Р. Ли. — М. : Наука, 1983. — 360 с.

54. *Чертовской В. Д.* Основы построения автоматизированных систем управления. Ч. 2 / В. Д. Чертовской.— Минск : МРТИ, 1974. — 168 с.

55. *Чертовской В. Д.* Теоретические основы автоматизированного управления : процедурное представление / В. Д. Чертовской. — М. : МГУП, 2004. — 218 с.

56. *Швецов А. Н.* Распределенные интеллектуальные информационные системы / А. Н. Швецов, С. А. Яковлев. — СПб. : СПбГЭТУ «ЛЭТИ», 2003. — 318 с.

57. *Шеховцов О. И.* Интеллектуальные средства поддержки принятия управленческих решений / О. И. Шеховцов, В. Д. Чертовской, Б. М. Шифрин. — СПб. : СПбГЭТУ «ЛЭТИ», 2000. — 59 с.

58. Экспертные системы для персональных компьютеров / В. С. Крисевич [и др.]. — Мн. : Вышэйшая школа. 1990. — 197 с.

59. *Ясницкий Л. Н.* Интеллектуальные информационные технологии и системы / Л. Н. Ясницкий. — Пермь : ПГУ, 2007. — 270 с.

60. *Artale A.* Part-Whole Relations in Object-Centered Systems : An Overview / A. Artale [ed] // Data & Knowledge Engineering, North-Holland, Elsevier, 1996. V. 20. — P. 347 — 383.

61. *Artale A. A.* Temporal Description Logic for Reasoning about Action and Plans / A. Artale, E. Frnconi // Journal of artificial Intelligence Research, issue 9, 1998. — P. 463 — 506.

62. *Belgrave M.* The Unified Agent Architecture : A White Paper / M. Belgrave. [Электронный ресурс] // http://www.ee.mcgill.cu/~belmarc.

63. *Belnap B.* The logic of questions and answers / D. Belnap, T Steel. — New Haven and London, 1976. — 287 p.

64. *Donini F.* Reasoning in Description Logics / F. Donini [ed] // Principles of Knowledge Representation and Reasoning, edited by G. Brewka; Studies in Logic, Language and Information, CLSI Publications, 1996. — P. 193 — 238.

65. *Durfee E. H.* Negotiating task decomposition and allocation using partial global planning / E. H. Durfee, V. Lesser. // Distributed Artificial Intelligence : Vol. II. M. : Gasser, L. and Hunhs, ed. San Francisco : Morgan Kaufmann, 1999. — P. 229 — 244.

66. FIPA Abstract Architecture Specification [Электронный ресурс] — http://www.fipa.org/specs/fipa 00001/XC00001J.html.

67. *Horrocks, I.* Ontology reasoning in the SHOQ(D) description logic / I. Horrocks, U. Sattler // In B. Nebel, editor, Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001). Morgan Kaufmann, 2001. — P. 199 — 204.

68. *Lukasiewicz J.* Aristotle's syllogistic from the standpoint of modern formal logic / J. Lukasiewicz. — Oxford, 1957. — 222 p.

69. *McGuinness D. L.* An Industrial-Strength Description Logic-Based Configurator Platfom / D. L. McGuinness, J. R. Wright. //IEEE Intelligent systems, Issue 4, 1998. — P. 69 — 77.

70. *Nwana H. S.* Software Agents : An Overview / H. S. Nwana. // Knowledge Engineering Review. — 1996. — Vol. 11, № 3. — P. 1 — 40.

71. *Valente A.* Building and (Re)Using an ontology of Air Campaign Planning / A. Valente [ed] // IEEE Intelligent systems, Issue 1, 1999. — P.27 — 36.

72. *Veloso M.* CNITED-98 RoboCup-98 small robot world champion team / M. Veloso [и др.]. // AI Magazine, 21(1), 2000. — С. 29 — 36.

73. *Wooldridge M. J.* Agent Theories, Architectures and Languages : A Survey / M. J. Wooldridge, N. R. Jennings. // Intelligent Agents. ECAI-94 Workshop on Agent Theories, Architecture and Languages, Amsterdam, The Netherlands, August 8 — 9, P. 3 — 39.

74. *Wooldridge M.J.* Intelligent Agents : Theory and Practice / M. J. Wooldridge, N. R. Jennings // Knowledge Engineering Review. — 1995. — № 10 (2). — P. 115 — 152.

ОГЛАВЛЕНИЕ

Учебное издание

Советов Борис Яковлевич,
Цехановский Владислав Владимирович,
Чертовской Владимир Дмитриевич

Интеллектуальные системы и технологии

Учебник

Редактор *М. М. Новикова*
Технический редактор *Е. Ф. Коржуева*
Компьютерная верстка: *Р. Ю. Волкова*
Корректоры *А. П. Сизова*, *И. А. Ермакова*

Изд. № 101116218. Подписано в печать 28.12.2012. Формат 60×90/16.
Гарнитура «Ньютон». Печать офсетная. Бумага офсетная № 1. Усл. печ. л. 20,0.
Тираж 1 200 экз. Заказ №

ООО «Издательский центр «Академия». www.academia-moscow.ru
129085, Москва, пр-т Мира, 101В, стр. 1.
Тел./факс: (495) 648-0507, 616-00-29.
Санитарно-эпидемиологическое заключение № РОСС RU. АЕ51. Н 16067 от 06.03.2012.

Отпечатано с электронных носителей издательства.
ОАО «Тверской полиграфический комбинат», 170024, г. Тверь, пр-т Ленина, 5.
Телефон: (4822) 44-52-03, 44-50-345. Телефон/факс: (4822) 44-42-15.
Home page — www.tverpk.ru Электронная почта (E-mail) — sales@tverpk.ru

ACADEMIA

Издательский центр «Академия»

Учебная литература для профессионального образования

Наши книги можно приобрести (оптом и в розницу)

Москва:

129085, Москва, пр-т Мира, д. 101в, стр. 1
(м. Алексеевская)
Тел.: (495) 648-0507, факс: (495) 616-0029
E-mail: sale@academia-moscow.ru

Филиалы:

Северо-Западный
194044, Санкт-Петербург, ул. Чугунная,
д. 14, оф. 319
Тел./факс: (812) 244-92-53
E-mail: spboffice@acadizdat.ru

Приволжский
603101, Нижний Новгород, пр. Молодежный,
д. 31, корп. 3
Тел./факс: (831) 259-7431, 259-7432, 259-7433
E-mail: pf-academia@bk.ru

Уральский
620142, Екатеринбург, ул. Чапаева, д. 1а, оф. 12а
Тел.: (343) 257-1006
Факс: (343) 257-3473
E-mail: academia-ural@mail.ru

Сибирский
630009, Новосибирск, ул. Добролюбова, д. 31, корп. 4, а/я 73
Тел./факс: (383) 362-2145, 362-2146
E-mail: academia_sibir@mail.ru

Дальневосточный
680038, Хабаровск, ул. Серышева, д. 22, оф. 519, 520, 523
Тел./факс: (4212) 56-8810
E-mail: filialdv-academia@yandex.ru

Южный
344082, Ростов-на-Дону, ул. Пушкинская,
д. 10/65
Тел.: (863) 203-5512
Факс: (863) 269-5365
E-mail: academia-UG@mail.ru

Представительства:

в Республике Татарстан
420034, Казань, ул. Горсоветская,
д.17/1, офис 36
Тел./факс: (843) 562-1045
E-mail: academia-kazan@mail.ru

в Республике Дагестан
Тел.: 8-928-982-9248

www.academia-moscow.ru