Т. А. ГАВРИЛОВА,
Д. В. КУДРЯВЦЕВ,
Д. И. МУРОМЦЕВ

ИНЖЕНЕРИЯ ЗНАНИЙ

МОДЕЛИ И МЕТОДЫ

Учебник

ЛАНЬ®
САНКТ-ПЕТЕРБУРГ • МОСКВА • КРАСНОДАР
2016

ББК 32.81я73
Г 12

Гаврилова Т. А., Кудрявцев Д. В., Муромцев Д. И.

Г 12 Инженерия знаний. Модели и методы: Учебник. — СПб.: Издательство «Лань», 2016. — 324 с.: ил. — (Учебники для вузов. Специальная литература).

ISBN 978-5-8114-2128-2

Учебник посвящен вопросам анализа и структурирования данных и знаний для разработки информационных и интеллектуальных систем. В нем детально рассматривается полный цикл разработки интеллектуальных систем — от извлечения и формализации знаний до программной реализации компонентов системы. Материал книги дает наиболее полное представление о современном состоянии развития инженерии знаний в России и мире.

Учебник будет полезен студентам, аспирантам, специалистам и разработчикам информационных систем, особенно интеллектуальных систем, а также бизнес-аналитикам и менеджерам, интересующимся данной проблематикой. Это первый учебник по инженерии знаний, полезное также при разработке корпоративных систем управления знаниями.

ББК 32.81я73

Рецензенты:
С. А. АРУСТАМОВ — доктор технических наук, профессор Санкт-Петербургского национального исследовательского университета информационных технологий, механики и оптики;
А. И. ВОДЯХО — доктор технических наук, профессор кафедры вычислительной техники Санкт-Петербургского государственного электротехнического университета.

Обложка
Е. А. ВЛАСОВА

© Издательство «Лань», 2016
© Коллектив авторов, 2016
© Издательство «Лань», художественное оформление, 2016

ПРЕДИСЛОВИЕ

Есть формы и ритмы, недоступные взгляду
наблюдателя, но открытые взгляду АНАЛИТИКА.
Р. Фейнман

В XXI в. множество профессий связано с интенсивной многофункциональной переработкой различных видов информации, постепенно большинство людей испытывает стресс — человечество захлебывается в информационном море. Компьютер и различные мобильные гаджеты могли бы стать помощником, но зачастую становятся дополнительным источником лишней и бесполезной информации. Специалисты нуждаются в интеллектуальном помощнике, который бы смог отфильтровать, обобщить и сжать информационный поток. Инженерия знаний — первая наука о способах компрессии информации. На ее основе возможно тиражирование знаний через интеллектуальные системы, которые призваны помочь специалистам в обработке сложной информации, принятии решений, анализе и интерпретации данных. Для создания ЛЮБОЙ компьютерной системы необходим АНАЛИТИК — человек, предварительно проанализировавший проблему и поставивший задачу машине. Но и помимо интеллектуальных систем инженерия знаний помогает сделать знания видимыми, это нужно для лучшего понимания проблем и широко используется в ведущих компаниях для управления корпоративными знаниями.

Этот учебник — первая книга для аналитиков. Под аналитиками будем понимать как системных аналитиков и архитекторов, проектирующих программные системы, так и бизнес-аналитиков, ставящих для них задачи. Так сложилось, что первыми о проблеме структурирования знаний задумались разработчики интеллектуальных систем. Но сегодня инженерия знаний перешагнула рамки разработки интеллектуальных систем и превратилась в междисциплинарную область, исследующую проблемы извлечения, структурирования, представления и обработки знаний. Решение этих задач тесно переплетается с разными научными направлениями менеджмента и маркетинга, психологии и философии, лингвистики и математики, биологии и кибернетики.

Большинство глав этой книги рассчитано на бизнес-аналитиков и постановщиков задач информатизации, требующих анализа информации. Акцент ставится на новых визуальных моделях структурирования знаний (интеллект-карты, концептуальные графы и др.), которые служат мощным инструментом анализа сложных предметных областей. Также эти методы полезны преподавателям для наглядного и компактного изложения материала.

Сейчас уже никому не нужно доказывать актуальность систем, основанных на знаниях, более того, знания стали модным флагом, под которым стремительно набирает обороты достаточно молодая наука — управление знаниями. Проблематика обработки знаний становится центральной в современном менеджменте. Специалисты уходят, унося с собой свой интеллектуальный капитал. Управление знаниями трактуется как совокупность процессов, которые управляют созданием, распространением, обработкой и использованием знаний внутри предприятия.

Особенно глубоко и детально рассматривается полный цикл разработки интеллектуальных систем тиражирования знаний — от извлечения и формализации знаний до реализации компонентов прикладной интеллектуальной системы.

При этом существенный акцент делается именно на инженерии знаний и онтологическом инжиниринге, новом разделе инженерии знаний, ориентированном на разработку концептуальных моделей любых сложных систем от государства до человеческого организма. Эти модели создаются именно для того, чтобы моделировать и тиражировать знания, которыми обладают люди-эксперты.

При этом вопросы представления знаний в памяти человека и процесс рассуждений и по сей день остаются мало изученными. Кажущийся на первый взгляд всесиль-

ным, аппарат логических рассуждений не может дать приемлемого решения для целого класса задач. Во многом это объясняется тем, что логика традиционно занималась так называемыми строгими рассуждениями, оставляя за бортом другие многочисленные формы человеческих рассуждений. Представления о модели памяти человека в психологии весьма громоздки и мало что дают для построения схемы рассуждений человека в конкретных ситуациях.

Авторы предприняли попытку максимально широко описать различные аспекты проектирования баз знаний и систем управления знаниями. Последовательно и детально рассматриваются вопросы, которые необходимо решить разработчику в процессе создания новой системы. Приводится описание возможных альтернативных решений, с указанием области применения тех или иных подходов, дается историческая справка.

В первой главе рассматриваются история интеллектуальных систем и современные направления разработок. Более подробно рассматриваются системы, основанные на знаниях. В этой части даются определения фундаментальных понятий. Вторая глава посвящена теоретическим и практическим вопросам извлечения знаний. Подробно рассматриваются современные подходы к построению ментальных моделей. Третья глава касается вопросов структурирования знаний. Это, пожалуй, наиболее трудоемкий этап при построении интеллектуальных систем. В четвертой главе рассматриваются фундаментальные модели представления знаний, на основе которых развиваются современные способы работы со знаниями. Пятая глава посвящена самым популярным на сегодня моделям представления знаний — онтологиям и онтологическому инжинирингу. В шестой показано, как могут применяться онтологии, в особенности, для задач моделирования коммерческих компаний, органов власти и городов. В седьмой главе рассмотрены системы на основе онтологий и средства их разработки.

Эта книга предназначена как для тех, кто интересуется проблемами информационного менеджмента и управления знаниями, так и для тех, кто изучает и проектирует интеллектуальные системы.

Порядок чтения данной книги зависит от конкретных целей читателя. Во всех случаях рекомендуется начать с первой части. Далее, если книга используется как основа лекционного курса, то материал может излагаться последовательно, в соответствии с содержанием книги. При использовании книги в качестве справочника можно ограничиться выборочным чтением глав, поскольку главы достаточно независимы.

Главы 1–3, введение и заключение написаны Т. А. Гавриловой, глава 4 написана Д. И. Муромцевым, главы 5–7 написаны Д. В. Кудрявцевым. Параграфы 3.2–3.4 и 3.7 написаны совместно Т. А. Гавриловой и Д. В. Кудрявцевым. Параграф 7.2 написан студенткой СПбПУ Викторией Рупасовой и отредактирован Д. В. Кудрявцевым, параграфы 7.3 в части «Использование онтологий для создания автомобилей без водителя» и 7.4 написаны студенткой СПбПУ Ульяной Дудко и отредактированы Д. В. Кудрявцевым. Параграф 2.6 основан на материале А. В. Блинниковой, параграф 3.8 написан на основе материалов Э. Р. Сукиасяна, Р. С. Гиляревского, А. В. Шапкина, В. Н. Белоозерова, параграф 5.4 — на основе материалов Н. В. Лукашевич, параграф 5.7 — на основе работы авторского коллектива Б. В. Добров, В. В. Иванов, Н. В. Лукашевич, В. Д. Соловьев. При подготовке параграфов 7.1 и 7.3 были использованы материалы Ю. А. Загорулько и Г. Б. Загорулько.

Авторы выражают глубокую благодарность своим учителям, коллегам и семьям за поддержку и помощь в написании этой книги.

ЧАСТЬ I

Глава 1

ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ

1.1. ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ: КРАТКАЯ ИСТОРИЯ

Загадки феномена мышления и жизни волновали человечество с древних времен. Так, идея создания искусственного подобия человека с элементами «разумности» восходит к египтянам, которые в III веке до н. э. создали «оживающую» механическую статую бога Амона, выдающую по запросу «предсказания» будущего. У Гомера в «Илиаде» бог Гефест ковал человекоподобные существа-автоматы. Сократ мечтал о механистическом «бесстрастном» судье.

Предыстория

Родоначальником искусственного интеллекта считается средневековый испанский философ, математик и поэт *Раймонд Луллий*, который еще в XIII в. попытался создать механическую машину для решения различных задач на основе разработанной им всеобщей классификации понятий.

В XVIII в. великие математики *Лейбниц* и *Декарт* независимо друг от друга продолжили эту идею, предложив универсальные языки классификации всех наук. Эти работы можно считать первыми теоретическими работами в области искусственного интеллекта. Однако только изобретенный Чарльзом Бэббиджем (Charles Babbage) в 1833 г. механический калькулятор может считаться прямым предком интеллектуального компьютера. Его помощник, леди Ада Лавлейс (Lady Lovelace), племянница Байрона, стала первым пользователем и программистом.

Краткая история

Рождение искусственного интеллекта как научного направления произошло только после создания ЭВМ в 40-х гг. XX в. В это же время Норберт Винер создал свои основополагающие работы по новой науке — кибернетике.

Термин «искусственный интеллект» — ИИ (artificial intelligence — AI) — был предложен в 1956 г. в рамках летнего проекта с аналогичным названием в Дартсмутском колледже (США). Проект был рассчитан на 2 месяца, в нем участвовали 10 человек. Тема семинара — разработка логических, а не вычислительных задач. В английском языке данное словосочетание не имеет той слегка фантастической антропоморфной окраски, которую оно приобрело в довольно неудачном русском переводе. Слово intelligence означает «умение рассуждать разумно», а вовсе не «интеллект», для которого есть термин intellect.

Стоит подробнее рассказать о Дартсмутском семинаре, один список участников которого уже вошел в историю ИИ, как легендарный: Джон Маккарти (John McCarthy) — отец языка ЛИСП (LISP) и глава лаборатории ИИ в Стэнфордском университете, Марвин Минский (Marvin Minsky) — пионер ИИ и автор идеи фрейма, Клод Шэннон (Claude Shannon) — лауреат Нобелевской премии, основоположник теории информации. Семинар начался с обсуждения программы «Логик-теоретик» для автоматического доказательства теорем, причем одна из теорем неожиданно была доказана новым улучшенным способом. К концу второго месяца идея ИИ была сформулирована, амбиции и энтузиазм участников были беспредельны — научить машину осуществлять некоторые «разумные», осмысленные, а не просто вычислительные процедуры, например, играть в шахматы, переводить на иностранные языки или сочинять музыку. Кое-что из этого сбылось.

Вскоре после признания искусственного интеллекта отдельной областью науки произошло разделение его на два направления: *нейрокибернетика* и *кибернетика «черного ящика»*. Эти направления развиваются практически независимо, существенно различаясь как в методологии, так и в технологии. И только в 90-е гг. стали заметны тенденции к объединению этих частей вновь в единое целое.

Основную идею нейрокибернетики можно сформулировать следующим образом: единственный объект, способный мыслить, — это человеческий мозг, поэтому любое «мыслящее» устройство должно каким-то образом воспроизводить его структуру.

Таким образом, нейрокибернетика ориентирована на программно-аппаратное моделирование структур, подобных структуре мозга. Физиологами давно установлено, что основой человеческого мозга является большое количество (до 10^{21}) связанных между собой и взаимодействующих нервных клеток — нейронов. Поэтому усилия нейрокибернетики были сосредоточены на создании элементов, аналогичных нейронам, и их объединении в функционирующие системы. Эти системы принято называть *нейронными сетями*, или *нейросетями*.

Первые нейросети были созданы *Розенблаттом* и *Мак-Каллоком* в 1956–1965 гг. Это были попытки создать системы, моделирующие человеческий глаз и его взаимодействие с мозгом. Устройство, созданное ими тогда, получило название *персептрона (perceptron)*. Оно умело различать буквы алфавита, но было чувствительно к их написанию. Например, буквы **A**, A и *A* для этого устройства были тремя разными знаками. Постепенно в 70–80-х гг. количество работ в этом направлении искусственного интеллекта стало снижаться. Слишком неутешительны были первые результаты. Авторы объясняли неудачи малой памятью и низким быстродействием существующих в то время компьютеров.

Однако в 1980-х гг. в Японии в рамках проекта «ЭВМ V поколения» был создан первый *нейрокомпьютер*, или компьютер VI поколения. К этому времени ограничения по памяти и быстродействию были практически сняты. Появились *транспьютеры* — параллельные компьютеры с большим количеством процессоров.

Транспьютерная технология — это только один из десятка новых подходов к аппаратной реализации нейросетей, которые моделируют иерархическую структуру мозга человека. Основная область применения нейрокомпьютеров — распознавание образов.

В целом, можно выделить 3 подхода к созданию нейросетей:

1. *Аппаратный* — создание специальных компьютеров, нейрочипов, плат расширения, наборов микросхем, реализующих все необходимые алгоритмы.

2. *Программный* — создание программ и инструментариев, рассчитанных на высокопроизводительные компьютеры. Сети создаются в памяти компьютера, всю работу выполняют его собственные процессоры.

3. *Гибридный* — комбинация первых двух. Часть вычислений выполняют специальные платы расширения (сопроцессоры), часть — программные средства.

Более глубокое рассмотрение этого перспективного и интересного направления исследований искусственного интеллекта выходит за рамки данной книги (подробнее см. [Каллан, 2001; Галушкин, 2000; Горбань, 1996, Тадеусевич, 2011]). Примером программной реализации нейросетей является, например, система SIMBRAIN. Сравнение систем программной эмуляции нейросетей выполнено в университете Колорадо в США (https://grey.colorado.edu/emergent/index.php/Comparison of Neural Network Simulators).

В основу подхода, названного кибернетикой «черного ящика», был положен принцип, противоположный нейрокибернетике: «Не имеет значения, как устроено “мыслящее” устройство. Главное, чтобы на заданные входные воздействия оно реагировало так же, как человеческий мозг».

Сторонники этого направления мотивировали свой подход тем, что человек не должен слепо следовать природе в своих научных и технологических поисках. Так, например, очевиден успех колеса, которого не существует в природе, или самолета, не машущего крыльями, подражая птице. К тому же пограничные науки о человеке не смогли внести существенного теоретического вклада, объясняющего хотя бы приблизительно, как протекают интеллектуальные процессы у человека, как устроена память и как человек познает окружающий мир.

Данное направление искусственного интеллекта было ориентировано на поиски алгоритмов решения интеллектуальных задач на существующих моделях компьютеров. Существенный вклад в становление новой науки внесли ее «пионеры»: Маккарти (уже упомянутый автор первого языка программирования для задач ИИ — ЛИСПа), Минский (автор идеи *фрейма* и *фреймовой* модели представления знаний), Ньюэлл, Саймон, Шоу, Хант и другие.

В 1956–1963 гг. велись интенсивные поиски моделей и алгоритмов человеческого мышления и разработка первых программ на их основе. Представители существующих гуманитарных наук — философы, психологи, лингвисты — ни тогда, ни сейчас не могли предложить таких алгоритмов. Тогда кибернетики начали создавать собственные модели. Последовательно были созданы и опробованы различные подходы:

В конце 50-х гг. родилась модель *лабиринтного поиска.* Этот подход представляет задачу как некоторое пространство состояний в форме графа, и в этом графе проводится поиск оптимального пути от входных данных к результирующим. Была проделана большая работа по разработке этой модели, но в решении практических задач идея большого распространения не получила. В первых учебниках по искусственному интеллекту [Хант, 1986; Эндрю, 1985] описаны эти программы — они играют в игру «15», собирают «Ханойскую башню», играют в шашки и шахматы.

Начало 60-х гг. — это эпоха *эвристического программирования*. *Эвристика* — правило, теоретически не обоснованное, которое позволяет сократить количество переборов в пространстве поиска. Эвристическое программирование — разработка стратегии действий на основе известных, заранее заданных эвристик [Александров, 1975].

В 1963–1970-х гг. к решению задач стали подключать методы *математической логики*. *Робинсон* разработал *метод резолюций*, который позволяет автоматически доказывать теоремы при наличии набора исходных аксиом. Примерно в это же время выдающийся отечественный математик Ю. С. Маслов предложил так называемый *обратный вывод*, впоследствии названный его именем, решающий аналогичную задачу другим способом [Маслов, 1983]. На основе метода резолюций француз *Альбер Кольмероэ* в 1973 г. создает язык логического программирования *Пролог*. Большой резонанс имела программа «*Логик-теоретик*», созданная *Ньюэлом*, *Саймоном* и *Шоу,* которая доказывала школьные теоремы. Однако большинство реальных задач не сводится к набору аксиом, и человек, решая производственные задачи, не использует классическую логику, поэтому логические модели при всех своих преимуществах имеют существенные ограничения по классам решаемых задач.

История искусственного интеллекта полна драматических событий, одним из которых стал в 1973 г. так называемый «доклад Лайтхилла», который был подготовлен в Великобритании по заказу Британского совета научных исследований. Известный математик *Д. Лайтхилл,* профессионально никак не связанный с ИИ,

подготовил обзор состояния дел в области ИИ. В докладе были признаны определенные достижения в области ИИ, однако уровень их определялся как разочаровывающий, и общая оценка была отрицательная с позиций практической значимости. Этот отчет отбросил европейских исследователей примерно на 5 лет назад, так как финансирование ИИ существенно сократилось.

Примерно в это же время существенный прорыв в практических приложениях искусственного интеллекта произошел в США, когда к середине 1970-х гг. на смену поискам универсального алгоритма мышления пришла идея моделировать конкретные знания специалистов-экспертов. В США появились первые коммерческие *системы, основанные на знаниях*, или *экспертные системы* (*ЭС*). Пришел новый подход к решению задач искусственного интеллекта — *представление знаний*. Созданы MYCIN и DENDRAL [Shortliffe, 1976; Buchanan, Feigenbaum, 1978], ставшие уже классическими, две первые экспертные системы для медицины и химии. Существенный финансовый вклад вносит Пентагон, предлагая базировать новую программу Министерства обороны США (Strategic Computer Initiative — SCI) на принципах ИИ. Уже вдогонку упущенных возможностей в начале 1980-х гг. объявлена глобальная программа развития новых технологии ESPRIT (Европейский Союз), в которую включена проблематика искусственного интеллекта.

В ответ на успехи США в конце 1970-х гг. в гонку включается Япония, объявив о начале проекта машин V поколения, основанных на знаниях. Проект был рассчитан на 10 лет и объединял лучших молодых специалистов (в возрасте до 35 лет) крупнейших японских компьютерных корпораций. Для этих специалистов был создан специально новый институт ICOT (Институт компьютерной технологии нового поколения). Они получили полную «свободу» действий, правда, без права публикации предварительных результатов. В итоге они создали достаточно громоздкий и дорогой символьный процессор, программно реализующий ПРОЛОГоподобный язык, не получивший широкого признания. Однако положительный эффект этого проекта был очевиден. В Японии появилась значительная группа высококвалифицированных специалистов в области ИИ, которая добилась существенных результатов в различных прикладных задачах. К середине 1990-х гг. японская ассоциация ИИ насчитывает 40 тыс. человек.

Начиная с середины 1980-х гг. повсеместно происходит коммерциализация искусственного интеллекта. Растут ежегодные капиталовложения, создаются промышленные экспертные системы, растет интерес к самообучающимся системам. Издаются десятки научных журналов, ежегодно собираются международные и национальные конференции по различным направлениям ИИ.

ИИ прошел длинный путь от момента своего рождения в 1956 г. до настоящего времени. К XXI в. искусственный интеллект остается одной из наиболее перспективных и престижных областей информатики (computer science). Основные вехи истории ИИ представлены в «History of Artificial Intelligence», приложение 1 [Stottler, 2002]. Успехи налицо, но «голубая мечта» создателей так и не достигнута. Идеальный интеллектуальный компьютер ждет нас впереди.

«Машинный разум» — тест Тьюринга

Алан Тьюринг (Alan M. Turing) уже в 30-е гг. предсказал возможность машинного разума. Позже, в 1950 г. Алан Тьюринг в статье «Вычислительные машины и разум» (*Computing machinery and intelligence*) предложил тест для проверки, является ли компьютер разумным в человеческом смысле слова. Тьюринг предложил тест, чтобы заменить бессмысленный, по его мнению, вопрос «может ли машина мыслить?» на более определенный.

Тест организован следующим образом. Судья (человек) переписывается на естественном языке с двумя собеседниками, один из которых — человек, другой — компьютер. Если судья не может надежно определить, кто есть кто, считается, что компьютер прошел тест. Предполагается, что каждый из собеседников стремится, чтобы человеком признали его. Тест должен проходить в виде переписки сообщениями. Переписка должна производиться через контролируемые промежутки времени, чтобы судья не мог делать заключения исходя из скорости ответов. Во времена Тьюринга компьютеры реагировали медленнее человека. Сейчас это правило необходимо, потому что компьютер реагирует гораздо быстрее, чем человек. Идея теста была навеяна салонной игрой, в ходе которой гости пытались угадать пол человека, находящегося в другой комнате, путем написания вопросов и чтения ответов. В оригинальной формулировке человек должен был притворяться человеком противоположного пола, а тест длился 5 минут. Тьюринг предсказал, что компьютеры в конечном счете пройдут его тест. Он считал, что к 2000 г. компьютер с памятью 1 миллиард бит (около 119 МБ) в ходе 5-минутного теста сможет обмануть судей в 30% случаев. Это предсказание не сбылось. Тьюринг также предсказал, что сочетание «мыслящая машина» не будет считаться нелепым, а обучение компьютеров будет играть важную роль в создании мощных компьютеров (с чем большинство современных исследователей согласны). Пока (на 2016 г.) ни одна программа не прошла успешно тест Тьюринга, хотя многочисленные программы могут создавать иллюзию беседы.

Первой попыткой такого рода была знаменитая «Элиза» Вейценбаума [Weizenbaum, 1966]. Появляются сообщения, что некоторые программы приблизились к 30%. Авторы книги не разделяют этого оптимизма. Эта книга дает представление о том, как представлять в машине опыт и знания профессионалов, но тайны здравого смысла, интуиции и творчества до сих пор не поддаются формализации. Пока машины «умны» настолько, насколько позволяет их база знаний.

История и современное состояние ИС в России

В 1954 г. в МГУ начал свою работу семинар «Автоматы и мышление» под руководством академика *А. А. Ляпунова* (1911–1973), одного из основателей российской кибернетики. В этом семинаре принимали участие физиологи, лингвисты, психологи, математики. Принято считать, что именно в это время родился искусственный интеллект в России. Как и за рубежом, выделились два основных направления — нейрокибернетики и кибернетики «черного ящика».

В 1954–1964 гг. создаются отдельные программы и проводятся исследования в области поиска решения логических задач. В Ленинграде (ЛОМИ — Ленинградское отделение Математического института им. Стеклова) создается программа, автоматически доказывающая теоремы (АЛПЕВ ЛОМИ). Она основана на оригинальном обратном выводе Маслова, аналогичном методу резолюций Робинсона. Среди наиболее значимых результатов, полученных отечественными учеными в 60-е гг., следует отметить алгоритм «Кора» *М. М. Бонгарда,* моделирующий деятельность человеческого мозга при распознавании образов. Большой вклад в становление российской школы ИИ внесли выдающиеся ученые *М. Л. Цетлин, В. Н. Пушкин, М. А. Гаврилов,* чьи ученики и явились пионерами этой науки в России (например, знаменитая *Гавриловская* школа).

В 1965–1980 гг. происходит рождение нового направления, связанного с ИИ, — *ситуационного управления* (соответствует представлению знаний, в западной терминологии). Основателем этой научной школы стал проф. *Д. А. Поспе-*

лов. Были разработаны специальные модели представления ситуаций — представления знаний [Поспелов, 1986].

Отношение к новым наукам в Советской России всегда было настороженное, и наука с таким «вызывающим» названием также не избежала этой участи и была встречена в Академии наук в штыки [Поспелов, 1997]. К счастью, даже среди членов Академии наук СССР нашлись люди, не испугавшиеся столь необычного словосочетания в качестве названия научного направления. Двое из них сыграли огромную роль в борьбе за признание ИИ в нашей стране. Это были академики *А. И. Берг* и *Г. С. Поспелов.*

Только в 1974 г. при Комитете по системному анализу при Президиуме АН СССР был создан Научный совет по проблеме «Искусственный интеллект». Его возглавил Г. С. Поспелов, заместителями были избраны Д. А. Поспелов и Л. И. Микулич. В состав Совета входили на разных этапах М. Г. Гаазе-Рапопорт, Ю. И. Журавлев, Л. Т. Кузин, А. С. Нариньяни, Д. Е. Охоцимский, А. И. Половинкин, О. К. Тихомиров, В. В. Чавчанидзе.

По инициативе Совета было организовано пять комплексных научных проектов, возглавляемые ведущими специалистами в данной области. Проекты объединяли исследования в различных коллективах страны: «Диалог» (работы по пониманию естественного языка, руководители — А. П. Ершов, А. С. Нариньяни), «Ситуация» (ситуационное управление — Д. А. Поспелов), «Банк» (банки данных — Л. Т. Кузин), «Конструктор» (поисковое конструирование — А. И. Половинкин), «Интеллект робота» (Д. Е. Охоцимский).

В 1980–1990 гг. проводятся активные исследования в области представления знаний, разрабатываются языки представления знаний, экспертные системы (более 300). В Московском университете создается язык РЕФАЛ.

Ассоциация искусственного интеллекта (АИИ) создается в 1988 г. по инициативе Дмитрия Александровича Поспелова. Ее членами становятся более 300 исследователей. Президентом Ассоциации избирается Д. А. Поспелов, выдающийся ученый, чей вклад в развитие ИИ в России трудно переоценить. Крупнейшие центры создаются в Москве, Петербурге, Переславле-Залесском, Новосибирске, Владивостоке. В Научный совет ассоциации вошли ведущие исследователи в области ИИ — А. Н. Аверкин, В. П. Гладун, В. И. Городецкий, Г. С. Осипов, Э. В. Попов, В. Л. Стефанюк, В. Ф. Хорошевский, В. К. Финн, Г. С. Цейтин, А. С. Эрлих и другие ученые. В рамках Ассоциации проводится большое количество исследований, организуются школы для молодых специалистов, семинары, симпозиумы, раз в два года собираются объединенные конференции, издается научный журнал «Искусственный интеллект и принятие решений». Научные исследования ведутся в ведущих университетах РФ и институтах РАН (под руководством В. В. Александрова, В. Н. Вагина, Т. А. Гавриловой, В. И. Городецкого, А. П. Еремеева, К. А. Кобринского, О. П. Кузнецова, Г. С. Осипова, В. Г. Редько, Г. В. Рыбиной, А. В. Смирнова, В. Б. Тарасова, В. Ф. Хорошевского, Т. В. Черниговской и многих других специалистов). Активно работают отделения ассоциации и отдельные группы исследователей в различных городах России и ближнего зарубежья: В. В. Голенков (Минск), В. П. Гладун и Ю. Р. Валькман (Киев), Ю. А. Загорулько (Новосибирск), В. М. Курейчик (Таганрог), А. Е. Клещев (Владивосток), В. Д. Соловьев (Казань), В. Ш. Рубашкин (Санкт-Петербург). В 2015 г. умер талантливый ученый, один из пионеров этой области — Николай Григорьевич Загоруйко (Новосибирск).

Уровень теоретических исследований по искусственному интеллекту в России долгое время соответствовал европейскому. К сожалению, начиная с 80-х гг. на

прикладных работах начало сказываться постепенное отставание в технологии. На данный момент отставание в области разработки промышленных интеллектуальных систем составляет порядка 3–5 лет.

1.2. ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ: НАПРАВЛЕНИЯ РАЗРАБОТОК

В данной книге в качестве рабочего определения можно согласиться с «коллективным разумом» Википедии [www.wikipedia.org]:

> Искусственный интеллект (англ. Artificial intelligence (AI)) — раздел информатики, изучающий возможность обеспечения разумных рассуждений и действий с помощью вычислительных систем и иных искусственных устройств. При этом в большинстве случаев алгоритм решения задачи заранее неизвестен.

Сегодня дерево ИИ объединяет десятки наук, однако максимум усилий и финансирования имеют несколько ведущих направлений, которые опишем чуть подробнее.

А) Интеллектуальные игры и компьютерное творчество

Во времена ранних исследований по ИИ машинный интеллект ассоциировался с логическими играми, и ставилась задача «научить» компьютер решать игровые задачи — шахматы, шашки, го. Первые программы на любительском уровне играли уже в 1957 г. Их автор, Алекс Бернштейн, также был участником Дартсмутского семинара. В основе программ лежал один из ранних подходов — лабиринтная модель мышления плюс эвристики. Позже в 1958 г. Ньюэлл, Саймон и Шоу усовершенствовали алгоритм.

В России это направление также было весьма успешно, в связи с традиционной силой отечественной шахматной и математической школ [Адельсон-Вельский, Арлазаров, Битман и Донской, 1983; Ботвинник, 1983]. Так, в 1974 г. состоялся международный шахматный турнир машин, снабженных соответствующими программами. Победу на этом турнире одержала советская шахматная программа «Каисса», разработанная при участии М. М. Ботвинника — чемпиона мира по шахматам.

Успех программ, выигравших даже у Каспарова, производит впечатление. Сейчас это скорее коммерческое направление, так как в научном плане эти идеи считаются тупиковыми. Однако, в 2016 г. программа компании Google победила чемпиона по игре го.

Кроме того, это направление охватывает сочинение компьютером музыки [Зарипов, 1983, Ерохин, 2011], стихов, сказок [Справочник по ИИ, 1986] и даже афоризмов [Любич, 1998]. Основным методом подобного «творчества» является метод *пермутаций* (перестановок) плюс использование некоторых баз знаний и данных, содержащих результаты исследований по структурам текстов, рифм, сценариям и т. п. Сейчас это направление рассматривается уже как «историческое», и всерьез этим занимаются лишь отдельные группы исследователей.

Б) Представление знаний и разработка систем, основанных на знаниях (knowledge-based systems)

Именно данному направлению посвящена эта книга и далее будет подробно рассмотрен этот класс систем.

В) Машинное обучение (machine learning, data mining, knowledge discovery)

Машинное обучение (machine learning, data mining, knowledge discovery) — активно развивающаяся область искусственного интеллекта, которая включает модели, методы и алгоритмы, ориентированные на автоматическое накопление и формирование знаний на основе анализа и обобщения данных [Гаек, Гавранек, 1984; Гладун, 1987; Финн, 1976, 2006]. Используются также обучение на примерах (или индуктивное) и традиционные подходы из теории распознавания образов. Для систем класса machine learning существует «свободный» софт — это системы SLIPPER, Orange, JMLR, Weka3, YALE, Bayda и другие.

В последние годы к этому направлению тесно примыкают стремительно развивающиеся системы data mining — анализа данных и knowledge discovery — поиска закономерностей.

Г) Понимание естественных языков (natural language processing)

Начиная с 50-х гг., одной из популярных тем исследований в области ИИ является область компьютерной лингвистики, и в частности машинного перевода. Машинный перевод — выполняемое на компьютере действие по преобразованию текста на одном естественном языке в эквивалентный по содержанию текст на другом языке, а также результат такого действия [Нелюбин, 1991, Гращенко, 2011].

В настоящее время для понимания и перевода используется несколько более сложных моделей:

- использование так называемых *«языков-посредников»*;
- *ассоциативный поиск* аналогичных фрагментов текста и их переводов в специальных текстовых репозиториях или базах данных;
- *структурный подход*, включающий последовательный анализ и синтез естественно-языковых сообщений. Традиционно такой подход подразумевает наличие нескольких фаз анализа:
  - *Морфологический анализ* — анализ слов в тексте.
  - *Синтаксический анализ* — разбор состава предложений и грамматических связей между словами.
  - *Семантический анализ* — анализ смысла составных частей каждого предложения на основе некоторой предметно-ориентированной базы знаний.
  - *Прагматический анализ* — анализ смысла предложений в реальном контексте на основе собственной базы знаний.

Таким образом, система машинного перевода включает в себя двуязычные словари, снабженные необходимой грамматической информацией (морфологической, синтаксической и семантической) для передачи переводных соответствий.

Примеры показывают необходимость участия человека-редактора в переводе. Полностью автоматический перевод в настоящее время невозможен, а качество программ-переводчиков пока далеко от совершенства (PROMT's Online Translator; Microsoft's Bing Translator; Google Translate; Systran; Babylon; и др.).

Синтез сообщений на естественных языках (ЕЯ) несколько легче, чем анализ, но также требует развернутой базы знаний (подробнее см. [Попов, 2004; Мальковский, 1985; Судаков, 2012]).

Серьезный анализ этой чрезвычайно перспективной области ИИ выходит за рамки данной книги.

Д) Распознавание образов (pattern recognition)

Распознавание образов (pattern recognition) — одно из направлений искусственного интеллекта, исходящее из самых его истоков, но в настоящее время практически выделившееся в самостоятельную науку. Ее основной подход — описание *классов объектов* через определенные значения значимых признаков. Каждому объекту ставится в соответствие матрица признаков, по которой происходит его распознавание. Процедура распознавания использует чаще всего специальные математические процедуры и функции, разделяющие объекты в классы. Это направление близко к машинному обучению, тесно связано с нейрокибернетикой [Загоруйко, 1999; Справочник по ИИ, 1990; Фомин, 2012].

Е) Интеллектуальные роботы (robotics)

Идея создания интеллектуальных роботов далеко не нова. Само слово «робот» появилось в 1920-х гг. как производное от чешского «робота» — тяжелой грязной работы. Его автор — чешский писатель Карел Чапек, описавший роботов в своем рассказе «РУР».

> Роботы — это электротехнические устройства,
> предназначенные для автоматизации человеческого труда.

Можно выделить несколько условных поколений в истории создания и развития робототехники:

I поколение: *роботы с жесткой схемой управления*. Практически все современные промышленные роботы принадлежат к первому поколению. Фактически это программируемые манипуляторы;

II поколение: *адаптивные роботы с сенсорными устройствами*. Есть образцы таких роботов, но в промышленности они пока используются мало;

III поколение: *самоорганизующиеся,* или *интеллектуальные, роботы*. Это конечная цель развития робототехники. Основные нерешенные проблемы при создании интеллектуальных роботов — проблема машинного зрения и адекватного хранения и обработки трехмерной визуальной информации.

В настоящее время в мире изготавливается более 60 000 роботов в год. Фактически робототехника сегодня — это инженерная наука, не отвергающая технологий ИИ, но не готовая пока к их внедрению в силу различных причин.

В 1980 г. Япония достигла лидирующей позиции в мире по производству роботов. 50% всех промышленных роботов сделано и используется в Японии. С тех пор Япония не теряет лидирующей позиции, вплоть до наших дней. В 2000 г. компания Honda разработала робота-гуманоида Asimo, последние версии которого являются одним из наиболее продвинутых андроидов (http://asimo.honda.com/). Робот передвигается на достаточно устойчивых ногах и имеет больше степеней свободы, чем его предшественник P3.

Чемпионаты мира по футболу среди роботов ежегодно проводятся с 1997 г. и носят название RoboCup. У андроидов почти все как у настоящих футболистов: голы, нереализованные моменты, подкаты, желтые карточки, пенальти. Пока получается неуклюже, но ученые не унывают: регулярно в Японии разыгрывается Робокубок — три сотни команд роботов играют в одну из самых популярных игр.

Основная тенденция в области робототехники сейчас — миниатюризация, вплоть до нанороботов, запускаемых в кровеносную систему живых организмов.

Однако, в целом, современные роботы продолжают демонстрировать механические успехи, по-прежнему «буксуя» в области интеллектуальности и проблем машинного зрения.

Ж) Новые архитектуры компьютеров (new hardware platforms and architectures)

Самые современные процессоры сегодня основаны на традиционной последовательной архитектуре фон Неймана, используемой еще в компьютерах первых поколений. Эта архитектура крайне неэффективна для символьной обработки. Поэтому усилия многих научных коллективов и фирм уже десятки лет нацелены на разработку аппаратных архитектур, направленных на обработку символьных и логических данных [Амамия, Танака, 1993; Hennessy & Patterson, 2011]. Создаются Пролог- и Лисп-машины, компьютеры V и VI поколений. Последние разработки посвящены компьютерам баз данных, параллельным и векторным компьютерам [Люгер, 2003; Sakamura, 2000; Dubois, Annavaram & Stenström, 2012].

И хотя удачные промышленные решения существуют, высокая стоимость, недостаточное программное оснащение и аппаратная несовместимость с традиционными компьютерами существенно тормозят широкое использование новых архитектур.

З) Программное обеспечение систем ИИ

Разработка программного обеспечения для систем ИИ играет ключевую роль. Существуют специальные языки для решения интеллектуальных задач, в которых акцент ставится на символьную и логическую обработку, например, LISP, PROLOG, SMALLTALK, РЕФАЛ, STRIPS, POP-11 и др. Кроме этого, создаются пакеты прикладных программ, ориентированные на промышленную разработку интеллектуальных систем, или программные инструментарии искусственного интеллекта, например KEE, ARTS, G2, Watson [Джексон, 2001; Попов и др., 1996; Tzou, Anderson, 2013]. Достаточно популярно также создание так называемых пустых экспертных систем или «оболочек»: KAPPA, EXSYS, E2glite и др. [Джарратано, Райли, 2006; Negnevitsky, 2002], которые можно наполнять базами знаний, создавая различные экспертные системы.

И) Многоагентные системы

В одном из наиболее популярных современных учебников по ИИ [Рассел и Норвиг, 2003] под агентом понимается «любая сущность, которая находится в некоторой среде, воспринимает ее посредством сенсоров, получая данные, которые отражают события, происходящие в среде, интерпретирует эти данные и действует на среду посредством эффекторов». Таким образом, здесь вычленяются четыре исходных агентообразующих фактора — среда, восприятие, интерпретация, действие.

Разработка технологии *искусственных агентов*, создание *многоагентных систем* (МАС) является одной из наиболее важных и многообещающих областей развития ИИ. У специалистов существует представление об *искусственных агентах* как *активных, автономных, коммуникабельных, а главное — мотивированных объектах,* «живущих» и «действующих» *в сложных, динамических и чаще всего в виртуальных средах* [Городецкий, 1995; Тарасов, 2002].

Агентно-ориентированный подход находит применение в таких областях, как распределенное решение сложных задач, совмещенное проектирование изделий, реинжиниринг бизнеса и построение виртуальных предприятий, имитационное

моделирование интегрированных производственных систем и электронная торговля, организация работы коллективов роботов и распределенная (совмещенная) разработка компьютерных программ. В ближайшем будущем он, возможно, займет центральное место при развитии средств управления информацией и знаниями, при создании и внедрении новейших систем телекоммуникации, развитии глобальных компьютерных сетей.

При определении понятия *«агент»* часто опираются на представление об объекте, развитом школой объектно-ориентированного программирования (ООП). Тогда *искусственный агент* может пониматься как *метаобъект*, наделенный некоторой долей *субъектности*, т. е. способный манипулировать другими объектами, создавать и уничтожать их, а также имеющий развитые средства взаимодействия со средой и себе подобными. Набор базовых характеристик агента включает:

1) *активность* — способность к организации и реализации действий;

2) *автономность* (полуавтономность) — относительная независимость от окружающей среды или наличие некоторой «свободы воли», связанное с хорошим ресурсным обеспечением его поведения;

3) *общительность* — вытекающая из необходимости решать свои задачи совместно с другими агентами и обеспечиваемая развитыми протоколами коммуникации;

4) *целенаправленность* — предполагающая наличие собственных источников *мотивации*, а в более широком плане — специальных интенциональных характеристик. Такое представление близко к одному из наиболее популярных определений понятия «агент» у М. Вулдриджа и Н. Дженнингса [Wooldridge, 2002].

Иногда агентные технологии называют распределенным ИИ (РИИ) (distributed AI). Ядро РИИ [Тарасов, 2002] составляют исследования взаимодействия небольшого числа интеллектуальных агентов, включающих базы знаний и решатели. Главной проблемой в РИИ является разработка интеллектуальных групп и организаций, способных решать задачи путем рассуждений, связанных с обработкой символов. Это предполагает согласование целей, интересов и стратегий различных агентов, координацию действий, разрешение конфликтов путем переговоров; теоретическую базу здесь составляют результаты, полученные в психологии малых групп и социологии организаций.

Важным разделом РИИ является кооперативное распределенное решение задач (КРРЗ). Речь идет о сети слабо связанных между собой решателей, которые совместно работают в целях решения задач, выходящих за рамки индивидуальных возможностей. Различные узлы подобной сети, как правило, имеют неодинаковый опыт (знания, точки зрения) и разные ресурсы.

Достаточно активно в разработке МАС проявляется направление, под названием *«искусственная жизнь»* (ИЖ) (в узком смысле термина artificial life). Оно связано с трактовкой интеллектуального поведения в контексте выживания, адаптации и самоорганизации в динамичной среде (ранее этим занимался психолог Ж. Пиаже). В русле ИЖ глобальное поведение всей системы рассматривается как результат локальных взаимодействий простых и необязательно интеллектуальных агентов. Здесь также используются термины «коллективный интеллект» (collective intelligence), или «интеллект роя» (swarm intelligence). Типичными примерами такого коллективного интеллекта из биологии являются колонии муравьев, пчелиные ульи и т. п. Соответственно, здесь базовыми дисциплинами могут служить различные области биологической науки и в первую очередь — эволюционная теория и генетика.

Существуют специальные программные среды и языки для взаимодействия агентов, например, KIF, KQML, FIPA-ACL и др. В последнее время активно используется онтологический подход (см. главу об онтологическом инжиниринге).

Другие направления

Как междисциплинарная наука, ИИ «питается» от смежных дисциплин — психологии, математики, лингвистики, философии, нейрофизиологии и нескольких других. Выше перечислены лишь те направления, которые прямо или косвенно связаны с основной тематикой данной книги — инженерией знаний. Стоит лишь взглянуть на основные рубрикаторы конференций по ИИ, чтобы понять, насколько широко простирается область исследований по ИИ:

- автоматическое доказательство теорем (theorem proving);
- автоматические рассуждения (automated reasoning);
- автоматическое обучение (machine learning);
- агенты и многоагентные системы (agents and multi-agent systems);
- байесовы сети (Bayesian networks);
- виртуальная реальность (virtual reality);
- генетические алгоритмы (evolutionary computation, genetic algorithms);
- дедуктивные модели (deductive models);
- игры (game playing);
- инженерия знаний (knowledge engineering);
- интеллектуальные базы данных (intelligent databases);
- инженерия знаний (knowledge engineering);
- интеллектуальные интерфейсы (intelligent interfaces);
- интеллектуальный Интернет (интеллектуальная паутина) (Internet / WWW intelligence, Semantic Web);
- искусственная жизнь (artificial life);
- когнитивное моделирование (cognitive modeling);
- концептуальные карты-графы (concept maps and graphs);
- логики (индуктивные, дескриптивные, нечеткие и др.) (logics (inductive / descriptive / fuzzy, etc.);
- логики пространства и времени (spatial / temporal reasoning);
- машинное творчество (music, Art and AI);
- машинное зрение (vision);
- машинный перевод (machine translation);
- менеджмент знаний (knowledge management);
- мягкие вычисления (soft computing);
- нейронные сети (neural networks);
- онтологии (ontologies);
- роботика (robotics);
- разведка данных (data mining, knowledge discovery);
- распознавание и синтез речи (speech processing);
- распределенный ИИ (distributed AI);
- решение задач (problem solving);
- понимание естественного языка (natural language processing);
- логический вывод (computer logics);
- рассуждения на прецедентах (case-based reasoning);
- представление знаний (knowledge representation);
- приобретение знаний (knowledge acquisition);
- формальные модели (formal models);

- человеко-машинные системы (human-computer interaction);
- эвристики (heuristics); и многое другое.

Далее в рамках данной книги подробно будет рассмотрено направление инженерии знаний.

1.3. СИСТЕМЫ, ОСНОВАННЫЕ НА ЗНАНИЯХ, И ИНЖЕНЕРИЯ ЗНАНИЙ

Представление знаний и разработка систем, основанных на знаниях (knowledge-based systems), — это одно из основных направлений искусственного интеллекта. Оно связано с разработкой моделей представления знаний, созданием баз знаний, образующих ядро экспертных систем. В последнее время включает в себя модели и методы извлечения и структурирования знаний и сливается с инженерией знаний. Именно исследованиям в этой области посвящена данная книга.

Центральная парадигма интеллектуальных технологий сегодня — это обработка знаний. Системы, ядром которых является база знаний или модель предметной области, описанная на языке сверхвысокого уровня, приближенном к естественному (ЯПЗ — языке представления знаний), называют интеллектуальными. Чаще всего интеллектуальные системы (ИС) применяются для решения сложных задач, где основная трудность решения связана с использованием слабоформализованных знаний специалистов-практиков и где логическая (или смысловая) обработка информации превалирует над вычислительной. Например, поддержка принятия решения в сложных ситуациях, постановка диагноза и рекомендации по методам лечения, анализ визуальной информации, управление диспетчерскими пультами и др.

Направленность этой книги позволит читателю далее подробно познакомиться с основными технологиями разработки систем, основанных на знаниях, — от домашинного этапа приобретения знаний до программных систем реализации.

Практически во всех областях информатики в настоящее время прослеживается тенденция опережения технологических средств разработки по отношению к их теоретическому обоснованию. Такое же положение существует в области интеллектуальных систем, основанных на парадигме обработки знаний (экспертные системы, лингвистические процессоры, обучающие системы и т. п.).

С одной стороны, это объясняется тем, что с первых шагов наука об искусственном интеллекте (ИИ) была направлена на моделирование слабоформализуемых смысловых задач, в которых неприменим традиционный математический аппарат; с другой стороны, ИИ — это ветвь информатики и активно развивается как промышленная индустрия программных средств в условиях жесткой конкуренции, где подчас важнее быстрое внедрение новых идей и подходов, чем их анализ и теоретическая проработка.

Необходимость разработки теоретических основ науки о методах разработки систем, основанных на знаниях — инженерии знаний, — обосновывается в работах Д. А. Поспелова, Э. В. Попова, В. Л. Стефанюка, Р. Шенка, М. Минского — ведущих специалистов в области ИИ в России и за рубежом. Первые шаги в создании методологии (работы Г. С. Осипова, В. Ф. Хорошевского, Wielinga, Slagle, etc.) фактически являлись пионерскими.

Так и появилась инженерия знаний, наука, посвященная теоретическим и практическим проблемам проектирования баз знаний — получению и структурированию знаний специалистов для последующей разработки интеллектуальных систем или систем управления знаниями.

Инженерия знаний охватывает проблематику получения (извлечения или приобретения) знаний, их структурирования и формализации (рис. 1.1).

Поскольку основной проблемой инженерии знаний является извлечение знаний, аналитику необходимо четко понимать природу и особенности, влияющие на эффективность этого процесса. Из множества аспектов проблематики извлечения знаний выделим три основных аспекта этой процедуры [Гаврилова, Червинская, 1992] (рис. 1.2):

- психологический;
- лингвистический;
- методологический (гносеологический).



Рис. 1.1

Структура инженерии знаний

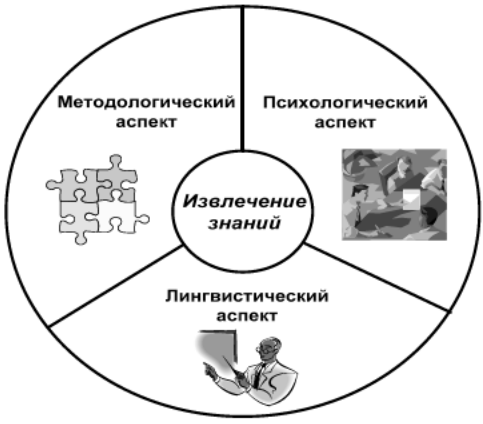


Рис. 1.2

Теоретические аспекты инженерии знаний

Далее они подробно будут рассмотрены в главе 2.

1.4. ОБ ЭКСПЕРТНЫХ СИСТЕМАХ

Экспертные системы (ЭС) — это наиболее распространенный класс ИС, ориентированный на тиражирование опыта высококвалифицированных специалистов в областях, где качество принятия решений традиционно зависит от уровня экспертизы, например, медицина, юриспруденция, геология, экономика, военное дело и др.

Современные ЭС — это сложные программные комплексы, аккумулирующие знания специалистов в конкретных предметных областях и распространяющие этот эмпирический опыт для консультаций менее квалифицированных пользователей. Разработка экспертных систем как активно развивающаяся ветвь информатики направлена на использование компьютеров для обработки информации в тех областях науки и техники, где малопригодны традиционные математические методы моделирования. В этих областях важны смысловая и логическая обработка информации, опыт экспертов.

Фактически интеллектуальные прикладные системы используются сейчас в тысячах приложений. Рынок интеллектуальных систем в 80–90-е гг. рос лавинообразно. В дальнейшем почти 30%-ный прирост дохода сменился более плавным ростом темпов [Попов, 1996; Поспелов, 1997; Хорошевский, 1997; Walker, Miller, 1987; Durkin, 1998; Giarratano, Riley, 2005]. Сейчас годовой объем мирового рынка интеллектуальных систем International Data Corporation (IDC) оценивает в $1,4 млрд (на 2014 г.). При этом примерно 20% затрат приходится на Японию, около 25% — на США и Канаду и более 40% — на европейские страны. А годовой доход от продаж искусственных нейронных сетей составляет $6,3 млрд (по данным компании Computer Associates).

ЭС эффективны лишь в специфических «экспертных» областях, где важен эмпирический опыт специалистов. Например, компания American Express сократила свои потери на 27 млн долл. в год благодаря экспертной системе, определяющей целесообразность выдачи или отказа в кредите той или иной фирме.

Ежегодно крупные фирмы разрабатывают десятки ЭС типа «in-house» для внутреннего пользования. Эти системы интегрируют опыт специалистов компании по ключевым и стратегически важным технологиям.

Первые системы такого рода MYCIN [Shortliffe, 1976] и DENDRAL [Buchanan, Feigenbaum, 1978] появились в США в середине 1970-х гг. В настоящее время в мире насчитывается несколько тысяч промышленных ЭС [Durkin, 1998, Turban & Aronson, 2005], которые дают советы в различных случаях, например: анализ ДНК — FSSI; управление сложными диспетчерскими пультами, например, сети распределения электроэнергии — Alarm Analyser; принятие личных решений — индивидуальные мобильные советчики; помощь при шоппинге — DIY-Expertbots; постановка медицинских диагнозов — PXDES (анализ рентгенограмм), ARAMIS, NEUREX; ответы на муниципальные запросы от жителей — PublicAccess Expert System; поиск неисправностей в электронных приборах, диагностика отказов контрольно-измерительного оборудования — Intelligence Ware, Plant Diagnostics, FOREST; прогноз военных действий — ANALYST, BATTLE; проектирование интегральных микросхем — DAA, NASL, QO; формирование портфеля инвестиций, оценка финансовых рисков — RAD; налогообложение и заполнение налоговых деклараций — Taxadvisor, Investor and Financial Advisor, ExperTax, RUNE; управление компанией — PortBLue; управление перевозками — AIRPLAN и т. д.

Факторы, свидетельствующие о необходимости разработки и внедрения экспертных систем (частично из [Уотермен, 1989]):

- невозможность или неэффективность математического моделирования;
- нехватка специалистов, затрачивающих значительное время на оказание помощи другим;
- выполнение небольшой задачи требует многочисленного коллектива специалистов, поскольку ни один из них не обладает достаточным знанием;
- сниженная производительность, поскольку задача требует полного анализа сложного набора условий, а обычный специалист не в состоянии просмотреть (за отведенное время) все эти условия;
- большое расхождение между решениями самых хороших и самых плохих исполнителей.

Подходящие задачи обычно являются сложными узкоспециализированными; имеют «шум» в данных (неточность, неполноту, некорректность); не зависят в значительной степени от общечеловеческих знаний или соображений здравого смысла.

В качестве рабочего определения экспертной системы примем следующее:

> Экспертные системы (ЭС) — это сложные программные комплексы, аккумулирующие знания специалистов в конкретных предметных областях и тиражирующие этот эмпирический опыт для консультаций менее квалифицированных пользователей.

Обобщенная структура экспертной системы представлена на рисунке 1.3. Реальные ЭС могут иметь более сложную структуру, однако блоки, изображенные на рисунке, непременно присутствуют в любой действительно экспертной системе, поскольку являют собой стандарт de facto на структуру современной ЭС.

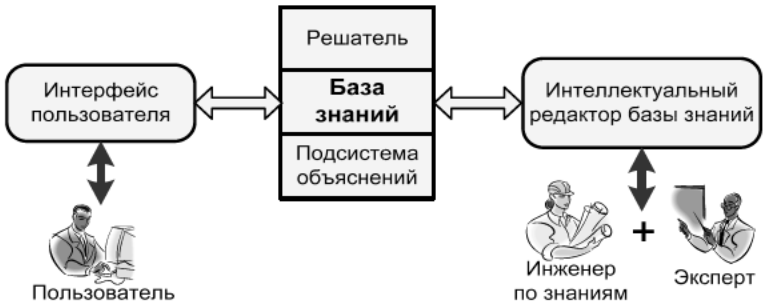


Рис. 1.3

Структура экспертной системы

Советы пользователю генерируются решателем на основании содержимого базы знаний, полученного от эксперта, система может объяснить ход своих «рассуждений» при помощи подсистемы объяснений.

Эксперт — «человек, переставший думать, на том основании, что он ЗНАЕТ» (из фольклора).

Пользователь — специалист предметной области, для которого предназначена система. Обычно его квалификация недостаточно высока и поэтому он нуждается в помощи и поддержке своей деятельности со стороны ЭС.

Инженер по знаниям — специалист по искусственному интеллекту, выступающий в роли промежуточного буфера между экспертом и базой знаний. Синонимы: *когнитолог, инженер-интерпретатор, аналитик.*

В процессе разработки и последующего расширения системы инженер по знаниям и эксперт обычно работают вместе. Инженер по знаниям помогает эксперту

структурировать знания, определять и формализовать понятия и правила, необходимые для решения проблемы.

Основные составляющие ЭС:

- *Интерфейс пользователя* — комплекс программ, реализующих диалог пользователя с ЭС как на стадии ввода информации, так и при получении результатов.
- *База знаний* (БЗ) — ядро ЭС, совокупность знаний предметной области, записанная на машинный носитель в форме, понятной эксперту и пользователю (обычно на некотором языке, приближенном к естественному). Параллельно такому «человеческому» представлению существует БЗ во внутреннем «машинном» представлении.
- *Решатель* — программа, моделирующая ход рассуждений эксперта на основании знаний, имеющихся в БЗ. Синонимы: *дедуктивная машина, машина вывода, блок логического вывода.*
- *Подсистема объяснений* — программа, позволяющая пользователю получить ответы на вопросы: «Как была получена та или иная рекомендация?» и «Почему система приняла такое решение?» Ответ на вопрос «как» — это трассировка процесса получения решения с указанием использованных фрагментов БЗ, т. е. основных шагов цепи умозаключений. Ответ на вопрос «почему» — ссылка на умозаключение, непосредственно предшествовавшее полученному решению, т. е. отход на один шаг назад. Развитые подсистемы объяснений поддерживают и другие типы вопросов.
- *Интеллектуальный редактор БЗ* — программа, представляющая инженеру по знаниям возможность создавать БЗ в диалоговом режиме. Включает в себя систему вложенных меню, шаблонов языка представления знаний, подсказок («help» — режим) и других сервисных средств, облегчающих работу с базой.

Класс ЭС сегодня объединяет несколько тысяч различных программных комплексов, которые можно классифицировать по различным критериям. Полезными могут оказаться классификации, представленные на рисунке 1.4.

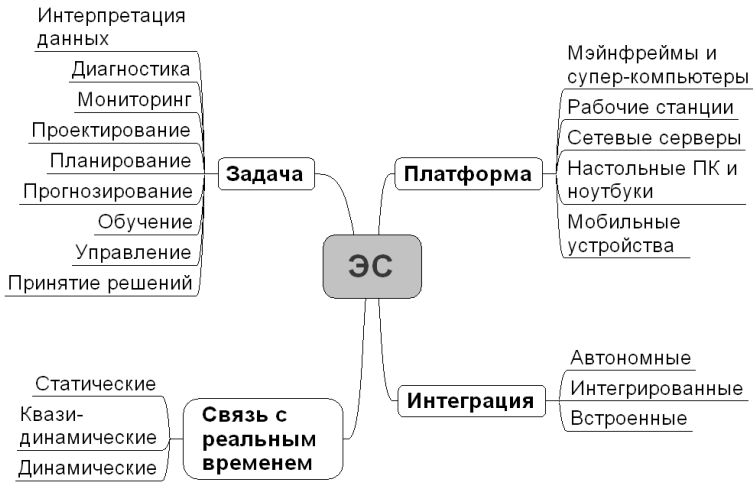


Рис. 1.4

Классификация экспертных систем

Классификация по решаемой задаче

Рассмотрим указанные на рисунке типы задач подробнее.

Интерпретация данных. Под интерпретацией понимается процесс определения смысла данных, результаты которого должны быть согласованными и корректными. Примеры (все примеры далее см.: [Попов и др. (ред.), 1990; Соловьев, Соловьева, 1989; Хейес-Рот и др., 1987; Джарратано, Райли, 2006; Dopico & Sierra, 2009, Melioli et al., 2014; Кобринский 1995, 2005; Иванов, 2013; Bahrammirzaee, 2010; Biemel et. al., 2011; Рыбина, 2010; Загорулько, 2010]):

- интерпретации тестов на наличие аллергии, основанная на технологии микрочипов, Allergenius;
- кредитный скоринг HIS (Hybrid credit ranking intelligent system) — быстрая, точная и устойчивая процедура оценки кредитного риска, имеющая научное обоснование;
- определение основных свойств личности по результатам психодиагностического тестирования в системах АВТАНТЕСТ и МИКРОЛЮШЕР и др.

Диагностика. Под диагностикой понимается процесс соотнесения объекта с некоторым классом объектов и/или обнаружение неисправности в некоторой системе. Неисправность — это отклонение от нормы. Такая трактовка позволяет с единых теоретических позиций рассматривать и неисправность оборудования в технических системах, и заболевания живых организмов, и всевозможные природные аномалии. Примеры:

- диагностика и терапия сужения коронарных сосудов — ANGY;
- диагностика заболеваний риса — MANAGE;
- диагностика, классификация и коррекция терапии острых расстройств кровообращения у детей — «Айболит».

Мониторинг. Основная задача мониторинга — непрерывная интерпретация данных в реальном масштабе времени и сигнализация о выходе тех или иных параметров за допустимые пределы. Главные проблемы — «пропуск» тревожной ситуации и инверсная задача «ложного» срабатывания. Примеры:

- контроль за работой электростанций — СПРИНТ;
- помощь диспетчерам атомного реактора — REACTOR;
- контроль аварийных датчиков на химическом заводе — FALCON;
- оценка действий космонавта в процессе тренировки по ручной стыковке транспортного грузового корабля (ТКГ) и Международной космической станции (МКС);
- помощь инвесторам в выборе акций для составления инвестиционного портфолио — PORSEL (PORtfolio SELection system) и др.

Проектирование. Проектирование состоит в подготовке спецификаций на создание «объектов» с заранее определенными свойствами. Под спецификацией понимается весь набор необходимых документов — чертеж, пояснительная записка и т. д. Основные проблемы здесь — получение четкого структурного описания знаний об объекте и проблема «следа». Примеры:

- проектирование конфигураций ЭВМ VAX — 11/780 в системе XCON (или R1);
- проектирование больших интегральных схем CADHELP (http://acl.ldc.upenn.edu/J/J81/J81-4011.pdf);
- синтез электрических цепей — SYN;
- система управления контроля доступа, концептуальная модель, позволяющая проектировать сценарии медицинских запросов доступа к данным — SitBAC (Situation-Based Access Control) и др.

Прогнозирование. Прогнозирование позволяет предсказывать последствия некоторых событий или явлений на основании анализа имеющихся данных. Прогнозирующие системы логически выводят вероятные следствия из заданных ситуаций. Примеры:

- предсказание погоды — система WILLARD;
- оценки будущего урожая — PLANT;
- прогнозы в экономике — ECON и др.

Планирование. Под планированием понимается поиск планов действий, относящихся к объектам, способным выполнять некоторые функции. В таких ЭС используются модели поведения реальных объектов с тем, чтобы логически вывести последствия планируемой деятельности. Примеры:

- планирование поведения робота — STRIPS;
- планирование промышленных заказов — ISIS;
- планирование туристической деятельности — SigTur/E-Destination;
- планирование экспериментов по клонированию генов в молекулярной генетике — MOLGEN.

Обучение. Под обучением понимается использование компьютера для обучения какой-то дисциплине или предмету. Системы обучения диагностируют ошибки при изучении какой-либо дисциплины с помощью компьютера и подсказывают правильные решения. Они аккумулируют знания о гипотетическом «ученике» и его характерных ошибках, способны диагностировать слабости в познаниях обучаемых и находить соответствующие средства для их ликвидации. Примеры:

- обучение языку программирования ЛИСП в системе «Учитель ЛИСПа»;
- система PROUST — обучение языку Паскаль;
- среда разработки широкого класса обучающих ЭС — АТ-ТЕХНОЛОГИЯ и др.

Управление. Под управлением понимается функция организованной системы, поддерживающая определенный режим деятельности. Такого рода ЭС осуществляют управление поведением сложных систем в соответствии с заданными спецификациями. Примеры:

- помощь в управлении газовой котельной — GAS;
- управление сельскохозяйственным комплексом — CALEX;
- управление системой календарного планирования Project Assistant;
- управление процессом очистки сточных вод (ОСВ) от нефтяного загрязнения на биологических очистных сооружениях.

Поддержка принятия решений. Поддержка принятия решения — это совокупность процедур, обеспечивающая лицо, принимающее решение, необходимой информацией и рекомендациями для облегчения процесса принятия решения. Примеры:

- выбор стратегии выхода фирмы из кризисной ситуации — CRYSIS;
- советы по борьбе с вредителями урожая — LIMEX;
- поддержка принятия решений для оптимизации работы с валютными опционами — Nereid;
- помощь в выборе страховой компании или инвестора — CHOICE;
- советы по стратегическому управлению и планированию деятельности предприятия трубопроводного транспорта нефтепродуктов, внедренная в ОАО «Уралтранснефтепродукт».

Классификация по связи с реальным временем

Статические ЭС разрабатываются в предметных областях, в которых база знаний и интерпретируемые данные не меняются во времени. Они стабильны. Пример: диагностика неисправностей в автомобиле.

Квазидинамические ЭС интерпретируют ситуацию, которая меняется с некоторым фиксированным интервалом времени. Пример: микробиологические ЭС, в которых снимаются лабораторные измерения с технологического процесса один раз в 4–5 часов (производство лизина, например) и анализируется динамика полученных показателей по отношению к предыдущему измерению.

Динамические ЭС работают в сопряжении с датчиками объектов в режиме реального времени с непрерывной интерпретацией поступающих в систему данных. Пример: управление гибкими производственными комплексами, мониторинг в реанимационных палатах и т. д. Пример: программный инструментарий для разработки динамических систем G2 [Попов, 1996; Stephanopoulos, Han, 1996].

Классификация по степени интеграции с другими программами

Автономные ЭС работают непосредственно в режиме консультаций с пользователем для специфически «экспертных» задач, для решения которых не требуется привлекать традиционные методы обработки данных (расчеты, моделирование и т. д.).

Гибридные ЭС представляют программный комплекс, агрегирующий стандартные пакеты прикладных программ (ППП) (например, математическую статистику, линейное программирование или системы управления базами данных) и средства манипулирования знаниями. Это может быть интеллектуальная надстройка над ППП или интегрированная среда для решения сложной задачи с элементами экспертных знаний.

О коллективе разработчиков экспертной системы

Над созданием ЭС трудится *коллектив разработчиков* (КР), под которым будем понимать всю группу специалистов, ответственных за создание ЭС.

В состав КР входят по крайней мере три человека — пользователь, эксперт и инженер по знаниям. Обычно в КР включают и программиста. Таким образом, минимальный состав КР включает четыре человека; реально же он разрастается до 8–10 человек. Часто в этот коллектив входят менеджер и его технический помощник.

Если использовать аналогии из близких областей, то КР более всего схож с группой администратора базы данных при построении интегрированных информационных систем или бригадой программистов, разрабатывающих сложный программный комплекс. При отсутствии профессионального менеджера руководителем КР, участвующим во всех стадиях разработки, является аналитик или инженер по знаниям, поэтому к его квалификации предъявляются самые высокие требования.

Для обеспечения эффективности сотрудничества любой творческой группы, необходимо возникновение атмосферы взаимопонимания и доверия, которое, в свою очередь, обусловлено психологической совместимостью членов группы; следовательно, при формировании КР должны учитываться психологические свойства участников.

Ниже приведены два аспекта характеристик членов КР: А — психофизиологический, Б — профессиональный.

Пользователь

А. К пользователю предъявляются самые слабые требования, поскольку его не выбирают. Желательные качества: а) дружелюбие; б) умение объяснять свои требования к системе; в) отсутствие психологического барьера к применению вычислительной техники; г) интерес к новому. От пользователя зависит, будет ли

применяться разработанная ЭС. Замечено, что наиболее ярко качества в) и г) проявляются в молодом возрасте, поэтому иногда такие пользователи охотнее применяют ЭС, при этом у них отсутствует комплекс неполноценности оттого, что компьютер им что-то подсказывает.

Б. Необходимо, чтобы пользователь имел некоторый базовый уровень квалификации, который позволит ему правильно истолковать рекомендации ЭС.

Эксперт

А. Эксперт в конечном счете задает уровень компетенции базы знаний, являясь важной фигурой в группе КР. Желательные качества: а) доброжелательность; б) готовность поделиться своим опытом; в) умение объяснить (педагогические навыки); г) интерес (моральный, а лучше еще и материальный) к успешности разработки. Возраст эксперта может оказаться зрелым, что необходимо учитывать всем членам группы.

Б. Помимо безусловно высокого профессионализма в выбранной предметной области, желательно знакомство эксперта с популярной литературой по искусственному интеллекту и экспертным системам для более эффективного процесса извлечения знаний.

Программист

А. Известно, что программисты обладают самой низкой потребностью в общении среди представителей разных профессий. Однако при разработке ЭС необходим тесный контакт членов группы, поэтому желательны следующие качества: а) общительность; б) способность отказаться от традиционных навыков и освоить новые методы; в) интерес к разработке.

Б. Поскольку современные ЭС — сложнейшие и дорогостоящие программные комплексы, программисты в КР должны иметь опыт и навыки разработки программ. Обязательно знакомство с основными структурами представления знаний и механизмами вывода, состоянием отечественного и мирового рынка программных продуктов для разработки ЭС и диалоговых интерфейсов.

Инженер по знаниям

Растет фактический спрос на уникальных специалистов по работе со знаниями — аналитиков. На современном рынке высоких информационных технологий потребность в аналитиках многократно превышает спрос на программистов. Этих специалистов называют по-разному — системные аналитики, постановщики задач, инженеры по знаниям, инженеры-когнитологи (по-английски — knowledge engineers). Близкие понятия и синонимы: постановщик задач / CKO (chief knowledge officer) — директор по управлению знаниями / менеджер знаний / функциональный аналитик или бизнес-аналитик.

А. Существуют такие профессии и виды деятельности, для которых природные качества личности (направленность, способности, темперамент) могут иметь характер абсолютного показания или противопоказания к занятиям. По-видимому, инженерия знаний принадлежит к таким профессиям. По различным оценкам, это одна из самых малочисленных, высокооплачиваемых и дефицитных в мире специальностей. Попытаемся дать наброски к портрету инженера по знаниям (без претензии на полноту и точность определений).

Пол. Психологи утверждают, что мужчины более склонны к широкому охвату явлений и в среднем у них выше аналитичность, чрезвычайно полезная инженеру по знаниям, которому надо иметь развитое логическое мышление и умение оперировать сложными формальными структурами. С другой стороны, известно, что у женщин выше наблюдательность к отдельным деталям объек-

тов. Так что пол не является окончательным показанием или противопоказанием к данной профессии.

Стиль общения. Инженер по знаниям «задает тон» в общении с экспертом, он ведет диалог, и от него, в конечном счете, зависит продуктивность. Можно выделить два стиля общения: деловой или жесткий и дружеский или мягкий, деликатный. Дружеский будет заведомо более успешным, так как снижает «эффект фасада» у эксперта, раскрепощает его. Деликатность, внимательность, интеллигентность, ненавязчивость, скромность, умение слушать и задавать вопросы, хорошая коммуникабельность и в то же время уверенность в себе — вот рекомендуемый стиль общения.

Портрет инженера по знаниям можно было бы дополнить другими характеристиками — широтой взглядов и интересов, артистичностью, чувством юмора, обаянием и т. д.

Б. При определении профессиональных требований к аналитику, следует учитывать, что ему необходимы различные навыки и умения для грамотного и эффективного проведения процессов извлечения, концептуализации и формализации знаний.

Успешность выбора и подготовки коллектива разработчиков ЭС определяет эффективность и продолжительность всего процесса разработки.

1.5. РАЗРАБОТКА ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

За разработку экспертных систем целесообразно браться компании, в которой накоплен опыт автоматизации рутинных процедур обработки информации, таких как, например, формирование корпоративных информационных систем или организация сложных расчетов. В процессе решения таких задач, во-первых, происходит подготовка аналитиков, во-вторых, появляется возможность отделить от экспертных систем неэкспертные задачи.

Процесс разработки промышленной экспертной системы, опираясь на традиционные технологии [Николов и др., 1990; Tuthill, 1990; Waterman, 2004], можно разделить на шесть более или менее независимых этапов (рис. 1.5), практически независимых от предметной области.

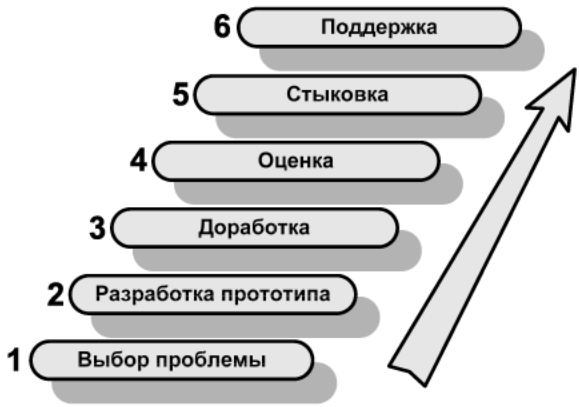


Рис. 1.5
Этапы разработки ЭС

Последовательность этапов дана только с целью получения общего представления о процессе создания идеального проекта. Конечно, последовательность эта не вполне фиксированная. В действительности каждый последующий этап разработки способен принести новые идеи, которые могут повлиять на предыдущие решения и даже привести к их переработке. Именно поэтому многие специалисты по информатике весьма критично относятся к методологии экспертных систем. Они считают, что на разработку таких систем расходы и слишком велики, а полученные в результате программы требуют значительных вычислительных ресурсов.

Правильный выбор проблемы представляет критическую часть разработки в целом. Если выбрать неподходящую проблему, можно увязнуть в «болоте» задач, которые никто не знает, как решать. Неподходящая проблема может также привести к разработке экспертной системы, создание которой будет стоить больше, чем предполагаемая экономия от ее внедрения. Этот этап традиционно включает [Николов и др., 1990; Adeli, 2003]:

- определение проблемной области и задачи;
- нахождение эксперта, желающего сотрудничать при решении проблемы, и назначение коллектива разработчиков;
- подготовку подробного плана разработки.

Экспертная система не заменяет «классическую» автоматизацию (базы данных, статистика, документооборот и прочее), а дополняют ее.

Экспертные системы разрабатываются путем получения специфических знаний от эксперта и ввода их в систему. Некоторые системы могут содержать стратегии одного индивида, следовательно, найти подходящего эксперта — это ключевой шаг в создании экспертных систем.

Прежде всего, инженер по знаниям должен убедиться, что на перечисленные ниже вопросы будет получен утвердительный ответ:

- данная задача может быть решена с помощью экспертной системы;
- экспертную систему можно создать предлагаемыми на рынке средствами;
- имеется подходящий эксперт;
- предложенные критерии производительности являются разумными;
- затраты и срок их окупаемости приемлемы для заказчика.

Сначала разрабатывается прототип (усеченная ЭС), при неудовлетворительном функционировании прототипа эксперт и инженер по знаниям имеют возможность оценить, что именно будет включено в разработку окончательного варианта системы.

Если первоначально выбранные объекты или свойства оказываются неподходящими, их следует изменить. Можно сделать оценку общего числа эвристических правил, необходимых для создания окончательного варианта экспертной системы. Иногда [Хювянен, Сеппянен, 1991] при разработке промышленной и/или коммерческой системы выделяют дополнительные этапы для перехода: *демонстрационный прототип — действующий прототип — промышленная система — коммерческая система* (табл. 1.1).

Однако чаще реализуется плавный переход от демонстрационного прототипа к промышленной системе, при этом, если программный инструментарий был выбран удачно, не обязательно даже переписывать окончательный вариант другими программными средствами.

Понятие же коммерческой системы в нашей стране входит в понятие промышленный программный продукт, или промышленной ЭС в этой работе.

Таблица 1.1

Переход от прототипа к промышленной экспертной системе

Этап разработки	Функция
Демонстрационный прототип ЭС	Система решает часть задач, демонстрируя жизнеспособность подхода (несколько десятков правил или понятий)
Исследовательский прототип ЭС	Система решает большинство задач, но неустойчива в работе и не полностью проверена (несколько сотен правил или понятий)
Действующий прототип ЭС	Система надежно решает все задачи на реальных примерах, но для сложной задачи требует много времени и памяти
Промышленная система	Система обеспечивает высокое качество решений при минимизации требуемого времени и памяти; переписывается с использованием более эффективных средств представления знаний
Коммерческая система	Промышленная система, пригодная к продаже, т. е. хорошо документирована и снабжена сервисом

Основная работа при доработке системы заключается в существенном расширении базы знаний, т. е. добавлении большого числа дополнительных правил, фреймов, узлов семантической сети или других элементов знаний. После установления основной структуры ЭС знаний инженер по знаниям приступает к разработке и адаптации интерфейсов, с помощью которых система будет общаться с пользователем и экспертом.

На этом этапе разработки большинство экспертов узнают достаточно о вводе правил и могут сами вводить в систему новые правила. Таким образом, начинается процесс, во время которого инженер по знаниям передает право собственности и контроля за системой эксперту для уточнения, детальной разработки и обслуживания.

После завершения этапа разработки промышленной экспертной системы необходимо провести ее тестирование в отношении критериев эффективности. Оценку можно проводить исходя из различных критериев, сгруппированных следующим образом:

- критерии пользователей (понятность и «прозрачность» работы системы, удобство интерфейсов и др.);
- критерии приглашенных экспертов (оценка советов-решений, предлагаемых системой, сравнение ее с собственными решениями, оценка подсистемы объяснений и др.);
- критерии коллектива разработчиков (эффективность реализации, производительность, время отклика, дизайн, широта охвата предметной области, непротиворечивость БЗ, количество тупиковых ситуаций, когда система не может принять решение, анализ чувствительности программы к незначительным изменениям в представлении знаний, весовых коэффициентах, применяемых в механизмах логического вывода, данных и т. п.).

Далее осуществляется стыковка экспертной системы с другими программными средствами в среде, в которой она будет работать, и обучение людей, которых она будет обслуживать.

Технология быстрого прототипирования

Важнейшую роль при разработке ЭС играет технология прототипирования. *Прототипная система* является усеченной версией экспертной системы, спроектированной для проверки правильности кодирования фактов, связей и стратегий рассуждения эксперта.

Объем прототипа — несколько десятков правил, фреймов или примеров. На рисунке 1.6 изображены шесть стадий разработки прототипа и минимальный коллектив разработчиков, занятых на каждой из стадий (пять стадий заимствованы из [Хейес-Рот и др., 1987; Джексон, 2001]). Приведем краткую характеристику каждой из стадий, хотя эта схема представляет грубое приближение к сложному итеративному процессу построения ЭС.

Хотя любое теоретическое разделение бывает часто условным, осознание коллективом разработчиков четких задач каждой стадии представляется целесообразным. Сроки приведены условно, так как зависят от квалификации специалистов и особенностей задачи.

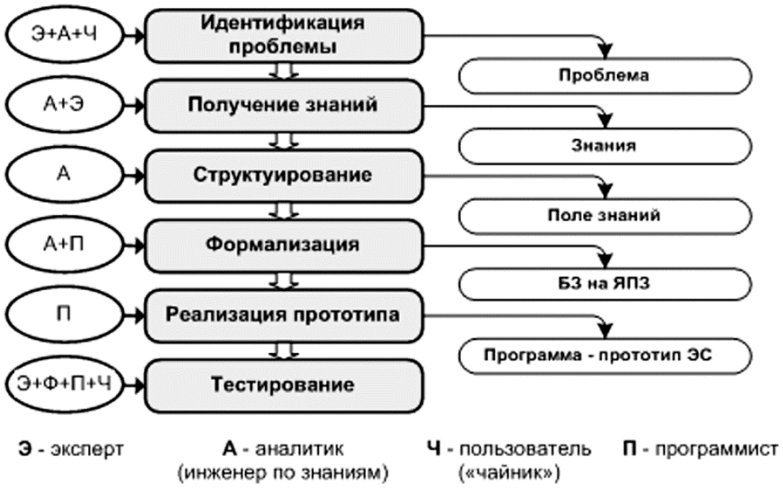


Рис. 1.6

Стадии разработки прототипа ЭС

Идентификация проблемы. На этом этапе происходит знакомство и обучение членов коллектива разработчиков, а также создание неформальной формулировки проблемы. Средняя продолжительность 1–2 недели. Уточняется задача, планируется ход разработки прототипа экспертной системы. Также определяются:

- необходимые ресурсы (время, люди, компьютеры и т. д.);
- источники знаний (книги, дополнительные эксперты, методики);
- имеющиеся аналогичные экспертные системы;
- цели (распространение опыта, автоматизация рутинных действий и др.);
- классы решаемых задач и т. д.

Извлечение знаний включает получение инженером по знаниям наиболее полного из возможных представления о предметной области и способах принятия решения в ней. Средняя продолжительность — 1–3 месяца. Происходит перенос компетентности экспертов на инженеров по знаниям с использованием различных методов (см. главу 2 подробнее).

Структурирование, или концептуализация, знаний. Производится разработка неформального описания знаний о предметной области в виде графа, таблицы, диаграммы или текста, которое отражает основные концепции и взаимосвязи между понятиями предметной области. Такое описание называется *полем знаний*. Средняя продолжительность этапа — 2–4 недели. Подробно стадия структурирования описана далее.

Формализация, или кодификация, знаний. Строится формализованное представление концепций предметной области на основе выбранного языка представления знаний (ЯПЗ). Выполняется разработка базы знаний на языке представления знаний, который, с одной стороны, соответствует структуре поля знаний, а, с другой, позволяет реализовать прототип системы на следующей стадии программной реализации. Средняя продолжительность — 1–2 месяца. Подробно языки представления знаний описаны в главе 4.

Реализация. Предварительный подход к программной реализации задачи определяется исходя из характеристик задачи и ресурсов, выделенных на ее решение. Средняя продолжительность — 1–2 месяца. Чаще всего первый прототип отбрасывается на этапе реализации действующей ЭС. Создается прототип экспертной системы, включающий базу знаний и остальные блоки, при помощи одного из следующих способов:

- программирование на традиционных языках типа PYTHON, C#, Java и др.;
- программирование на специализированных языках, применяемых в задачах искусственного интеллекта: LISP [Хювянен, Сеппянен, 1991], ПРОЛОГ [Адаменко, 2002], SMALLTALK [Справочник по ИИ, 1990] и др.;
- использование инструментальных средств разработки ЭС типа CLIPS [Джарратано, Райли, 2006; Частиков и др., 2003], ПИЭС [Хорошевский, 1993]; Simer + MIR [Осипов, 1997], G2 [Попов, Фоминых, Кисель, 1996];
- использование «пустых» ЭС или «оболочек» типа EXSYS, KAPPA [Гаврилова, 2004; Муромцев, 2005] и др.

Средняя продолжительность — 1–2 месяца. Более подробно эти вопросы рассматриваются далее.

Перед разработчиками ЭС часто остро стоит вопрос о выборе адекватного программного инструментария разработки. Быстро изменяющийся рынок средств разработки предлагает десятки программных систем, однако при выборе надо принимать во внимание целый ряд как объективных, так и субъективных факторов. Единственным непреложным правилом является следующее:

«Выбор системы реализации должен осуществляться ПОСЛЕ этапа извлечения и структурирования знаний».

Значимым фактором при выборе средства разработки является структура знаний предметной области, заранее предсказать которую зачастую сложно. Основные факторы, влияющие на выбор программного инструмента:

- особенности предметной области и наличие квалифицированной команды разработчиков;
- бюджет проекта и необходимость стыковки с другими программными продуктами;
- количество пользователей будущей системы и их квалификация.

Приведем некоторые примеры инструментов разработки, согласно приведенной классификации.

Языки программирования. Основные языки реализации были перечислены выше. Но фаворитом в Европе и Японии остается PROLOG, а в США — LISP.

Оболочки. Особенной популярностью пользуются «пустые» ЭС или оболочки из-за простоты и скорости разработки БЗ.

Свободно распространяемые и дешевые «оболочки»:

- E2glite (http://www.expertise2go.com/webesie/e2gdoc/);
- JESS (http://herzberg.ca.sandia.gov/jess/download.shtml);
- JEOPS (http://www.cin.ufpe.br/~jeops/);
- TyRuBa (http://tyruba.sourceforge.net/).

Некоторые коммерческие «оболочки» представлены в таблице 1.2.

Таблица 1.2

Примеры коммерческих оболочек ЭС

Имя	Разработчик	Адрес
ACQUIRE SDK	Acquired Intelligence Inc, Канада, 2006	http://www.aiinc.ca/
ANGOSS KNOWLEDGE SEEKER	Angoss Software Corporation	http://www.angoss.com/predictive-analytics-software/software/knowledgeseeker/
EZ-Xpert	AI Developers Inc, США, 2006	http://www.ez-xpert.com/
Exsys 2006	Exsys, США, 2006	http://www.exsys.com/
FuzzyShell	Purdue Robot Vision Lab, США, 2006	http://cobweb.ecn.purdue.edu/RVL/Projects/Fuzzy
ECLIPSE	The Eclipse Foundation	http://www.haley.com https://eclipse.org/

Программные среды или инструментарии. Программные инструментальные системы (например, CLIPS [CLIPS 2006] (C Language Integrated Production System)) включают в себя специальные языки (например, объектно-ориентированный язык COOL) для написания экспертных систем. Далее с помощью этого языка описывается база знаний.

Отдельно следует упомянуть ЭС реального времени, для разработки которых используются более сложные и дорогие инструменты. Например, G2 — программный продукт, предназначенный для быстрой разработки экспертных систем реального времени, которые отслеживают и обнаруживают неполадки в сложных програмно-аппаратных комплексах [Nan, Khan & Iqbal, 2008]. Первая версия инструментальной системы G2 [Moore, Rosenhof & Stanley, 1989] была разработана в 1988 г. компанией Gensym. Это мощное инструментальное средство, обладающее всеми свойствами, необходимыми для создания экспертных систем реального времени [Попов, 1996].

Тестирование. Этап включает выявление ошибок в подходе и реализации прототипа и выработку рекомендаций по доводке системы до промышленного варианта. Оценивается и проверяется работа прототипа с целью приведения в соответствие с реальными запросами пользователей. Средняя продолжительность — 1–2 недели. Проверяются в основном следующие свойства прототипа:

- удобство и адекватность интерфейсов ввода-вывода (характер вопросов в диалоге, связность выводимого текста результата и др.);
- эффективность стратегии управления (порядок перебора, использование нечеткого вывода и др.);
- корректность базы знаний (полнота и непротиворечивость правил).

Средняя продолжительность — 1–2 недели.

1.6. ПРИКЛАДНЫЕ ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ

Наиболее популярные приложения ИС отражены на рисунке 1.7 [Durkin, 1998; David, Krivine & Simmons, 2012].

Классическим примером экспертной системы является система FocalPoint (компания TriPath Imaging) [ES 2006]. Ее задачей является определение наличия раковых клеток по взятой у пациента пробе. Вначале создатели системы

сформировали при помощи экспертов набор основных признаков, по которым отличаются отклонившиеся от нормы клетки. Далее система проходит обучение, анализируя пробы, для которых уже был поставлен диагноз. Однако после того как систему выпускают, функции обучения в ней блокируются, поскольку это может привести к тому, что неопытный врач может ухудшить качество системы, что является неприемлемым. Система ежегодно производит анализ 5 млн изображений клеток, что составляет 10% от всех подобных анализов в США.

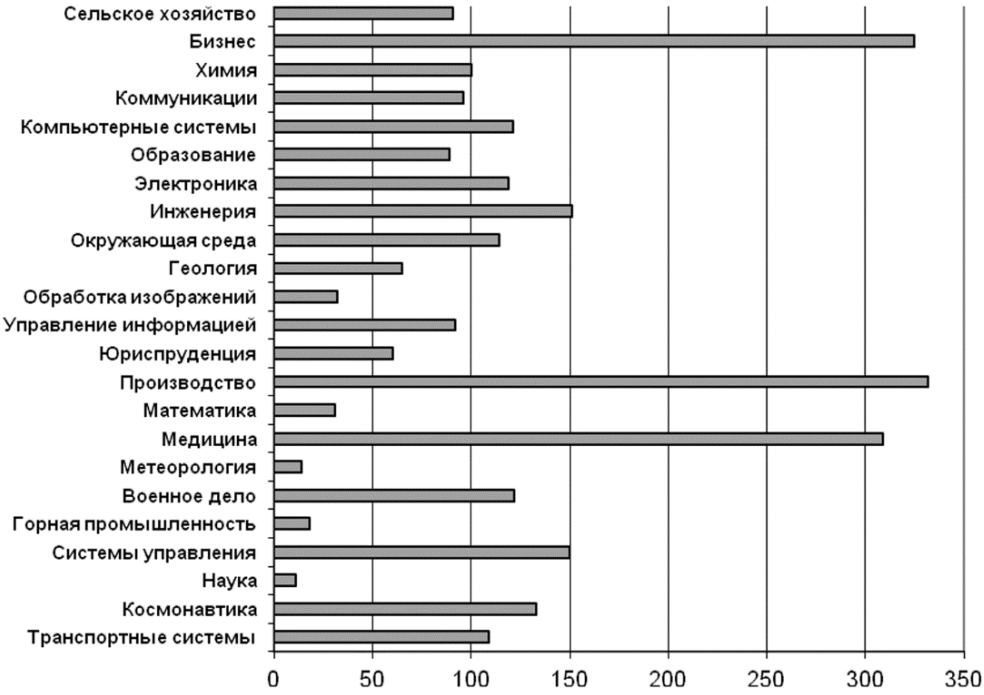


Рис. 1.7

Основные приложения ИС

Из рисунка 1.7 видно преобладание производственных ЭС и бизнес-приложений. При этом отчетливый рост наблюдается в области финансовых ЭС.

Сложно переоценить значение финансовых экспертных систем для бизнеса. Финансовые ЭС системы активно используются в коммерческих предприятиях — для анализа собственного экономического, финансового, бухгалтерского, инвестиционного, инновационного состояний, для анализа рентабельности, составления отчетов перед собственниками предприятия и держателями акций либо для тактического и стратегического планирования производства и сбыта.

Главное отличие ИС и ЭС от других программных средств — это наличие базы знаний (БЗ), где знания хранятся в форме, понятной специалистам предметной области, и могут быть изменены и дополнены также в понятной форме. Это и есть языки представления знаний — ЯПЗ.

До последнего времени именно различные ЯПЗ были центральной проблемой при разработке ЭС. Сейчас существуют десятки языков или моделей представления знаний (см. далее главу 4). Наибольшее распространение получили следующие модели:

1) продукционные правила;
2) семантические сети;
3) фреймы FRL;
4) логика первого порядка;
5) объектно-ориентированные языки (например, CLOS).

Для перечисленных выше моделей существует соответствующая математическая нотация, разработаны системы программирования, реализующие эти ЯПЗ, и имеется большое количество реальных коммерческих ЭС.

Сейчас в России разработок в области ЭС ведется немного, но имеется определенный интерес среди наиболее «продвинутых» и квалифицированных слоев специалистов — финансистов, топ-менеджеров, преподавателей, инженеров, медиков, психологов, программистов, лингвистов. В последние годы этот интерес имеет пока достаточно слабое материальное подкрепление — явная нехватка грамотных специалистов, ограниченное финансирование исследований в этой области, слабый отечественный рынок программных продуктов для разработки ЭС.

Поэтому продолжается спекуляция терминологии, появляется возможность распространения «подделок» под экспертные системы в виде многочисленных диалоговых систем и интерактивных пакетов прикладных программ, которые дискредитируют в глазах пользователей это перспективное направление.

Наибольшие трудности в разработке ЭС вызывает сегодня не процесс машинной реализации систем, а домашинный этап анализа знаний и проектирования базы знаний. Этим занимается специальная наука — инженерия знаний [Абдикеев, Киселев, 2011; Гаврилова, Червинская, 1992; Гаврилова, Хорошевский, 2000; Adeli, 1994; Scott, Clayton, Gibson, 1991].

Современному состоянию этой науки и посвящены последующие главы этой книги.

Глава 2

ПОЛУЧЕНИЕ ЗНАНИЙ

2.1. ЗНАНИЯ И ДАННЫЕ

Междисциплинарный термин «знание» чаще всего трактуется как «проверенный практикой результат познания действительности, верное ее отражение в сознании человека». Для практической работы со знаниями (производственными, учебными, корпоративными) необходимо создать некоторый концептуальный терминологический каркас, в рамках которого эта работа становится возможной.

При работе с информационными и/или интеллектуальными системами традиционно возникает вопрос о том, как соотносятся понятия «информация», «данные», «контент» и «знания».

Оставив за рамками этой книги философские и сугубо теоретические определения, можно предложить несколько рабочих определений, благодаря которым становится возможной практическая работа с любой информацией.

Понятие «информация» будем понимать максимально широко. По Шэннону [Шэннон, Уивер, 1963], «это все, что снимает неопределенность по поводу исхода того или иного события». Обобщая множество определений, можно сказать, что *информация* — это набор сведений, которые воспринимаются и обрабатываются живым существом или устройством и передаются с помощью знаков (символов, звуков, жестов).

Современное представление об информации позволяет ее разделить на три группы:

- контент (слабоструктурированная гетерогенная смесь — письма, заметки, фото, аудио- и видеофайлы, интервью, и т. д.);
- данные (упорядоченные таблицы, каталоги, и пр.);
- знания (осмысленная информация).

В рамках этой книги нас интересуют все три составляющие, но в данной главе мы в большей степени будем работать с данными и знаниями. Что же такое знания и чем они отличаются от обычных данных, десятилетиями обрабатываемых компьютером?

Данные — это отдельные упорядоченные факты, характеризующие объекты, процессы и явления предметной области, а также их свойства.

Знания связаны с данными, основываются на них, но представляют результат мыслительной деятельности человека, обобщая его опыт, полученный в ходе выполнения какой-либо практической деятельности. Часто они получаются эмпирическим путем.

Знания — это закономерности предметной области (принципы, связи, законы), полученные в результате практической деятельности и профессионального опыта, позволяющие специалистам ставить и решать задачи в этой области.

Часто используются такие определения знаний:

Знания — это хорошо структурированные данные, или данные о данных, или метаданные.

Существует множество способов определять понятия. Один из широко применяемых способов основан на идее интенсионала. *Интенсионал* понятия — это определение его через соотнесение с понятием более высокого уровня абстракции с указанием специфических свойств (рис. 2.1). Интенсионалы формулируют зна-

ния об объектах. Другой способ определяет понятие через соотнесение с понятиями более низкого уровня абстракции или перечисление фактов, относящихся к пределяемому объекту. Это есть определение через данные, или *экстенсионал понятия.*

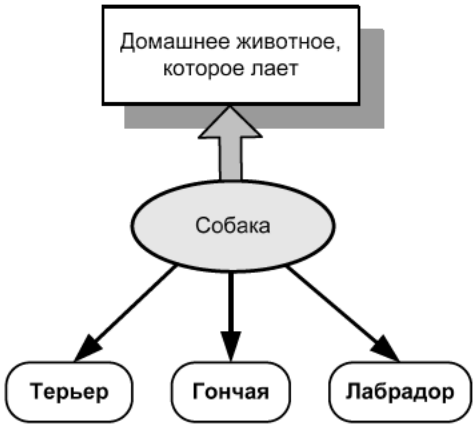


Рис. 2.1
Интенсионал и экстенсионал понятия

Пример 2.1

Понятие «птица». Его интенсионал: «*Летающее животное с крыльями и клювом*». Экстенсионал этого понятия: «*Птица — это голубь, воробей, ворона, чайка и др.*».

Для хранения данных используются базы данных (для них характерны большой объем и относительно небольшая удельная стоимость информации), для хранения знаний — базы знаний (небольшого объема, но исключительно дорогие информационные массивы). *База знаний* — основа любой интеллектуальной системы.

Знания могут быть классифицированы по различным признакам (рис. 2.2).

Пояснений требует только классификация по глубине.

Глубинные — абстракции, аналогии, схемы, отображающие структуру и природу процессов, протекающих в предметной области. Эти знания объясняют явления и могут использоваться для прогнозирования поведения объектов.

Поверхностные — знания о видимых взаимосвязях между отдельными событиями и фактами в предметной области.

Пример 2.2

Поверхностные знания:

- «Если нажать на кнопку звонка, раздастся звук»;
- «Если болит голова, то принять аспирин»;
- «Если неприятность может случиться, она случается» (Закон Мерфи).

Глубинные знания:

- принципиальная электрическая схема звонка и проводки;
- знания физиологов и врачей высокой квалификации о причинах, видах головных болей и методах их лечения.

Современные интеллектуальные системы работают, в основном, с поверхностными, профессиональными, явными и коллективными знаниями. Это связано

с тем, что на данный момент нет универсальных методик, позволяющих выявлять глубинные и неявные структуры знаний и работать с ними.



Рис. 2.2

Классификация видов и формы знаний

2.2. ИСТОЧНИКИ И СПОСОБЫ ПОЛУЧЕНИЯ ЗНАНИЙ

Под *полем знаний* [Гаврилова, 1984] будем понимать некоторое неформальное описание основных понятий предметной области и связей между ними. При формировании поля знаний ключевым вопросом является сам процесс получения знаний, когда происходит перенос компетентности эксперта на инженеров по знаниям. Этот процесс в специальной литературе получил несколько названий:

- приобретение (acquisition);
- добыча (knowledge capture);
- извлечение (elicitation);
- выявление (discovery);
- формирование знаний и др.

Авторы склонны использовать термин «извлечение» как более точно выражающий смысл процедуры переноса компетентности эксперта через инженера по знаниям в базу знаний.

Извлечение знаний (knowledge elicitation) — это процесс взаимодействия эксперта с источником знаний, в результате которого становятся явными процесс рассуждений специалистов при принятии решения и структура их представлений о предметной области.

В настоящее время большинство разработчиков интеллектуальных систем отмечают, что процесс *извлечения знаний* остается самым «узким» местом при построении промышленных ЭС. При этом им приходится практически самостоятельно разрабатывать методы извлечения, сталкиваясь со следующими трудностями:

- организационные неувязки;
- неудачный метод извлечения, не совпадающий со структурой знаний в данной области;
- неадекватная модель (язык) для представления знаний.

Можно добавить к этому [Гаврилова, Червинская, 1992]:

- неумение наладить контакт с экспертом;
- терминологический разнобой;
- отсутствие целостной системы знаний в результате извлечения только «фрагментов»;
- упрощение «картины мира» эксперта и др.

Процесс извлечения знаний — это длительная и трудоемкая процедура, в которой инженеру по знаниям, вооруженному специальными знаниями по когнитивной психологии, системному анализу, математической логике и пр., необходимо воссоздать модель предметной области, которой пользуются эксперты для принятия решения. Часто начинающие разработчики, желая упростить эту процедуру, пытаются подменить инженера по знаниям самим экспертом. По многим причинам это нежелательно.

Во-первых, большая часть знаний эксперта — это результат многочисленных наслоений, ступеней опыта. И зная, что из A следует B, эксперт часто не отдает себе отчета, что цепочка его рассуждений была гораздо длиннее, например:

$$A \rightarrow D \rightarrow C \rightarrow B, \quad \text{или} \quad A \rightarrow Q \rightarrow R \rightarrow B.$$

Во-вторых, как утверждал еще Платон, мышление диалогично. И поэтому диалог инженера по знаниям и эксперта — наиболее естественная форма изучения лабиринтов памяти эксперта, в которых хранятся знания, частично носящие невербальный характер, т. е. выраженные не в форме слов, а, например, в форме наглядных образов. И именно в процессе объяснения инженеру по знаниям эксперт на эти размытые ассоциативные образы надевает четкие словесные ярлыки, т. е. вербализует знания.

В-третьих, эксперту труднее создать модель предметной области вследствие глубины и объема информации, которой он владеет. Еще в ситуационном управлении [Поспелов, 1986] было выявлено, что объекты реального мира связаны более чем 200 типами отношений (временные, пространственные, причинно-следственные, типа «часть — целое» и др.). Эти отношения и связи предметной области образуют сложную систему, из которой выделить «скелет» или главную структуру иногда доступнее аналитику, владеющему к тому же системной методологией.

Термин *приобретение* в рамках данной книги оставлен за автоматизированными системами прямого общения с экспертом. Они действительно непосредственно приобретают уже готовые фрагменты знаний в соответствии со структурами, заложенными разработчиками систем. Большинство этих инструментальных средств специально ориентировано на конкретные ЭС с жестко обозначенной предметной областью и моделью представления знаний, т. е. не являются универсальными.

> Приобретение знаний (knowledge acquisition) — процесс наполнения базы знаний экспертом с использованием специализированных программных средств.

Например, система TEIRESIAS [Davis, 1982], ставшая прародительницей всех инструментариев для приобретения знаний, предназначена для пополнения базы знаний системы MYCIN или ее дочерних ветвей, построенных на «оболочке» EMYCIN [Shortliffe, Buchanan, 1976] в области медицинской диагностики с использованием продукционной модели представления знаний.

Термин *формирование знаний* традиционно закрепился за чрезвычайно перспективной и активно развивающейся областью инженерии знаний, которая занимается разработкой моделей, методов и алгоритмов обучения. Она включает индуктивные модели формирования знаний и автоматического порождения гипотез, например, ДСМ-метод [Аншаков, Скворцов, Финн, 1986] на основе обучающих выборок, обучение по аналогии и другие методы. Эти модели позволяют выявить причинно-следственные эмпирические зависимости в базах данных с неполной информацией, содержащих структурированные числовые и символьные объекты (часто в условиях неполноты информации).

> Формирование знаний (machine learning) — процесс анализа данных и выявление скрытых закономерностей с использованием специального математического аппарата и программных средств.

Таким образом, можно выделить три основные стратегии проведения стадии получения знаний (рис. 2.3).

A. *Извлечение* знаний — без использования вычислительной техники путем непосредственного контакта инженера по знаниям и источника знаний (будь то эксперт, специальная литература или другие источники).

B. *Приобретение* знаний — с использованием ЭВМ при наличии подходящего программного инструментария.

C. *Формирование* знаний — с использованием программ обучения при наличии репрезентативной (т. е. достаточно представительной) выборки примеров принятия решений в предметной области и соответствующих пакетов прикладных программ. Этот способ подробнее будет рассмотрен в п. 2.6.

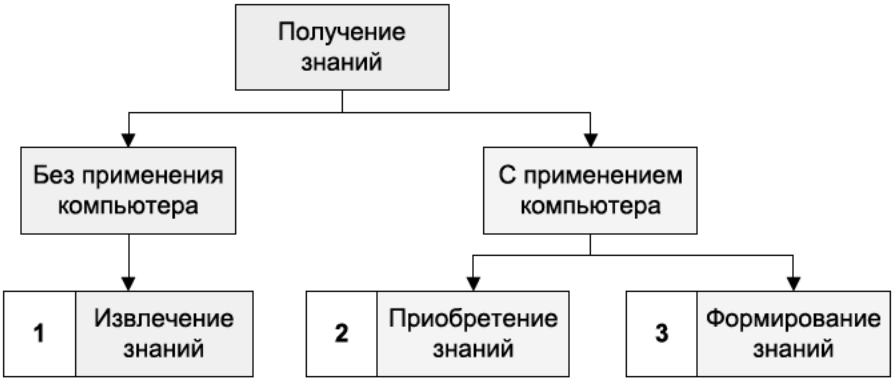


Рис. 2.3

Стратегии получения знаний

А. Прямое извлечение знаний

Прямое извлечение знаний происходит посредством общения аналитика и эксперта. Общение, или коммуникация (от лат. *Communicatio* — связь), — это междисциплинарное понятие, обозначающее все формы непосредственных контактов между людьми — от дружеских до деловых. Существует несколько десятков теорий общения, и единственное, в чем сходятся все авторы, — это сложность, многоплановость процедуры общения. Подчеркивается, что общение — не просто однонаправленный процесс передачи сообщений и не двухтактный обмен порциями сведений, а нерасчлененный процесс циркуляции информации, т. е. совместный поиск истины [Каган, 1988] (рис. 2.4).

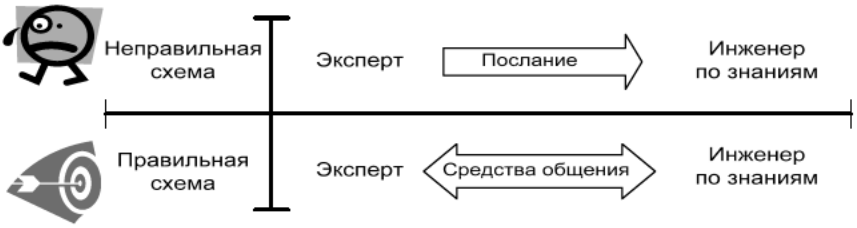


Рис. 2.4

Структура процесса общения

Итак, общение есть процесс выработки новой информации, общей для общающихся людей и рождающей их общность. И хотя общение — первый вид деятельности, которым овладевает человек в онтогенезе, по-настоящему владеют культурой и наукой общения единицы.

Извлечение знаний — это особый вид общения, который можно отнести к духовно-информационному типу. Согласно [Каган, 1988], общение делится на материально-практическое; духовно-информационное; практически-духовное. При этом информационный аспект общения для инженера по знаниям с прагматической точки зрения важнейший.

Известно, что потери информации при разговорном общении велики [Мицич, 1987] (рис. 2.5).

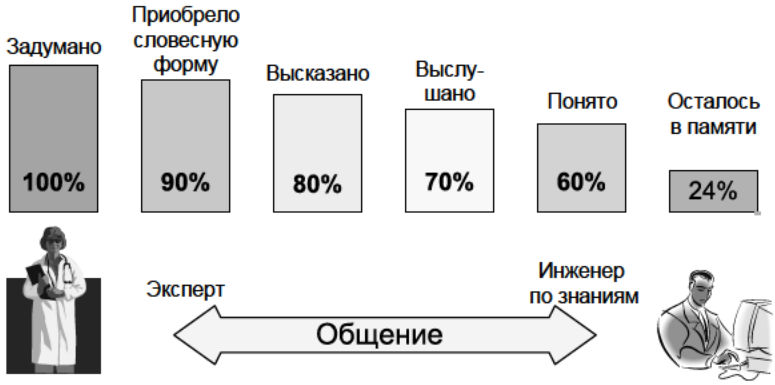


Рис. 2.5

Потери информации при разговорном общении

В связи с этим важно увеличение информативности общения аналитика и эксперта за счет использования профессиональных методов и приемов (см. далее эту и следующую главы).

Анализ современного состояния проблематики прямого приобретения знаний и поддержки деятельности инженера по знаниям позволяет выявить две группы проблем: методологические проблемы и технологические проблемы.

Б. Автоматизированное приобретение знаний

Ниже мы кратко рассмотрим автоматизированный подход к проблеме извлечения и структурирования знаний, традиционно называемый *приобретением знаний* (knowledge acquisition).

Первое поколение. Первые системы приобретения знаний (СПЗ) появились в середине 1980-х гг. (TEIRESIAS [Davis, 1982], SIMER+MIR [Осипов, 1988], АРИАДНА [Моргоев, 1988]). Это были средства наполнения так называемых «пустых» ЭС, т. е. систем, из БЗ которых изъяты знания (например, EMYCIN — EMPTY MYCIN, опустошенная медицинская ЭС MYCIN со специальной диалоговой системой заполнения базы знаний TEIRESIAS). Их авторы считали, что прямой диалог эксперта с компьютером через СПЗ поможет сократить жизненный цикл разработки. Однако опыт создания и внедрения СПЗ продемонстрировал несовершенство такого подхода.

Основные недостатки СПЗ первого поколения:

- слабая проработка методов извлечения и структурирования знаний;
- жесткость модели представления знаний, встроенной в СПЗ и связанной с привязкой к программной реализации;
- ограничения на предметную область.

Таким образом, традиционная схема разработки СПЗ первого поколения:

создание конкретной ЭС ->
опустошение БЗ ->
разработка СПЗ для новых наполнений БЗ ->
формирование новой БЗ для другой ЭС

оказалась несостоятельной для промышленного применения.

Второе поколение СПЗ появилось в конце 1980-х гг. и было ориентировано на более широкий модельный подход [Gaines, 1989; Борисов, Федоров, Архипов, 1991] с акцентом на предварительном детальном анализе предметной области. Так, в Европе широкое применение получила методология KADS (Knowledge Acquisition and Documentation Structuring) [Wielinga et al., 1989], в основе которой лежит понятие интерпретационной модели, позволяющей процессы извлечения, структурирования и формализации знаний рассматривать как «интерпретацию» лингвистических знаний в другие представления и структуры.

Успехи СПЗ второго поколения позволили значительно расширить рынок ЭС, который к концу 1980-х гг. оценивался в 300 млн долл. в год [Попов, 1996]. Тем не менее, и эти системы были не свободны от недостатков, к основным из которых можно отнести:

- несовершенство интерфейса, в результате чего неподготовленные эксперты не способны овладеть системой и отвергают ее;
- сложность настройки на конкретную профессиональную языковую среду;
- необходимость разработки дорогостоящих лингвистических процессоров для анализа естественно-языковых сообщений и текстов.

Третье поколение СПЗ (например, KEATS [Eisenstadt et al., 1990], MACAO [Aussenac-Gilles, Natta, 1992], и др.) — перенесло акцент в проектировании с эксперта на инженера по знаниям [Гаврилова, 1988; Wallace & Gruber, 1989]. Новые СПЗ — это программные средства для аналитика, более сложные, гибкие, а главное использующие *графические* возможности современных рабочих станций и достижения CASE-технологии (Computer-Aided Software Engineering). Эти системы позволяют не задавать заранее интерпретационную модель, а формировать структуру БЗ динамически.

Следует отметить ряд отечественных систем, например, SIMER + MIR, разработанная в ИПС РАН под руководством Г. С. Осипова [Осипов, 1997], представляет собой совокупность программных средств для формирования модели и базы знаний предметной области. Система ориентирована преимущественно на области с неясной структурой объектов, с неполно описанным множеством свойств объектов и богатым набором связей различной «связывающей силы» между объектами.

Существуют различные классификации СПЗ — по выразительности и мощности инструментальных средств [Попов, Фоминых и др., 1996]; по обобщенным характеристикам [Boose, 1989]; в рамках структурно-функционального подхода [Волков, Ломнев, 1989]; интегрированная классификация предложена в работе [Гаврилова, Червинская, 1992].

Учитывая новейшие тенденции в инженерии знаний можно предложить следующую схему таксономии СПЗ, представленную на рисунке 2.6.

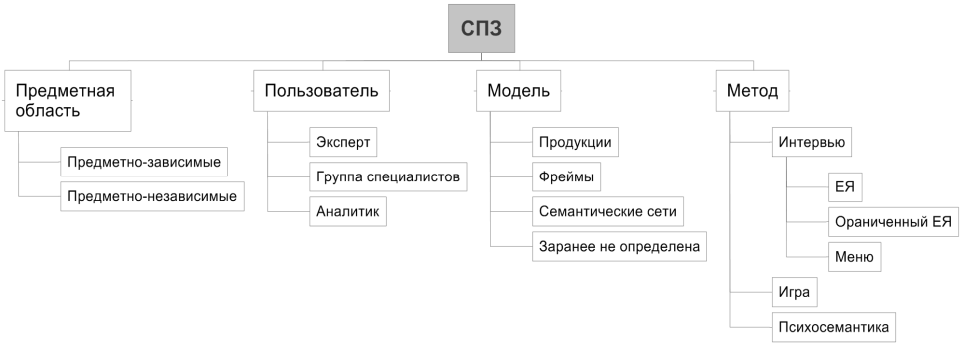


Рис. 2.6

Классификация систем приобретения знаний

Однако и современные СПЗ не полностью лишены серьезных недостатков СПЗ первого и второго поколений, большая часть которых обусловлена отсутствием теоретической концепции проектирования БЗ (подробно см. в [Гаврилова, Хорошевский, 2000]). В результате эта область до настоящего времени справедливо считается скорее «искусством», чем наукой, и основана на «ad hoc»-технологии (т. е. применительно к случаю).

Завершая обзор прямых методов приобретения знаний, отметим следующее:

- эти методы не устраняют посредника между системой и экспертом;
- автономное использование описанных методов не решает таких проблем инженерии знаний, как устранение «пробелов» в знаниях, выявление «глубинных», невербальных знаний; сохраняется большая «время-емкость» и субъективность интервью;

- фаза приобретения знаний идеологически и теоретически не связывается со следующими фазами инженерии знаний.

B. Формирование знаний (knowledge discovery)

Традиционно к задачам формирования знаний или машинного обучения относятся задачи прогнозирования, идентификация (синтеза) функций, расшифровки языков, индуктивного вывода и синтеза с дополнительной информацией [Епифанов, 1984]. В широком смысле к обучению по примерам можно отнести и методы обучения распознаванию образов [Аткинсон, 1989; Bishop, 2006]. Индуктивный вывод правил из фактов применен также в системах AQ, AQUINAS, KSSI, INSTIL и некоторых других.

Формирование знаний, тяготеющее в большей степени к области machine learning, т. е. индуктивному обучению, основывается на хорошо исследованном аппарате распознавания образов [Гаек, Гавранек, 1983] и обнаружения сходства объектов [Гусакова, Финн, 1987], в последнее время все чаще называется data mining или knowledge discovery (подробнее см. далее или в [Berry & Linoff, 2006; Чубукова, 2006]). Эти методы приобретают сейчас широкую популярность, базируясь на анализе данных и поиске закономерностей. Методы и инструменты формирования знаний из данных для бизнес-приложений подробно рассмотрены в главе Business Intelligence.

Однако применение методов формирования знаний пока не стало промышленной технологией разработки баз знаний. Для того чтобы эти методы стали элементами технологии интеллектуальных систем, необходимо решить ряд задач [Осипов, 1997]:

- обеспечить механизм сопряжения независимо созданных баз данных, имеющих различные схемы, с базами знаний интеллектуальных систем;
- установить соответствие между набором полей базы данных и множеством элементов декларативной компоненты базы знаний;
- выполнить преобразование результата работы алгоритма обучения в способ представления, поддерживаемый программными средствами интеллектуальной системы.

Помимо перечисленных существуют также и другие стратегии получения знаний, например, в случае обучения на примерах (case-based reasoning), когда источник знаний — это множество примеров предметной области [Осипов, 1997; Попов, Фоминых, Кисель, 1996; Иберла, 1980]. Обучение на основе примеров (прецедентов) включает настройку алгоритма распознавания на задачу посредством предъявления примеров, классификация которых известна.

Подробнее о современных информационно-аналитических системах рассказано в п. 2.6.

2.3. ОПРЕДЕЛЕНИЕ И СТРУКТУРА ИНЖЕНЕРИИ ЗНАНИЙ

Инженерия знания (knowledge engineering) — достаточно молодое направление искусственного интеллекта, возникшее тогда, когда практические разработчики столкнулись с весьма нетривиальными проблемами трудности «добычи» и формализации знаний. В первых книгах по ИИ эти факты обычно только постулировались, в дальнейшем начались серьезные исследования по выявлению эффективных стратегий выявления знаний [Chang, 2002; Chorafas, 1990; Cuena, 2000; Davenport, 2000; Debenham, 1998; Kelly, 1999 и др.].

Инженерия знаний — это ветвь информатики, изучающая модели и методы извлечения, структурирования и формализации (представления) знаний для их обработки в интеллектуальных и информационных системах.

Традиционно при разработке систем, основанных на знаниях (knowledge based systems), отчетливо выделяют три фазы домашинной обработки (рис. 2.7).

- Первая — добыча (получение) знаний из источника (эксперты, Интернет, специальная литература). Трудоемкость этой фазы недооценена. Результат ее — огромное количество гетерогенных (разнохарактерных) противоречивых фрагментов знаний.
- Вторая — концептуализация (структурирование) разрозненных фрагментов в единую модель. Результат — часто слабоформализованное представление, называемое полем знаний.
- Третья — формализация поля знаний при помощи специализированных языков представления знаний (ЯПЗ), описанных в предыдущей главе. Результат — база знаний, описанная на ЯПЗ.

Рисунок 2.7 иллюстрирует основные фазы обработки знаний — начиная с получения знаний от источника через структурирование к формализации.

Центральным понятием на стадиях получения и структурирования является так называемое *поле знаний* [Гаврилова, 1984].

Поле знаний — это условное неформальное описание основных понятий и взаимосвязей между понятиями предметной области, выявленных из системы знаний эксперта, в виде графа /диаграммы, таблицы или текста.

Очевидно, что поле знаний — это один из способов «компрессии» знаний и их наглядного представления.

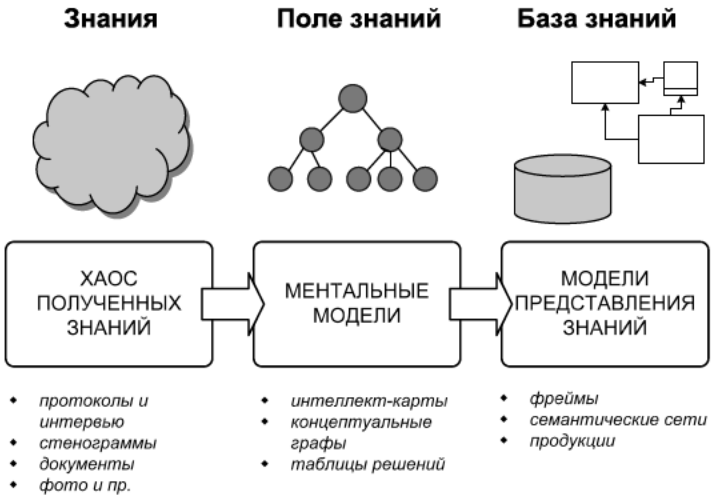


Рис. 2.7

Основные фазы обработки знаний

Теперь можно проиллюстрировать структуру молодого научного направления, ветви искусственного интеллекта — инженерии знаний (рис. 2.8).

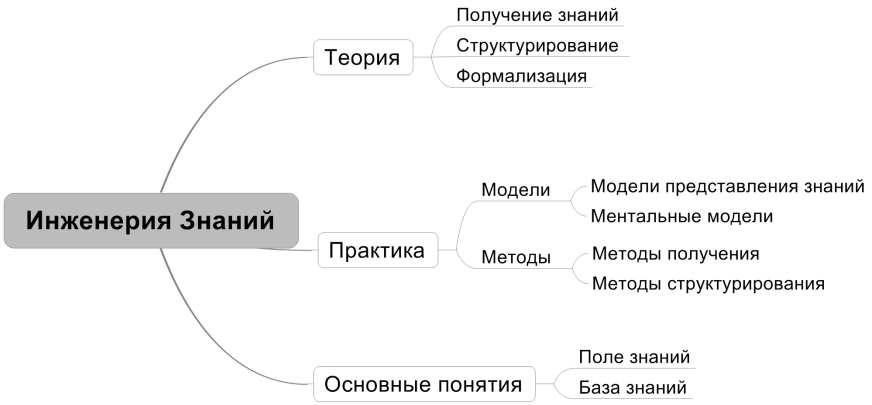


Рис. 2.8
Структура инженерии знаний

2.4. ОСНОВНЫЕ АСПЕКТЫ ИНЖЕНЕРИИ ЗНАНИЙ

А) Психологический аспект

Из трех аспектов извлечения знаний психологический является ведущим, поскольку он определяет успешность и эффективность взаимодействия инженера по знаниям (аналитика) с основным источником знаний — экспертом-профессионалом. Психологический аспект выделяется еще и потому, что извлечение знаний происходит чаще всего в процессе непосредственного общения разработчиков системы. А законы психологии общения, к сожалению, часто игнорируются начинающими аналитиками.

Упрощенно можно выделить следующие структурные компоненты модели общения [Каган, 1986]:

- участники общения (*партнеры*);
- средства общения *(процедура*);
- предмет общения (*знания*).

В соответствии с этой структурой выделим и последовательно обсудим три «слоя» психологических проблем, возникающих при извлечении знаний [Гаврилова, Червинская, 1992] (рис. 2.9):

- контактный (проблемы взаимодействие *партнеров*);
- процедурный (проблемы грамотного проведения *процедуры* извлечения);
- когнитивный (проблема получения *знания* как такового).

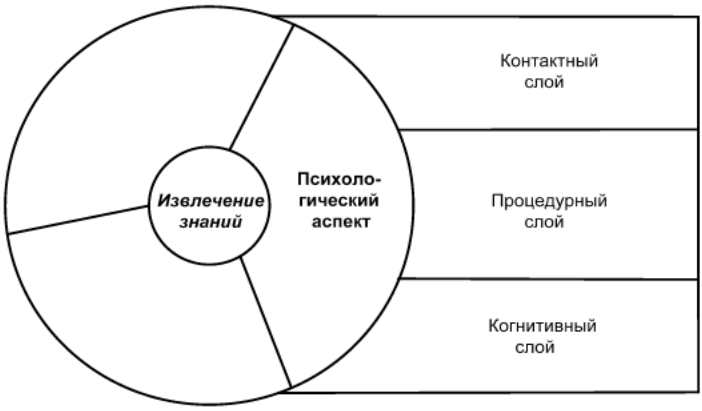


Рис. 2.9

Психологический аспект извлечения знаний

Контактный слой

Психологи отмечают, что на любой коллективный процесс влияет атмосфера, возникающая в группе участников. Существуют эксперименты, результаты которых неоспоримо говорят, что часто дружеская атмосфера в коллективе больше влияет на результат, чем индивидуальные способности отдельных членов группы [Брукс, 2000; Брукс, 2012]. Особенно важно, чтобы в коллективе разработчиков складывались кооперативные, а не конкурентные отношения. Для кооперации характерна атмосфера сотрудничества, взаимопомощи, заинтересованности в успехах друг друга, т. е. уровень нравственного общения, а для отношений конкурентного типа — атмосфера индивидуализма и межличностного соперничества (более низкий уровень общения).

В настоящее время прогнозировать совместимость в общении со 100%-ной гарантией невозможно. Однако можно выделить ряд факторов и черт личности, характера и других особенностей участников общения, оказывающих влияние на эффективность процедуры.

Наш опыт работы и анализ специальной литературы позволили выявить следующие параметры партнеров, влияющие на результаты процедуры извлечения знаний (рис. 2.10):

- демографические (пол, возраст, национальность, образование);
- личностные (особенности личностного портрета, темперамент, мотивация и др.).

Демографические параметры, хотя и влияют на эффективность контакта, но не являются критическими. В литературе [Иванов, 1986] отмечается, что при работе в паре хорошие результаты дают гетерогенные пары (мужчина/женщина) и соотношение:

$20 > (Вэ — Ва) > 5$, где $Вэ$ — возраст эксперта; $Ва$ — возраст аналитика.

Следует учитывать также гендерные различия (табл. 2.1) [Брагина, Доброхотова, 1988; Черниговская, 2004].

Социокультурные и национальные особенности следует безусловно учитывать (особенно в международных проектах и коллективах), хотя это и чрезвычайно «тонкий» аспект.



Рис. 2.10

Контактный слой проблем

Таблица 2.1

Гендерные отличия

Мужчины	Женщины
Лучше пространственнная ориентация	Лучше коммуникативные свойства
Выше аналитичность и поленезависимость	Сильнее межполушарные связи
Интерес к поиску новых решений (охота)	Минимизация риска, боязнь нового
Лучше получается генерализация	Лучше получается детализация
Хуже восприятие на слух	Хуже пространственная ориентация. Путают «право-лево» (50%)

Перейдем к личностным параметрам. Под личностью обычно понимается устойчивая система психологических черт, характеризующая индивидуальность человека. Рекомендуемый личностный портрет описан в работе [Гаврилова, 1989] и дополнен качествами из руководства для журналистов в [Schouksmith G., 1978]:

- доброжелательность, общительность;
- аналитические способности, находчивость;
- хорошая память, внимание;
- наблюдательность, воображение, впечатлительность;
- лидерские качества, настойчивость;
- чувство юмора.

Отдельно следует отметить характеристики темперамента. Со времен Галена и Гиппократа, выделивших четыре классических типа темперамента, вошли в научную терминологию понятия «холерик», «сангвиник», «меланхолик» и «флегматик».

Известно, что флегматики и меланхолики медленнее усваивают информацию [Лунева, Хорошилова, 1987]. И для обеспечения психологического контакта с ними не следует задавать беседе слишком быстрый темп, торопить их с ответом. Зато они гораздо лучше усваивают новое, в отличие от холериков, для которых свойственно поверхностное усвоение информации. На эффективность коллективного решения задач влияет также и мотивация, т. е. стремление к успеху.

Инженер по знаниям в зависимости от условий разработки должен изыскивать разнообразные стимулы для экспертов (включая, разумеется, и материальные). Эксперт передает аналитику один из самых дорогих в мире продуктов — знания. И если одни люди делятся опытом добровольно и с удовольствием, то другие весьма неохотно приоткрывают свои профессиональные тайны. Иногда полезно возбудить в эксперте дух соперничества, конкуренции (не нарушая, естественно, обстановки кооперативности в коллективе).

Процедурный слой

Параметры процедурного слоя описывают непосредственно процесс проведения процедуры извлечения знаний. Фактически это профессиональные параметры (рис. 2.11):

- ситуация общения (место, время, продолжительность);
- оборудование (вспомогательные средства, освещенность, мебель);
- профессиональные приемы (темп, стиль, мимика, интонации и др.).



Рис. 2.11

Процедурный слой психологического аспекта инженерии знаний

Инженеру по знаниям, успешно овладевшему наукой и искусством установления дружеского контакта, доверия и взаимопонимания с экспертом, нужно еще суметь воспользоваться благоприятным воздействием этой победы. Проблемы процедурного слоя касаются проведения самого процесса извлечения знаний. Здесь недостаточно проницательности и обаяния, полезных для решения проблемы контакта, а потому необходимы профессиональные знания.

Остановимся на общих закономерностях проведения процедуры.

Ситуацию общения определяют следующие компоненты:

- место проведения сеансов;
- время проведения сеансов.

Беседу с экспертом лучше всего проводить в небольшом помещении наедине, поскольку посторонние люди нарушают доверительность беседы и могут породить эффект «фасада». Рабочее место эксперта является не самым оптимальным вариантом, так как его могут отвлекать телефонные звонки, сотрудники и пр. Атмосфера замкнутого пространства и уединенности положительно влияют на эффективность. Чай или кофе помогут снять излишнее напряжение эксперта.

Американский психолог И. Атватер считает, что для делового общения наиболее благоприятная дистанция от 1,2 до 3 м [Schouksmith, 1978]. Минимальным «комфортным» расстоянием в евро-американском социуме можно считать 0,7–0,8 м. На Востоке (Япония) это расстояние существенно больше. В настоящее время существует специальное направление инженерной психологии, которое называется проксемика, изучающее влияние пространственных размещений на эффективность труда.

Реконструкция собственных рассуждений — трудоемкий процесс, и поэтому длительность одного сеанса обычно не превышает 1,5–2 часа. Лучше провести сеанс в первой половине дня, например, с 10 до 12 часов, если эксперт относится к типу «жаворонок», или во второй половине дня (15–17 часов) для «сов». Так называемые циркадные, или циркадианные, ритмы, связанные с суточным циклом Солнца, имеют существенное влияние на активность головного мозга [Алпатов, 1993]. Так, известно о двух среднестатистических пиках суточной активности — 11 часов утра (для «жаворонков») и 4 часа дня (для «сов»).

Оборудование включает:

- свет;
- аппаратуру;
- мебель.

Освещение должно быть комфортным для эксперта, чтобы процедура извлечения знаний не напоминала допрос с пристрастием. Желательно иметь локальный светильник с абажуром либо использовать естественное освещение. Не рекомендуется сажать эксперта против света, особенно эксперта-женщину.

Под аппаратурой понимаются вспомогательные средства:

- средства для протоколирования результатов;
- средства для увеличения эффективности самого процесса извлечения знаний.

К средствам для увеличения эффективности процесса извлечения знаний, прежде всего, относится наглядный материал [Наст, 2008; Михалко, 2015]. Можно считать универсальным совет активно использовать наглядный материал. Такие методы, как свободный диалог и игры, предоставляют богатые возможности применять слайды, чертежи, рисунки (см. далее главу 3).

Для протоколирования результатов в настоящее время используются следующие способы:

- запись на бумагу непосредственно по ходу беседы (недостатки — это часто мешает беседе, кроме того, трудно успеть записать все даже при наличии навыков стенографии);
- видеозапись, помогающая аналитику проанализировать весь ход сеанса и свои ошибки (недостатки — требует оператора и сковывает эксперта);
- использование диктофона (недостаток — может сковывать эксперта);
- запоминание с последующей записью после беседы (недостаток — годится только для аналитиков с блестящей памятью).

Наиболее распространенным способом на сегодня является первый. При этом существует опасность потери знаний, поскольку любая запись ответов — это уже интерпретация, т. е. привнесение субъективного понимания предмета.

Мебель является второстепенным фактором, но обшарпанный стул или обстановка неубранного стола могут существенно снизить эффективность процедуры. Уютное кресло, напротив, может удлинить сеанс. Однако доводить дело до психоаналитической кушетки, наверное, не стоит.

Профессиональные приемы аналитика включают в частности:

- речь;
- невербальные компоненты.

Учет индивидуального темпа и стиля эксперта позволяет аналитику снизить напряженность процедуры извлечения знаний. Типичной ошибкой является навязывание собственного темпа и стиля. Громкость также следует настроить под эксперта, ориентируясь на его реакции. С пожилым экспертом следует говорить громче, иначе он сочтет речь аналитика невнятной, при этом переспрашивать, скорее всего, не будет.

На успешность также влияет длина фраз, которые произносит инженер по знаниям. Этот факт был установлен американскими учеными — лингвистом Ингве и психологом Миллером при проведении исследования о причинах низкой усвояемости команд на военно-морском флоте США [Gammack, Young, 1985]. Причина была в длине команд. Оказалось, что человек лучше всего воспринимает предложения глубиной (или длиной) 7+2(–2) слова. Это число (7 ± 2) получило название число Ингве — Миллера. Можно считать его мерой «разговорности» речи. Опытные лекторы используют в лекции в основном короткие фразы, уменьшая потерю информации с 20–30% (у плохих лекторов) до 3–4% [Горелов, 1987].

Большая часть информации поступает к инженеру по знаниям в форме предложений на естественном языке. Однако внешняя речь эксперта есть воспроизведение его внутренней речи (мышления), которая гораздо богаче и многообразнее. При этом для передачи внутренней речи эксперт использует и невербальные средства, такие как: интонация, мимика, жесты. Опытный инженер по знаниям старается записывать по возможности в протоколы (в форме ремарок) эту дополнительную интонацию.

Имеется достаточно литературы по поводу «языка тела» и «языков жестов» [Пиз, Гарднер, 2000], но зачастую она носит спекулятивно-развлекательный характер. Где-то на этой границе находится и литература по НЛП (нейролингвистическое программирование) [Андреас и др., 2000], обучающая скрытым техникам манипулирования людьми. На наш взгляд, следует использовать некоторые безопасные и безобидные приемы из подобного рода арсенала. Например, о позе [Egan, 1994]:

- сесть прямо напротив специалиста;
- принять открытую позу (ноги и руки не скрещены, ладони разжаты);
- легкий наклон вперед;
- расслабиться и сохранять зрительный контакт.

В целом невербальная компонента стиля общения важна и для проблем контактного слоя, когда по отдельным жестам и выражению лица эксперта инженер по знаниям может установить границу возможной «дружественности» общения.

Когнитивный слой

Когнитивные (от англ. *Cognition* — познание) науки исследуют познавательные процессы человека с позиций возможности их моделирования (психология, нейрофизиология, эргономика, инженерия знаний). Наименее исследованы на сегодняшний день проблемы когнитивного слоя, связанные с изучением семантического пространства памяти эксперта и реконструкцией его понятийной структуры и модели рассуждений.

Основные факторы, влияющие на когнитивную адекватность, показаны на рисунке 2.12. К ним относятся:

- когнитивный стиль;
- семантическая репрезентативность поля знаний и концептуальной модели.

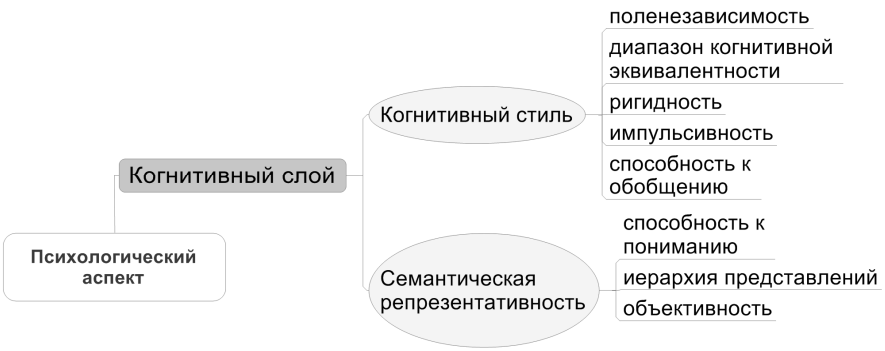


Рис. 2.12

Когнитивный слой

Под *когнитивным стилем* человека понимается совокупность критериев предпочтения при решении задач и познании мира, специфическая для каждого человека. Когнитивный стиль определяет не столько эффективность деятельности, сколько способ достижения результата [Алахвердов, 1986; Холодная, 2002]. Это способ познания, позволяющий людям с разными способностями добиваться одинаковых результатов в деятельности. Это система средств и индивидуальных приемов, к которым прибегает человек для организации своей деятельности.

Инженеру по знаниям полезно изучить и прогнозировать свой когнитивный стиль, а также стиль эксперта. Особенно важны следующие характеристики когнитивного стиля:

- полезависимость — поленезависимость;
- диапазон когнитивной эквивалентности;
- импульсивность — рефлективность (рефлексивность);
- ригидность — гибкость;
- способность к обобщению (генерализации) понятий.

Поленезависимость позволяет человеку акцентировать внимание лишь на тех аспектах проблемы, которые необходимы для решения конкретной задачи, и уметь отбрасывать все лишнее, т. е. не зависеть от фона или окружающего задачу шумового поля. Эта характеристика коррелирует с такими чертами личности, как невербальный интеллект, аналитичность мышления, способность к пониманию сути. Очевидно, что помимо того, что самому аналитику необходимо иметь высокое значение этого параметра, поленезависимый эксперт — тоже желательный фактор. Однако приходится учитывать, что больше нуждаются в общении полезависимые люди, а потому они и более контактны [Орехов, 1985].

Особенно полезны для общения гетерогенные (смешанные) пары, например, «полезависимый — поленезависимый» [Иванов, 1986]. В литературе описаны различные эксперименты, моделирующие общение, требующее понимания и сов-

местной деятельности. Наиболее успешным в понимании оказались поленезависимые испытуемые (92% успеха), для сравнения полезависимые давали 56% успеха [Кулюткин, Сухобская, 1971].

Для совместной профессиональной деятельности важна также гибкость когнитивной организации, которая связана с поленезависимостью. Итак, большую способность к адекватному пониманию партнера демонстрируют субъекты с высокой психологической дифференциацией, т. е. поленезависимостью.

Поленезависимость является одной из характерных профессиональных черт когнитивного стиля наиболее квалифицированных инженеров по знаниям. По некоторым результатам [Алахвердов, 1986] мужчины более поленезависимы, чем женщины.

Когнитивная эквивалентность характеризует способность человека к различению понятий и разбиению их на классы и подклассы. Чем ỳже диапазон когнитивной эквивалентности, тем более тонкую классификацию способен провести индивид, тем большее количество признаков понятий он может выделить. Обычно у женщин диапазон когнитивной эквивалентности ỳже, чем у мужчин.

Ригидность (в противовес *гибкости*) характеризует способность или неспособность человека к изменению установок и точек зрения в соответствии с изменяющейся ситуацией. Ригидные люди не склонны менять свои представления и структуру восприятия, напротив, гибкие легко приспосабливаются к новой обстановке. Увеличение ригидности с возрастом отмечается многими психологами [Кулюткин, Сухобская, 1971; Орехов, 1985].

Под *импульсивностью* понимается быстрое принятие решения (часто без его достаточного обоснования), а под *рефлексивностью* — склонность к рассудительности. Рефлексивность по экспериментальным данным коррелирует со способностью к формированию понятий и продуктивностью стратегий решения логических задач [Кулюткин, Сухобская, 1971]. Таким образом, и инженеру по знаниям, и эксперту желательно быть рефлексивным, хотя собственный стиль изменяется лишь частично и с большим напряжением.

Способность к обобщению понятий чрезвычайно важна в инженерии знаний, так как поле знаний иерархично. Генерализация — сугубо креативная функция, основанная на анализе свойств понятий, она позволяет синтезировать метапонятия, интегрирующие понятия в группы со сходными свойствами. Отдельный вопрос возникает по поводу присвоения новых имен полученным классам понятий.

Семантическая репрезентативность подразумевает подход, исключающий традиционное навязывание эксперту некой модели представлений (например, продукционной или фреймовой), а напротив, заставляет инженера по знаниям последовательно воссоздавать модель мира эксперта, используя как неформальные методы, так и математический аппарат, например, многомерное шкалирование. Проблема семантической репрезентативности ориентирована на достижение когнитивной адекватности поля знаний и концептуальной модели. В настоящий момент она может быть сформулирована как проблема «испорченного телефона» [Гаврилова, Червинская, 1992] — возможные трансформации и потери в цепи передачи информации.

На достижение семантической репрезентативности влияют следующие факторы:

- способности к пониманию и обобщению инженера по знаниям;
- умение работать с разноуровневыми моделями с учетом иерархии понятий;
- объективность.

Б) Лингвистический аспект

Лингвистический аспект касается исследования языковых проблем, так как язык — это основное средство общения в процессе извлечения знаний.

Сразу же уточним, что поскольку тема данной книги ограничена изложением теории и технологии инженерии знаний, то область разработки естественно-языковых интерфейсов и весь спектр проблем, связанных с нею, — лексических, синтаксических, семантических, прагматических и т. д. [Мальковский, 1985; Попов, 1982; Рубашкин, 2001; Лукашевич, 2011; Jensen, Heidorn & Richardson, 2012], не рассматривается.

В рамках инженерии знаний можно выделить три слоя лингвистических проблем (рис. 2.13):

- «общий код»;
- понятийная структура;
- словарь пользователя.

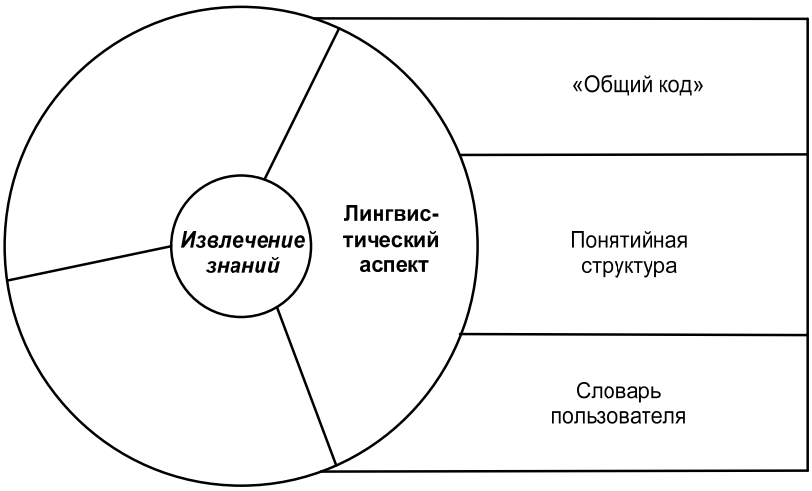


Рис. 2.13

Лингвистический аспект извлечения знаний

«Общий код»

Общий код решает проблему так называемых языковых ножниц между профессиональной терминологией эксперта и обыденной литературной речью инженера по знаниям.

Различия языков аналитика и эксперта обусловливает «языковый барьер» или «языковые ножницы» в общении инженера по знаниям и эксперта. Формирование общего кода включает изучение и управление процессом разработки специального промежуточного языка, необходимого для взаимодействия инженера по знаниям и эксперта.

Понятийная структура

Проблемы формирования понятийной структуры представляют следующий слой лингвистического аспекта проблемы извлечения знаний и описан подробнее в следующей главе.

Словарь пользователя

Лингвистические результаты, соотнесенные к слоям общего кода и понятийной структуры, направлены на создание адекватной базы знаний. Однако часто профессиональный уровень конечного пользователя не позволяет ему применить специальный язык предметной области в полном объеме.

В) Методологический аспект

Методология инженерии знаний находится на стадии становления. Как в любой междисциплинарной области, методологические аспекты исследований вызывают разночтения.

Некоторую «поддержку» можно получить в области философии, в разделе гносеологии. *Гносеология* — это раздел философии, связанный с теорией познания, или теорией отражения действительности в сознании человека. Гносеологический аспект извлечения знаний объединяет методологические проблемы получения нового научного знания, поскольку при создании БЗ эксперт часто впервые формулирует некоторые закономерности, до того момента составлявшие его личный опыт.

Инженерия знаний как наука, если можно так выразиться, дважды гносеологична — сначала действительность (*O*) отражается в сознании эксперта (*M1*), а затем деятельность и опыт эксперта интерпретируются сознанием инженера по знаниям (*M2*), что служит уже основой для построения третьей интерпретации (*P*) — поля знаний экспертной системы. Процесс познания, в сущности, направлен на создание внутренней репрезентации окружающего мира в сознании человека.

Познание часто сопровождается созданием новых понятий и теорий. Иногда эксперт порождает новые знания прямо в ходе беседы с аналитиком. Такая генерация знаний полезна и самому эксперту, который до того момента мог не осознавать ряд соотношений и закономерностей предметной области. Аналитику может помочь тут и инструментарий системной методологии, позволяющий использовать известные принципы логики научных исследований, понятийной иерархии науки. Эта методология заставляет его за частным всегда стремиться увидеть общее, т. е. строить гносеологические цепочки:

Гносеологическая цепочка: *факт → обобщенный факт →*

→ эмпирический закон → теоретический закон.

Не всегда удается дойти до последнего звена этой цепочки, но уже само стремление к движению бывает чрезвычайно плодотворным. Такой подход полностью согласуется со структурой самого знания, которое имеет два уровня: 1) эмпирический (наблюдения, явления); 2) теоретический (законы, абстракции, обобщения).

Основные характеристики эмпирического знания:

- модальность;
- противоречивость;
- неполнота.

Модальность знания означает возможность его существования в различных категориях, т. е. в конструкциях существования и долженствования (эксперт знает, что …; эксперт думает, что …; эксперт хочет, чтобы …; и т. п.).

Возможная противоречивость эмпирического знания — естественное следствие из основных законов диалектики, и противоречия эти не всегда должны разрешаться в поле знаний, а напротив, именно противоречия служат чаще всего отправной точкой в рассуждениях экспертов.

Неполнота знания связана с невозможностью полного описания предметной области.

Теоретические знания — это не только стройная система обобщения знания, а также некоторый способ производства новых знаний. Основными методологическими критериями таких знаний являются [Коршунов, Манталов, 1988]:

- внутренняя согласованность;
- системность;
- объективность;
- историзм.

Внутренняя согласованность пересекается с пониманием знания, очевидно, что согласованное применение системы обозначений, символов, терминов, способствует увеличению понятности.

Системность ориентирует аналитика на рассмотрение любой предметной области с позиций закономерностей системного целого и взаимодействия составляющих его частей.

Современный структурализм исходит из многоуровневой иерархической организации любого объекта, т. е. все процессы и явления можно рассматривать как множество более мелких подмножеств (признаков, деталей) и, наоборот, любые объекты можно (и нужно) рассматривать как элементы более высоких классов обобщений.

Объективность препятствует глубокому субъективизму познания, так он существенно зависит от особенностей самого познающего субъекта. «Факты существуют для одного глаза и отсутствуют для другого» (Роберт Виппер, русский историк). Таким образом, субъективность начинается уже с описания фактов и увеличивается по мере углубления идеализации объектов.

В психологии известен результат [Величковский, Капица, 1987], подтверждающий факт, что люди, быстро и успешно решающие интеллектуальные задачи, большую часть времени тратят на ее понимание, в то время как плохие решатели быстро приступают к поискам решения и чаще всего не могут его найти.

Историзм связан с развитием. Познание настоящего есть познание породившего его прошлого. И хотя большинство экспертных систем дают «горизонтальный» срез знаний без учета времени (в статике), инженер по знаниям должен всегда рассматривать процессы с учетом временных изменений — как связь с прошлым, так и связь с будущим.

Рассмотрев основные критерии научности познания, попытаемся теперь описать его структуру (подробнее в [Гаврилова, Хорошевский, 2000]). Методологическая структура познания может быть представлена как некоторая последовательность этапов [Коршунов, Манталов, 1988].

Поскольку основную трудность в создании интеллектуальных систем представляет *домашинный этап проектирования*, выполняемый инженером по знаниям (или аналитиком), — анализ предметной области, получение знаний и их структурирование, эти процедуры традиционно считаются «узким местом» (bottleneck) проектирования экспертных систем [Gaines, 1987; Boose, 1990]. Теоретические аспекты домашинного этапа обсуждались выше. Далее в этой главе будут рассмотрены практические методы и профессиональные приемы аналитика при извлечении и структурировании знаний.

2.5. КЛАССИФИКАЦИЯ ПРАКТИЧЕСКИХ МЕТОДОВ ИЗВЛЕЧЕНИЯ ЗНАНИЙ

Подробно рассмотрев выше теоретические аспекты инженерии знаний, мы, однако, в явном виде не определили, как эти знания будут получены. В неявном виде предполагалось, что это некоторое взаимодействие инженера по знаниям и

эксперта в форме непосредственного живого общения. Однако это не единственная форма извлечения знаний, хотя и довольно распространенная. В работах [Волков, Ломнев, 1989; Осипов, 1998; Boose, 1989; Gullen, Bryman, 1988; Kendal & Creen, 2006; Gavrilova, Andreeva, 2013] упоминаются более 20 ручных (неавтоматизированных) методов извлечения и более 10 автоматизированных методов приобретения и формирования знаний.

Введем классификацию методов извлечения знаний (рис. 2.14), обобщающую основные техники приемы и позволяющую инженерам по знаниям в зависимости от конкретной задачи и ситуации выбрать подходящий метод.

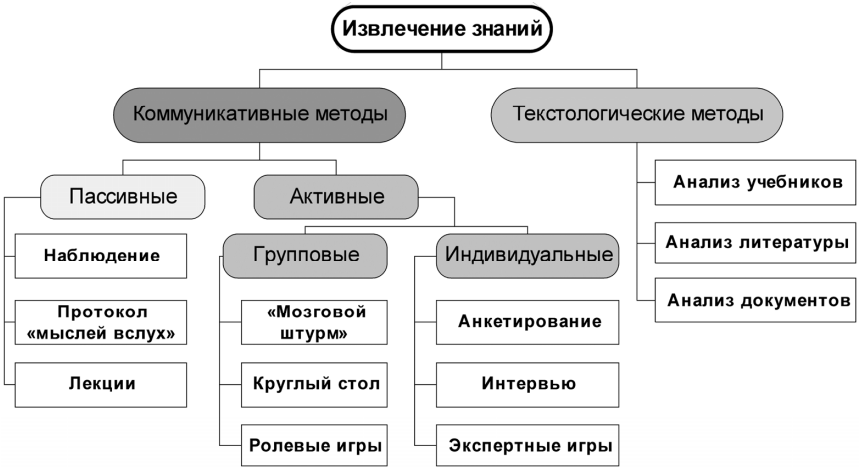


Рис. 2.14

Классификация методов извлечения знаний

Из предложенной схемы классификации видно, что основной принцип деления связан с источником знаний.

> Коммуникативные методы извлечения знаний охватывают методы и процедуры контактов инженера по знаниям с непосредственным источником знаний — экспертом, а текстологические включают методы извлечения знаний из документов (методик, пособий, руководств) и специальной литературы (статей, монографий, учебников).

Разделение этих групп методов на верхнем уровне классификации не означает их антагонистичности, обычно аналитик комбинирует различные методы.

В свою очередь, коммуникативные методы можно также разделить на две группы: активные и пассивные. *Пассивные* методы подразумевают, что ведущая роль в процедуре извлечения передается эксперту, а аналитик только протоколирует рассуждения эксперта во время его реальной работы по принятию решений или записывает то, что эксперт считает нужным самостоятельно рассказать. В *активных* методах, напротив, инициатива полностью в руках аналитика, который общается с экспертом различными способами — в играх, диалогах, беседах за «круглым столом» и т. д.

Активные методы можно разделить на две группы в зависимости от числа экспертов, отдающих свои знания. Если их число больше одного, то целесообразно

помимо серии индивидуальных контактов с каждым применять и методы групповых обсуждений предметной области. Такие *групповые* методы обычно активизируют мышление участников дискуссий и позволяют выявлять весьма нетривиальные аспекты их знаний. В свою очередь, *индивидуальные* методы на сегодняшний день остаются ведущими, поскольку столь деликатная процедура, как «отъем знаний», не терпит лишних свидетелей.

Отдельно следует сказать об играх. *Игровые* методы сейчас широко используются в социологии, экономике, менеджменте, педагогике для подготовки руководителей, учителей, врачей и других специалистов. Игра — это особая форма деятельности и творчества, где человек раскрепощается и чувствует себя намного свободнее, чем в обычной трудовой деятельности.

На выбор метода влияют три фактора: личностные особенности инженера по знаниям, личностные особенности эксперта и характеристика предметной области.

Одна из возможных классификаций людей по психологическим характеристикам [Обозов, 1986] делит всех на три типа:

- мыслитель (познавательный тип);
- собеседник (эмоционально-коммуникативный тип);
- практик (практический тип).

Мыслители ориентированы на интеллектуальную работу, учебу, теоретические обобщения и обладают такими характеристиками когнитивного стиля, как поленезависимость и рефлексивность. *Собеседники* — это общительные, открытые люди, готовые к сотрудничеству. *Практики* предпочитают действие разговорам, хорошо реализуют замыслы других, направлены на результативность работы.

Для характеристики предметных областей можно предложить следующую классификацию:

- хорошо документированные;
- средне документированные;
- слабо документированные.

Эта классификация связана с соотношением двух видов знаний Z_1 и Z_2, введенных ранее, где Z_1 — это экспертное «личное» знание, а Z_2 — материализованное в книгах «общее» знание в данной конкретной области. Если представить знания Zno предметной области как объединение Z_1 и Z_2, *т. е.* $Zno = Z_1 \cup Z_2$, то рисунок 2.15 наглядно иллюстрирует предложенную классификацию.

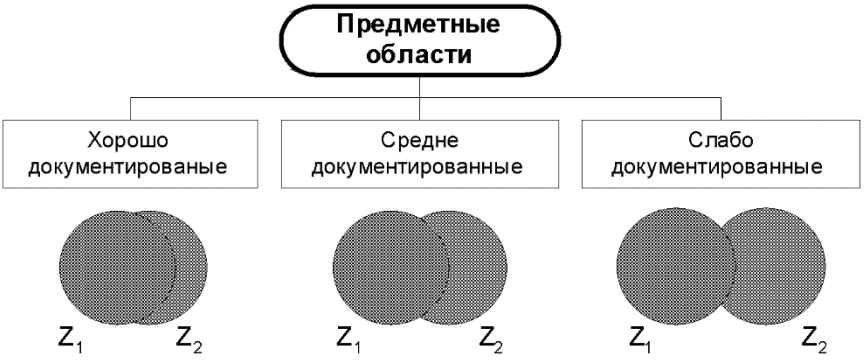


Рис. 2.15

Классификация предметных областей

Кроме этого, предметные области можно разделить по критерию структурированности знаний. Под *структурированностью* будем понимать степень теоретического осмысления и выявленности основных закономерностей и принципов, действующих в данной предметной области. И хотя ЭС традиционно применяются в слабоструктурированных предметных областях, сейчас наблюдается тенденция расширения сферы внедрения экспертных систем.

По степени структурированности знаний предметные области могут быть:

- *хорошо структурированными* — с четкой аксиоматизацией, широким применением математического аппарата, устоявшейся терминологией;
- *среднеструктурированными* — с определившейся терминологией, развивающейся теорией, явными взаимосвязями между явлениями;
- *слабоструктурированными* — с размытыми определениями, богатой эмпирикой, скрытыми взаимосвязями, с большим количеством «белых пятен».

Введенные в данном параграфе классификации методов и предметных областей помогут инженеру по знаниям, четко определив свою предметную область, соотнести ее с предложенными типами и наметить подходящий метод или группу методов извлечения знаний, учитывая, что метод должен подходить к эксперту как «ключ к замку».

Однако реальная работа может зачеркнуть выбор, и окажется, что его хорошо документированная область является слабо документированной, а метод наблюдений надо заменять анкетированием или вербальными отчетами. Такова реальная сложность процедуры извлечения знаний.

В соответствии с классификацией, представленной на рисунке 2.14, рассмотрим подробнее обе разновидности коммуникативных методов: *пассивные* и *активные*.

Коммуникативные пассивные методы

Пассивные методы на первый взгляд достаточно просты, но на самом деле требуют от инженера по знаниям умения четко анализировать поток сознания эксперта и выявлять в нем значимые фрагменты знаний. Отсутствие обратной связи (пассивность инженера по знаниям) значительно ослабляет эффективность этих методов, чем и объясняется их обычно вспомогательная роль при активных методах.

Сам термин «пассивные» не должен вызывать иллюзий, поскольку введен как противовес к «активным» методам. В реальности же пассивные методы требуют от инженера по знаниям не меньшей отдачи, чем такие активные методы, как игры и диалог.

Пассивные методы извлечения знаний включают такие методы,
где ведущая роль в процедуре извлечения знаний передается эксперту,
а инженер по знаниям только фиксирует рассуждения эксперта
во время работы по принятию решений.

Согласно классификации (см. рис. 2.14) к этой группе относятся:

- наблюдения;
- анализ протоколов «мыслей вслух»;
- лекции.

Наблюдения

В процессе наблюдений инженер по знаниям находится непосредственно рядом с экспертом во время его профессиональной деятельности или имитации этой деятельности. При подготовке к сеансу извлечения эксперту необходимо объяснить цель наблюдений и попросить максимально комментировать свои действия.

Во время сеанса аналитик записывает все действия эксперта, его реплики и объяснения. При согласии эксперта возможна организация видеозаписи в реальном

масштабе времени. Именно метод наблюдений является единственно «чистым» методом, исключающим вмешательство инженера по знаниям и навязывание им каких-то своих структур представлений.

Существуют две основные разновидности проведения наблюдений:

- наблюдение за реальным процессом;
- наблюдение за имитацией процесса.

Обычно используются обе разновидности. Сначала инженеру по знаниям полезно наблюдать за реальным процессом, чтобы глубже понять предметную область и отметить все внешние особенности процесса принятия решения. Кроме того, только наблюдение позволит аналитику увидеть предметную область, и, как известно, «лучше один раз увидеть, чем сто раз услышать».

Наблюдение за имитацией процесса проводят обычно также за рабочим местом эксперта, но сам процесс деятельности запускается специально для аналитика. Преимущество этой разновидности в том, что эксперт менее напряжен, чем в первом варианте, когда он работает на «два фронта» — и ведет профессиональную деятельность, и демонстрирует ее. Наблюдения за имитацией проводят также и в тех случаях, когда наблюдения за реальным процессом по каким-либо причинам невозможны (например, профессиональная этика врача-психиатра может не допускать присутствия постороннего на приеме).

Сеансы наблюдений могут потребовать от инженера по знаниям:

- овладения техникой стенографии для фиксации действий эксперта в реальном масштабе времени;
- ознакомления с методиками хронометража для четкого структурирования производственного процесса по времени;
- развития навыков «чтения по глазам», т. е. наблюдательности к жестам, мимике и другим невербальным компонентам общения;
- серьезного предварительного знакомства с предметной областью, так как из-за отсутствия «обратной связи» иногда многое непонятно в действиях экспертов.

Протоколы наблюдений после сеансов в ходе домашней работы тщательно расшифровываются, а затем обсуждаются с экспертом.

Таким образом, наблюдения — один из наиболее распространенных методов извлечения знаний на начальных этапах разработки. Обычно он применяется не самостоятельно, а в совокупности с другими методами.

Анализ протоколов «мыслей вслух»

Протоколирование «мыслей вслух» отличается от наблюдений тем, что эксперта просят не просто прокомментировать свои действия и решения, но и объяснить, как это решение было найдено, т. е. продемонстрировать всю цепочку своих рассуждений. Во время рассуждений эксперта все его слова, весь «поток сознания» протоколируется инженером по знаниям, при этом полезно отметить даже паузы и междометия. Иногда этот метод называют «вербальные отчеты» [Моргоев, 1988].

Вопрос об использовании для этой цели диктофонов является дискуссионным, поскольку сам факт записи иногда парализующе действует на эксперта, разрушая атмосферу доверительности.

Основной трудностью при протоколировании «мыслей вслух» является принципиальная сложность для любого человека объяснить процесс размышлений. Автор теории фреймов М. Минский считает, что «только как исключение, а не как правило, человек может объяснить то, что он знает» [Minsky, 1981]. Однако существуют люди, склонные к рефлексии, для которых эта работа является вполне до-

ступной. Следовательно, такая характеристика когнитивного стиля, как рефлексивность, является для эксперта более чем желательной.

Расшифровка полученных протоколов производится инженером по знаниям самостоятельно с коррекциями на следующих сеансах извлечения знаний.

Лекции

Лекция является самым старым способом передачи знаний. Но нас сейчас интересует не столько способность к подготовке и чтению лекций, сколько способность эту лекцию слушать, конспектировать и усваивать. Уже говорилось, что чаще всего экспертов не выбирают, и поэтому учить эксперта чтению лекции инженер по знаниям не сможет. В лекции эксперту предоставлено много степеней свободы для самовыражения; при этом необходимо сформулировать эксперту тему и задачу лекции. Например, тема цикла лекций «Постановка диагноза — воспаление легких», тема конкретной лекции «Рассуждения по анализу рентгенограмм», задача — научить слушателей по перечисленным экспертом признакам ставить диагноз воспаления легких и делать прогноз. При такой четкой постановке опытный лектор может заранее структурировать свои знания и ход рассуждений. От инженера по знаниям в этой ситуации требуется лишь грамотно законспектировать лекцию и в конце задать необходимые вопросы.

Студенты хорошо знают, что конспекты лекций одного и того же лектора у разных студентов существенно отличаются. Люди, умело ведущие конспект, обычно сильные студенты. Обратное неверно. В чем же заключается искусство ведения конспекта? В «помехоустойчивости» — записывать главное, опускать второстепенное, выделять фрагменты знаний (параграфы, подпараграфы), записывать только осмысленные предложения, уметь обобщать.

Хороший вопрос по ходу лекции помогает и лектору, и слушателю. Серьезные и глубокие вопросы могут существенно поднять авторитет инженера по знаниям в глазах эксперта.

Опытный лектор знает, что все вопросы можно условно разбить на три группы:

- умные вопросы, углубляющие лекцию;
- глупые вопросы или вопросы не по существу;
- вопросы «на засыпку», или провокационные.

Вопросы второго и третьего типа фактически бесполезны. Продолжительность лекции рекомендуется стандартная — от 40 до 50 минут и через 5–10 минут — еще столько же. Метод извлечения знаний в форме лекций, как и все пассивные методы, используют в начале разработки как эффективный способ быстрого погружения инженера по знаниям в предметную область.

В заключение несколько советов, как слушать лекции [Ребельский, 1989]:

1. К лекции подготовьтесь, т. е. познакомьтесь с предметной областью.

2. Слушайте с максимальным вниманием, для этого: устраните мешающие факторы (скрип двери, шорохи и т. д.); удобно устройтесь; поменьше двигайтесь.

3. Учитесь отдыхать во время слушания (например, когда лектор приводит цифры, которые можно взять из справочника).

4. Слушайте одновременно и лектора, и самого себя (параллельно с мыслями лектора по ассоциации возникают собственные мысли).

5. Слушайте и одновременно сокращенно записывайте, используя условные значки.

6. Расшифруйте записи лекции в тот же день.

7. Не спорьте с лектором в процессе лекции.

8. Рационально используйте перерывы в лекции для подведения итогов прослушанного.

Коммуникативные активные методы

Индивидуальные

Активные индивидуальные методы извлечения знаний на сегодняшний день — наиболее распространенные. В той или иной степени, к ним прибегают при разработке практически любой ЭС.

К основным активным методам можно отнести:

1) анкетирование;
2) интервью;
3) свободный диалог;
4) игры с экспертом.

Во всех этих методах активную функцию выполняет аналитик, который пишет сценарий и режиссирует сеансы извлечения знаний. Игры с экспертом существенно отличаются от трех других методов. Три оставшихся метода очень схожи между собой и отличаются лишь по степени свободы, которую может себе позволить инженер по знаниям при проведении сеансов извлечения знаний. Их можно назвать вопросными методами поиска знаний.

Анкетирование

Анкетирование — наиболее жесткий метод, т. е. наиболее стандартизированный. В этом случае инженер по знаниям заранее составляет вопросник или анкету, размножает ее и использует для опроса нескольких экспертов. Это основное преимущество анкетирования.

Сама процедура может проводиться тремя способами:

1) аналитик вслух задает вопросы и сам заполняет анкету по ответам эксперта;

2) эксперт самостоятельно заполняет анкету после предварительного инструктирования;

3) анкета размещается в Интернете или на отдельном компьютере и затем автоматически обрабатывается.

Выбор способа зависит от конкретных условий (например, от бюджета проекта, оформления анкеты, ее понятности, готовности эксперта). Второй и третий способ нам представляются предпочтительнее, так как у эксперта появляется неограниченное время на обдумывание ответов.

Если анализировать процесс непосредственного общения, то можно выделить три компоненты — участники, средства и ситуация общения. Основными факторами, на которые можно существенно повлиять при анкетировании, являются средства общения (в данном случае это вопросник) и ситуация общения.

Существует несколько универсальных рекомендаций при составлении анкет. Наибольший опыт работы с анкетами накоплен в социологии и психологии, поэтому часть рекомендаций заимствована из [Ноэль, 1978; Погосян, 1985; Тавокин, 2009]:

- анкета не должна быть монотонной и однообразной, т. е. вызывать скуку или усталость;
- анкета должна быть приспособлена к языку экспертов;
- следует учитывать, что вопросы влияют друг на друга, и поэтому последовательность вопросов должна быть строго продумана;
- желательно стремиться к оптимальной избыточности. Известно, что в анкете всегда много лишних вопросов, часть из них необходима — это так называемые контрольные вопросы (см. о них ниже), а другая часть должна быть минимизирована;
- анкета должна иметь «хорошие манеры», т. е. ее язык должен быть ясным, понятным, предельно вежливым.

Пример 2.3

Лишние вопросы появляются, например, в таких ситуациях. Фрагмент анкеты:

«В12. Считаете ли Вы, что для привлечения клиента можно применять материальное стимулирование?

В13. В каких формах Вы обычно это делаете?»

При отрицательном ответе на 12-й вопрос 13-й является лишним. Его можно избежать, усложнив вопрос.

«В12. Применяете ли материальное стимулирование для привлечения клиента, и если да, то в каких формах?»

В последнее время большую популярность приобретают сетевое анкетирование через опросы на Web-серверах (например, http://webanketa.com/, https://www.surveymonkey.ru/, Google Формы) и автоматизированные системы опросов (stand-alone). Это существенно упрощает обработку, но требует тщательной подготовки и вложения средств.

Интервью

Под *интервью* будем понимать специфическую форму общения инженера по знаниям и эксперта, в которой инженер по знаниям задает эксперту серию заранее подготовленных вопросов с целью извлечения знаний о предметной области. Наибольший опыт в проведении интервью накоплен, наверное, в журналистике и социологии. Большинство специалистов этих областей отмечают тем не менее крайнюю недостаточность теоретических и методологических исследований по тематике интервьюирования [Брейс, 2005; Ноэль, 1978; Шумилина, 1973].

Интервью близко тому способу анкетирования, когда аналитик сам заполняет анкету, занося туда ответы эксперта. Основное отличие интервью в том, что оно позволяет аналитику опускать ряд вопросов в зависимости от ситуации, вставлять новые вопросы в анкету, изменять темп, разнообразить ситуацию общения. Кроме этого, у аналитика появляется возможность «взять в плен» эксперта своим обаянием, заинтересовать его самой процедурой и тем самым увеличить эффективность сеанса извлечения.

Теперь несколько подробнее о центральном звене активных индивидуальных методов — о вопросах. Инженеры по знаниям редко сомневаются в своей способности задавать вопросы. В философии и математике эта проблема обсуждается с давних лет. Существует даже специальная ветвь математической логики — *эротетическая* логика (логика вопросов). Есть интересная работа Белнапа «Логика вопросов и ответов» [Белнап, Стил, 1981], но, к сожалению, использовать результаты, полученные логиками, непосредственно при разработке интеллектуальных систем не удается.

Все вопросительные предложения можно разбить на два типа [Шатуновский, 2001]:

- вопросы *с неопределенностью*, относящейся ко всему предложению («Действительно, введение бартера усложняет схему?»);
- вопросы *с неполной информацией* («При каких условиях необходимо включать кнопку?»), часто начинающиеся со слов «кто», что», «где», «когда» и т. д.

Это разделение можно дополнить классификацией, расширяющей описаную в [Шумилина, 1973] и представленной на рисунке 2.16.

Вербальные вопросы — это традиционные устные вопросы. *Невербальные или вопросы с использованием наглядного материала* разнообразят интервью и снижают утомляемость эксперта. В этих вопросах используют фотографии, рисунки и карточки. Например, эксперту предлагаются цветные картонные карточки, на которых выписаны признаки заболевания. Затем аналитик просит разложить эти карточки в порядке убывания значимости для постановки диагноза.

Открытый вопрос называет тему или предмет, оставляя полную свободу эксперту по форме и содержанию ответа («Не могли бы Вы рассказать, как Вы организуете подготовку выставки?»).

В *закрытом* вопросе эксперт выбирает ответ из набора предложенных («Укажите, пожалуйста, с чего Вы рекомендуете начать подготовку к выставке: а) рекламы; б) анализа рынка; в) прямого поиска спонсоров; г) сбора команды организаторов»). Закрытые вопросы легче обрабатывать при последующем анализе, но они более опасны, так как «закрывают» ход рассуждений эксперта и «программируют» его ответ в определенном направлении. При составлении сценария интервью полезно чередовать открытые и закрытые вопросы, особенно тщательно продумывать закрытые, поскольку для их составления требуется определенная эрудиция в предметной области.



Рис. 2.16
Классификация вопросов

Деление вопросов по функции на основные, зондирующие, контрольные связано с тем, что часто основные вопросы интервью, направленные на выявление знаний, не срабатывают — эксперт по каким-то причинам уходит в сторону от вопроса, отвечает нечетко.

Тогда аналитик использует *зондирующие* вопросы, которые направляют рассуждения эксперта в нужную сторону. Например, если не сработал основной вопрос: «Какие параметры определяют момент окончания процесса ферментации лизина?», аналитик начинает задавать зондирующие вопросы: «Всегда ли процесс ферментации длится 72 часа? А если он заканчивается раньше, как это узнать? Если он продлится больше, то что заставит микробиолога не закончить процесс на 72-м часу?» и т. д.

Контрольные вопросы применяют для проверки достоверности и объективности информации, полученной в интервью ранее («Скажите, пожалуйста, а московская школа психологов так же, как Вы, трактует шкалу „К“ опросника MMPI?» или «Рекомендуете ли Вы инъекции АТФ?» (АТФ — препарат, снятый с производства)). Контрольные вопросы должны быть «хитро» составлены, чтобы не обидеть эксперта недоверием (для этого используют повторение вопросов в другой форме, уточнения, ссылки на другие источники). «Лучше два раза спросить, чем один раз напутать» (Шолом-Алейхем).

При составлении вопросов следует учитывать, что языковые способности эксперта, как правило, ограничены, и вследствие скованности, замкнутости, робости он не может сразу высказать свое мнение и предоставить знания, которые от него требуются (даже если предположить, что он их четко для себя формулирует). Часто при «зажатости» эксперта используют не *прямые* вопросы, которые непосредственно указывают на предмет или тему («Как Вы относитесь к методике компании KPMG?», а *косвенные,* которые лишь косвенно указывают на интересующий предмет («Применяете ли Вы методику KPMG? Опишите, пожалуйста, основные проблемы»). Иногда аналитику приходится задавать несколько десятков косвенных вопросов вместо одного прямого.

Личный вопрос касается непосредственно личного индивидуального опыта эксперта («Скажите, пожалуйста, Иван Данилович, в Вашей практике Вы применяете аутсорсинг?»). Личные вопросы обычно активизируют мышление эксперта, «играют» на его самолюбии, они всегда украшают интервью.

Безличный вопрос направлен на выявление наиболее распространенных и общепринятых закономерностей предметной области («Что влияет на скорость процесса ферментации лизина?»).

И наконец, о наводящих вопросах. В принципе интервьюеру (в нашем случае инженеру по знаниям) рекомендуют быть беспристрастным, отсюда и вопросы его должны носить *нейтральный* характер, т. е. не должны указывать на отношение интервьюера к данной теме. Напротив, *наводящие* вопросы заставляют респондента (в данном случае эксперта) прислушаться или даже принять во внимание позицию интервьюера. Нейтральный вопрос: «Совпадают ли симптомы кровоизлияния в мозг и сотрясения мозга?» Наводящий вопрос: «Не правда ли, довольно трудно дифференцировать симптомы кровоизлияния в мозг?»

Кроме перечисленных выше, полезно различать и включать в интервью следующие вопросы [Ноэль, 1978; Тавокин, 2009]:

- контактные («ломающие лед» между аналитиком и экспертом);
- буферные (для разграничения отдельных тем интервью);
- оживляющие память экспертов (для реконструкции отдельных случаев из практики);
- «провоцирующие» (для получения спонтанных, неподготовленных ответов).

В заключение описания интервью укажем три основные характеристики вопросов [Шумилина, 1973], которые влияют на качество интервью:

- язык вопроса (понятность, лаконичность, терминология);

- порядок вопросов (логическая последовательность и немонотонность);
- уместность вопросов (этика, вежливость).

Вопрос в интервью — это не просто средство общения, но и способ передачи мыслей и позиции аналитика.

«Вопрос представляет собой форму движения мысли, в нем ярко выражен момент перехода от незнания к знанию, от неполного, неточного знания к более полному и более точному» [Лимантов, 1971]. Отсюда необходимость в протоколах фиксировать не только ответы, но и вопросы, предварительно тщательно отрабатывая их форму и содержание.

Очевидно, что любой вопрос имеет смысл только в контексте. Поэтому вопросы может готовить инженер по знаниям, уже овладевший ключевым набором знаний.

Вопросы имеют для эксперта диагностическое значение — несколько откровенных «глупых» вопросов могут полностью разочаровать эксперта и отбить у него охоту к дальнейшему сотрудничеству. Большое значение имеет порядок вопросов. В практике социологических и демографических опросов [Ноэль, 1984; Брейс, 2005] часто пользуются принципом «воронки», постепенно суживая предмет интервью, и переходя от более общих вопросов к детально-конкретным.

Свободный диалог

Свободный диалог — это метод извлечения знаний в форме беседы инженера по знаниям и эксперта, в которой нет жесткого регламентированного плана и вопросника. Это определение не означает, что к свободному диалогу не надо готовиться. Напротив, внешне свободная и легкая форма этого метода требует высочайшей профессиональной и психологической подготовки. Подготовка к свободному диалогу практически может совпадать с предлагаемой в [Шумилина, 1973] подготовкой к журналистскому интервью. Рисунок 2.17 графически иллюстрирует схему такой подготовки, дополненную в связи со спецификой инженерии знаний.



Рис. 2.17

Подготовка к извлечению знаний

Квалифицированная подготовка к диалогу помогает аналитику стать истинным драматургом или сценаристом будущих сеансов, т. е. запланировать гладкое течение

процедуры извлечения: от приятного впечатления в начале беседы переход к профессиональному контакту через пробуждение интереса и завоевание доверия эксперта. При этом для обеспечения желания эксперта продолжить беседу необходимо проводить «поглаживания» (терминология Э. Берна [Берн, 1992]), т. е. подбадривать эксперта и подтверждать всячески его уверенность в собственной компетентности (фразы-вставки: «Я Вас понимаю...», «...это очень интересно» и т. д.).

Так, в одном из исследований по технике ведения профессиональных журналистских диалогов [Matarozzo, Wettman, Weins, 1963] было экспериментально доказано, что одобрительное и поощрительное «хмыканье» интервьюера увеличивает длину ответов респондента. При этом одобрение должно быть искренним, как показал опрос интервьюеров Института демоскопии Германии: «Лучшая уловка — это избегать всяких уловок: относиться к опрашиваемому с истинным человеколюбием, не с наигранным, а с подлинным интересом» [Ноэль, 1978]. Чтобы расположить собеседника к разговору, сначала аналитику можно рассказать о себе, о работе, т. е. поговорить самому.

Мы уже писали о профессиональной пригодности инженеров по знаниям и необходимости предварительного психологического тестирования при подготовке инженеров по знаниям. Здесь только приведем каталог свойств идеального интервьюера [Ноэль, 1978]. На наш взгляд, это вполне подходящий образец портрета инженера по знаниям перед серией свободных диалогов: «Он должен выглядеть здоровым, спокойным, уверенным, внушать доверие, быть искренним, веселым, проявлять интерес к беседе, быть опрятно одетым, ухоженным». Хороший аналитик может личным обаянием и умением скрыть изъяны подготовки. Блестящая краткая характеристика интервьюера приведена в той же работе — *«общительный педант»*.

В свободном диалоге важно выбрать правильный темп или ритм беседы: без больших пауз, так как эксперт может отвлечься, но и без гонки, иначе быстро утомляются оба участника, и нарастает напряженность, кроме того, некоторые люди говорят и думают очень медленно. Умение чередовать разные темпы, напряжение и разрядку в беседе существенно влияет на результат.

Подготовка к диалогу так же, как и к другим активным методам извлечения знаний, включает составление плана проведения сеанса извлечения, в котором необходимо предусмотреть следующие стадии.

1. Начало беседы (знакомство, создание у эксперта «образа» аналитика, объяснение целей и задач работы).

2. Диалог по извлечению знаний.

3. Заключительная стадия (благодарность эксперту, подведение итогов, договор о последующих встречах).

Девизом для инженера по знаниям могут послужить взгляды одного из классиков отечественного литературоведения М.М. Бахтина [Бахтин, 1975]:

«Диалог — столкновение разных умов, разных истин, несходных культурных позиций, составляющих единый ум, единую истину, общую культуру».

Индивидуальные экспертные игры

Игрой называют такой вид человеческой деятельности, который отражает (воссоздает) другие ее виды [Комаров, 1989; Фопель, 2009]. При этом для игры характерны одновременно условность и серьезность.

Понятие экспертной игры или игры с экспертами в целях извлечения знаний восходит к трем источникам — это деловые игры, широко используемые при подготовке специалистов и моделировании [Борисова, Соловьева и др., 1988; Бурков, 1980; Комаров, 1989]; диагностические игры, описанные в [Алексеевская,

Недоступ, 1988; Гельфанд, Розенфельд, Шифрин, 1988; Продеус, Захрабова, 2004], и компьютерные игры, все чаще применяемые в обучении [Пажитнов, 1987; Панфилова, 2009].

В настоящее время в психолого-педагогических науках нет единой теоретической концепции деловых игр и других игровых методов обучения. Тем не менее на практике эти игры широко используются. Под *деловой* игрой чаще всего понимают эксперимент, где участникам предлагается производственная ситуация, а они на основе своего жизненного опыта, общих и специальных знаний и представлений принимают решения [Бурков, 1980]. Решения анализируются, вскрываются закономерности мышления участников эксперимента. Именно эта анализирующая часть деловой игры полезна для получения знаний. И если участниками такой игры становятся эксперты, то игра из деловой превращается в экспертную. Из трех основных типов деловых игр (учебных, планово-производственных и исследовательских) к экспертам ближе всего исследовательские, которые используются для анализа систем, проверки правил принятия решений.

Диагностическая игра — та же деловая игра, но применяемая конкретно для диагностики методов принятия решения в медицине (диагностика методов диагностики). Эти игры возникли при исследовании способов передачи опыта от опытных врачей новичкам. В нашем понимании диагностическая игра — это игра, безусловно, экспертная без всяких оговорок, только с жестко закрепленной предметной областью — медициной.

Плодотворность моделирования реальных ситуаций в играх подтверждается сегодня практически во всех областях науки и техники. Они развивают логическое мышление, умение быстро принимать решения, вызывают интерес у экспертов.

В соответствии с классификацией, введенной ранее, будем разделять экспертные игры на *индивидуальные экспертные игры* и *групповые* (*ролевые*) *экспертные игры*.

Кроме этого, продолжим и разовьем эту классификацию введением других критериев:

- использование специального оборудования;
- применение вычислительной техники.

Игры с тренажерами

Игры с тренажерами в значительной степени ближе не к играм, а к имитационным упражнениям в ситуации, приближенной к действительности.

Наличие тренажера позволяет воссоздать почти производственную ситуацию и понаблюдать за экспертом. Тренажеры широко применяют для обучения (например, летчиков или операторов атомных станций).

Компьютерные экспертные игры

Идея использовать компьютеры в деловых играх известна давно. Но только тогда, когда компьютерные игры взяли в плен практически всех пользователей, стала очевидной особая притягательность такого рода игр.

Существует несколько классификаций компьютерных игр, например, по жанрам, из журнала GAME.EXE:

- **Action/Arcade games (экшн/аркады).** Игры-действия. Требуют хорошего глазомера и быстрой реакции;
- **3D Action games («стрелялки»).** То же, что и экшн, но с активным использованием трехмерной графики;
- **Simulation games (симуляторы).** Базируются на моделировании реальной действительности и отработки практических навыков, например, в вождении автомобиля, пассажирского самолета, поезда, авиадиспетчера и даже

симуляторы рыбной ловли. Также популярны спортивные симуляторы — теннис, бокс и др.;

- **Strategy games (стратегические игры).** Требуют стратегического планирования и ответственности при принятии решений, например, развитие цивилизаций, соперничество миров, экономическая борьба. Особый класс стратегических игр — wargames (военные игры). В последнее время упор в 3D Action делается на многопользовательский режим (игру по сети);
- **головоломки/логические игры/Puzzles (настольные игры-головоломки).** Компьютерные реализации различных логических игр;
- **Adventure/Quest (приключенческие игры).** Обычно обладают разветвленным сценарием, красивой графикой и звуком. Управляя одним или несколькими персонажами, игрок должен правильно вести диалоги, разгадывать множество загадок и головоломок, замечать и правильно использовать предметы, спрятанные в игре;
- **Role-playing games RPG (ролевые игры).** Распространенный жанр, берущий свое начало в старых английских настольных играх. Существуют один или несколько персонажей, обладающих индивидуальными способностями и характеристиками. Им приходится сражаться с врагами, решать загадки. По мере выполнения этих задач у героев накапливается опыт, и по достижению определенного значения их характеристики улучшаются;
- **традиционные** (англ. Traditional) **и настольные** (англ. Board), **спортивные игры, текстовые и другие.**

Или по количеству пользователей:

- одиночные (синглплеер, англ. singleplayer);
- многопользовательские (англ. multiplayer);
- многопользовательские на одном компьютере (англ. Hot Seat, Splitscreen);
- многопользовательские онлайн-игры;
- массовые онлайновые (англ. Massively Multiplayer Online).

Следует отметить, что многие игры могут быть отнесены сразу к нескольким классам, и в целом эту классификацию нельзя считать строгой. Игры иногда полезны для развлечения экспертов перед сеансом извлечения знаний. Кроме того, очевидно, что экспертные игры, сочетая элементы перечисленных выше классов, могут успешно применяться для непосредственного извлечения знаний. Однако разработка и программная реализация такой игры потребуют существенных вложений временных и денежных ресурсов.

Индивидуальные экспертные игры

В этом случае с экспертом играет инженер по знаниям, который берет на себя какую-нибудь роль в моделируемой ситуации. Например, одному из авторов часто приходилось разыгрывать с экспертом игру «Учитель и ученик», в которой инженер по знаниям берет на себя роль ученика и на глазах у эксперта выполняет его работу (например, пишет психодиагностическое заключение), а эксперт поправляет ошибки «ученика».

Пример 2.4

Другая игра [Гельфанд, Розенфельд, Шифрин, 1988; Продеус, Захрабова, 2004] заставляет инженера по знаниям взять на себя роль врача, который хорошо знает больного, а роль консультанта играет эксперт. Консультант задает вопросы и делает прогноз о целесообразности применения того или иного вида лечения (в описанной игре это был прогноз целесообразности электростимульной терапии при сердечной аритмии). Игра «двух врачей» позволила выявить, что эксперту понадобилось всего 30 вопросов для успешного прогноза, в то время как

первоначальный вариант вопросника, составленный медиками для той же цели, содержал 170 вопросов.

Пример 2.5

Сначала эксперта просят написать обоснование для собственного прогноза. Например, почему он считает, что язва у больного *X* заживает. Накапливается несколько таких обоснований [Гельфанд, Розенфельд, Шифрин, 1988; Продеус, Захрабова, 2004], а через некоторое время эксперту зачитывают только его обоснование и просят сделать прогноз. Как правило, этого он сделать не может, т. е. обоснование (или его знания) было неполным. Эксперт дополняет обоснование, тем самым выявляются скрытые (для самого эксперта) пласты знаний. Так, в играх «Обоснование прогноза рецидива язвенного кровотечения» удалось выявить, что значимыми для прогноза являются всего три правила. Причем два правила входили в традиционно-диагностический вопросник, а третье было сформулировано во время игры.

Активные групповые методы

К групповым методам извлечения знаний относятся ролевые игры, дискуссии за «круглым столом» с участием нескольких экспертов и «мозговые штурмы». Основное достоинство групповых методов — это возможность одновременного «поглощения» знаний от нескольких экспертов, взаимодействие которых вносит в этот процесс элемент принципиальной новизны от наложения разных взглядов и позиций.

«Круглый стол»

Метод *круглого стола* (термин заимствован из журналистики) предусматривает обсуждение какой-либо проблемы из выбранной предметной области, в котором принимают участие с равными правами несколько экспертов. Обычно вначале участники высказываются в определенном порядке, а затем переходят к живой свободной дискуссии. Число участников дискуссии колеблется от трех до пяти-семи.

Большинство общих рекомендаций по извлечению знаний, предложенных ранее, применимо и к данному методу. Однако существует и специфика, связанная с поведением человека в группе.

Задача дискуссии — коллективно, с разных точек зрения, исследовать спорные гипотезы предметной области. Обычно эмпирические области богаты таким дискуссионным материалом. Для остроты на «круглый стол» приглашают представителей разных направлений и разных поколений, это также уменьшает опасность получения односторонних однобоких знаний.

Обмен мнениями по научным вопросам имеет давнюю традицию в истории человечества (античная Греция, Индия). До наших дней дошли литературные памятники обсуждения спорных вопросов (например, Протагор «Искусство спорить», работы софистов), послужившие первоосновой диалектики — науки вести беседу, спорить, развивать теорию. В самом слове дискуссия (от лат. *discussio* — исследование) содержится указание на то, что это метод научного познания, а не просто споры (для сравнения, полемика — от греч. *polemikos* — воинственный, враждебный).

Несколько практических советов по процедурным вопросам «круглого стола» из [Соколов, 1980; Фопель, 2010]. Перед началом дискуссии ведущему полезно:

- убедиться, что все правильно понимают задачу (т. е. происходит сеанс извлечения знаний);
- установить регламент;
- четко сформулировать тему.

«Мозговой штурм»

Активные групповые методы обычно используются в качестве острой приправы при извлечении знаний, сами по себе они не могут служить источником более или менее полного знания. Их применяют как дополнительные к традиционным индивидуальным методам (наблюдения, интервью и т. д.), для активизации мышления и поведения экспертов.

«Мозговой штурм», или «мозговая атака», — один из наиболее распространенных методов раскрепощения и активизации творческого мышления. Другие методы (метод фокальных объектов, синектика, метод контрольных вопросов [Шепотов, Шмаков, Крикун, 1985; Михалко, 2015]) применяются гораздо реже из-за меньшей эффективности.

Впервые этот метод был использован в 1939 г. в США А. Осборном как способ получения новых идей в условиях запрещения критики. Замечено, что боязнь критики мешает творческому мышлению, поэтому основная идея штурма — это отделение процедуры генерирования идей в замкнутой группе специалистов от процесса анализа и оценки высказанных идей.

Обычно за первые 30–40 минут высказывается более 50 идей. Регламент — до 2 минут на выступление. Самый интересный момент штурма — это наступление пика (ажиотажа), когда идеи начинают «фонтанировать». Этот пик имеет теоретическое обоснование в работах выдающегося швейцарского психолога и психиатра З. Фрейда о бессознательном. При последующем анализе всего лишь 10–15% идей оказываются разумными, но среди них бывают весьма оригинальные. Оценивает результаты обычно группа экспертов, не участвовавшая в генерации.

Ведущий «мозгового штурма» — инженер по знаниям — должен свободно владеть аудиторией, подобрать активную группу экспертов — «генераторов», не зажимать плохие идеи — они могут служить катализаторами хороших. Искусство ведущего — это искусство задавать вопросы аудитории, «подогревая» генерацию. Вопросы служат «крючком» [Шепотов, Шмаков, Крикун, 1985], которым извлекаются идеи. Основной девиз штурма — «чем больше идей, тем лучше». Фиксация хода сеанса — традиционная (протокол или диктофон).

Ролевые игры

Ролевые игры в группе

Групповые игры предусматривают участие в игре нескольких экспертов. К такой игре обычно заранее составляется сценарий, распределяются роли, к каждой роли готовится портрет-описание (лучше с девизом) и разрабатывается система фиксации использованных знаний игроков [Борисова, Соловьева и др., 1988; Панфилова, 2009].

Пример 2.6

Деловую игру можно превратить в экспертную. Например, для извлечения знаний из специалистов предприятия, разрабатывающих маркетинговую политику, можно разбить в игре экспертов на три игровые группы: $ЛПР_1$ — группа планирования; $ЛПР_2$ — группа менеджеров; Э — группа экспертизы по оцениванию действия $ЛПР_1$ и $ЛПР_2$. Группам $ЛПР_1$ и $ЛПР_2$ задавать различные ситуации на рынке и тщательно протоколировать их споры, рассуждения, аргументы по принятию решений.

Обычно в игре принимают участие от трех до шести экспертов, если их больше, то можно разбить всех на несколько конкурирующих игровых бригад. Элемент состязательности оживляет игру. Например, чей диагноз окажется ближе к истинному, чей план рациональнее использует ресурсы, кто быстрее определит причину неисправности в техническом блоке.

2.6. ИНТЕЛЛЕКТУАЛЬНЫЙ АНАЛИЗ ДАННЫХ

Расширяя понятие «формирование знаний из данных» следует рассказать о системах интеллектуального анализа данных, накапливаемых в современных хранилищах данных. Этот термин является приближенным переводом с английского языка терминов Knowledge Discovery in Databases и Data Mining.

> Интеллектуальный анализ данных (ИАД) обычно определяют, как технологию извлечения новых знаний из хранилищ данных путем выявления закономерностей в накопленной информации.

ИАД — это процесс обнаружения в исходных данных ранее неизвестных, нетривиальных, практически полезных и доступных интерпретации знаний, необходимых для принятия решений в различных сферах человеческой деятельности. ИАД использует сложный статистический анализ и моделирование для нахождения моделей и отношений, скрытых в базе данных, — таких отношений, которые не могут быть найдены обычными методами.

Интеллектуальные средства анализа возникли и развиваются на стыке таких дисциплин, как статистика, распознавание образов, искусственный интеллект, теория информации, теория баз данных и др. Отсюда обилие методов и алгоритмов, реализованных в различных действующих системах ИАД. Многие из таких систем интегрируют сразу несколько подходов.

Эта технология не заменяет аналитиков и менеджеров, а дает им мощный и современный инструмент для поддержания принятия решений. Обзор технологий ИАД дан с использованием материала А.В. Блинниковой [Мильнер, Румянцева, Смирнова, Блинникова, 2006].

Процессы ИАД можно разделить на три большие группы: поиск зависимостей, прогнозирование, анализ аномалий.

Поиск зависимостей состоит в просмотре базы данных с целью автоматического выявления зависимостей. Проблема заключается в отборе действительно важных зависимостей из огромного числа существующих в БД.

Прогнозирование предполагает, что пользователь может предъявить системе записи с незаполненными полями и запросить недостающие значения. Система сама анализирует содержимое базы и делает правдоподобное предсказание относительно этих значений.

Анализ аномалий — это процесс поиска нетипичных данных, сильно отклоняющихся от устойчивых зависимостей.

Особенно широко эта технология используется в финансовых и бизнес-приложениях.

Традиционно выделяют следующие этапы ИАД (рис. 2.18).

1. Подготовка исходного набора данных. Этот этап включает создание набора данных, в том числе из различных источников, выбор обучающей выборки и т. д.

2. Предобработка данных. Данные могут содержать пропуски, шумы, аномальные значения и т. д. Кроме того, данные могут быть избыточны, недостаточны и т. д. В некоторых задачах требуется дополнить данные некоторой априорной информацией. Данные должны быть качественны и корректны с точки зрения используемого метода. Более того, иногда размерность исходного пространства может быть очень большой, и тогда желательно применение специальных алгоритмов понижения размерности.

3. Трансформация, нормализация данных. Этот этап необходим для тех методов, которые требуют, чтобы исходные данные были в каком-то определенном виде.

4. Анализ данных. На этом этапе применяются различные алгоритмы для нахождения знаний: нейронные сети, деревья решений, алгоритмы кластеризации и установления ассоциаций.

5. Постобработка данных. Это интерпретация результатов и применение полученных знаний в бизнес-приложениях.

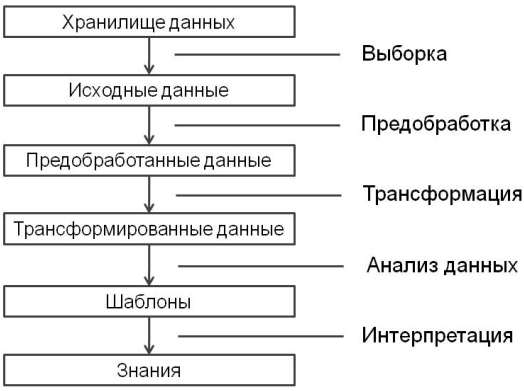


Рис. 2.18

Этапы интеллектуального анализа данных

Целью технологии ИАД является производство нового знания, которое пользователь может в дальнейшем применить для улучшения результатов своей деятельности.

У профессионалов обработки данных часто возникает вопрос о разнице между средствами ИАД и средствами оперативной аналитической обработки данных (On-Line Analytical Processing, OLAP). OLAP-технологии обычно применяются на ранних стадиях процесса ИАД. Это позволяет лучше понять полученные данные, помогает аналитикам выявить актуальные порции данных, а технологии ИАД обогащают эту функциональность.

Средства ИАД отличаются от средств OLAP тем, что они позволяют выдвигать новые гипотезы о характере неизвестных, но реально существующих отношений между данными. Кроме того, средства ИАД вместо проверки предполагаемых взаимозависимостей на основе имеющихся данных могут формировать модели, позволяющие количественно оценить степень влияния исследуемых факторов. В целом инструментарий оперативной добычи и аналитической обработки данных позволяет проводить развернутый анализ информации, что, в свою очередь, позволяет получить более эффективный результат процесса обнаружения новых знаний в хранилищах данных.

Обратимся к основным методам, которые используются для проведения интеллектуального анализа данных. К ним относятся: нейронные сети, деревья решений, индукция правил.

Кроме этих методов существует еще несколько дополнительных: системы рассуждения на основе аналогичных случаев, нечеткая логика, генетические алгоритмы, алгоритмы определения ассоциаций и последовательностей, анализ с избирательным действием, логическая регрессия, эволюционное программирование, визуализация данных.

Нейронные сети. Это большой класс систем, архитектура которых имеет аналогию с построением нервной ткани из нейронов. Нейросетевые технологии

предоставляют сегодня широкие возможности для решения задач прогнозирования, обработки сигналов и распознавания образов. По сравнению с традиционными методами математической статистики, классификации и аппроксимации эти технологии обеспечивают достаточно высокое качество решений при меньших затратах. Они позволяют выявлять нелинейные закономерности в сильно зашумленных неоднородных данных, дают хорошие результаты при большом числе входных параметров и обеспечивают адекватные решения при относительно небольших объемах данных.

Деревья решений являются одним из наиболее популярных подходов к решению задач ИАД. Они создают иерархическую структуру классифицирующих правил типа «если... то...» (if — then), имеющую вид дерева. Для принятия решения, к какому классу отнести объект или ситуацию, требуется ответить на вопросы, стоящие в узлах этого дерева, начиная с его корня. Вопросы имеют вид: «Значение параметра Л больше Х?» Если ответ положительный, осуществляется переход к правому узлу следующего уровня, если отрицательный — к левому узлу; затем снова следует вопрос, связанный с соответствующим узлом. Популярность подхода связана как бы с наглядностью и понятностью. Но деревья решений принципиально не способны находить «лучшие» (наиболее полные и точные) правила в данных. Они реализуют примитивный принцип последовательного просмотра признаков и «цепляют» фактически осколки настоящих закономерностей, создавая лишь иллюзию логического вывода.

Индукция правил создает неиерархическое множество условий, которые могут перекрываться. Она осуществляется путем генерации неполных деревьев решений, а для того, чтобы выбрать, какое из них будет применено к входным данным, используются статистические методы.

Идея *систем рассуждения на основе аналогичных случаев* (прецедентов) крайне проста. Для того чтобы сделать прогноз на будущее или выбрать правильное решение, эти системы находят в прошлом близкие аналоги наличной ситуации и выбирают тот же ответ, который был для них правильным. Поэтому данный метод называют еще методом «ближайшего соседа» (nearest neighbour). Системы рассуждения на основе аналогичных случаев показывают очень хорошие результаты при решении самых разнообразных задач. Главный их минус заключается в том, что они вообще не создают каких-либо моделей или правил, обобщающих предыдущий опыт, — в выборе решения они основываются на всем массиве доступных исторических данных, поэтому невозможно сказать, на основе каких конкретно факторов эти системы строят свои ответы.

Системы визуализации многомерных данных. В подобных системах основное внимание сконцентрировано на дружелюбности пользовательского интерфейса, позволяющего ассоциировать с анализируемыми показателями различные параметры диаграммы рассеивания объектов (записей) базы данных. К таким параметрам относятся цвет, форма, ориентация относительно собственной оси, размеры и другие свойства графических элементов изображения. Кроме того, системы визуализации данных снабжены средствами, удобными для масштабирования и вращения изображений.

Важным фактором эффективности применения методов ИАД являются:

- качество самих данных;
- мощность используемого программного обеспечения (развитые средства формирования запросов и визуализации результатов, наличие графического инструментария и т. д.);
- мастерство аналитика, который участвует в процессе построения модели.

Области применения ИАД чрезвычайно широки — от прогнозирования результатов маркетинга до анализа эффективности продаж и оценки емкости рынков.

Очевидно, что перечисленные области применения актуальны практически для всех отраслей бизнеса:

- банковское дело (оценка кредитных рисков, оценка закладных, выявление злоупотреблений с кредитными карточками, и пр.);
- страхование (выявление профилей пользователей, оценка эффективности региональных отделений, вероятность подачи заявки на выплату страховки и др.);
- финансовые рынки (прогнозирование, анализ портфелей, моделирование индексов);
- производство (прогнозирование спроса, контроль качества, оценка дизайна продукции).

Глава 3

СТРУКТУРИРОВАНИЕ ЗНАНИЙ

3.1. ИЕРАРХИЧЕСКОЕ СТРУКТУРИРОВАНИЕ

Ключевым моментом в процессе структурирования знаний является формирование понятийной структуры. Понятийная структура — это упорядоченная система основных понятий предметной области. Наличие такой структуры обусловлено постулатом когнитивной психологии о взаимосвязи понятий в памяти человека. Так психологи считают, что у человека существует семантическая (или понятийная) сеть, объединяющая отдельные термины во фрагменты, фрагменты в сценарии и т. д. Большинство специалистов по когнитивным наукам от нейрофизиологов до лингвистов утверждают, что основная особенность естественного интеллекта и памяти в частности — это именно связанность всех понятий в сеть. Поэтому для разработки базы знаний нужен не словарь, а «энциклопедия» [Шенк, Бирнбаум, Мей, 1989], в которой все термины объяснены в словарных статьях со ссылками на другие термины.

Таким образом, лингвистическая работа инженера по знаниям заключается в построении таких связанных фрагментов с помощью «сшивания» терминов. Фактически эта работа является подготовкой к этапу концептуализации, где это «шитье» (по Шенку — КОП, *концептуальная организация памяти* [Шенк, Хантер, 1987]) приобретает некоторый законченный вид.

Тщательная работа аналитика и эксперта позволяет определять иерархии в понятийных структурах. Такие структуры имеют большое гносеологическое и дидактическое значение, и в последнее годы для них используется специальный термин — «онтологии» [Gruber, 1993; Guarino, 1995] (см. далее часть 2 книги). Следует заметить, что эта иерархическая организация хорошо согласуется с теорией универсального предметного кода (УПК) [Горелов, 1987; Жинкин, 1982], согласно которой при мышлении используются не языковые конструкции, а их коды в форме некоторых абстракций, что, в общем, согласуется с результатами когнитивной психологии [Величковский, 2006].

Иерархия абстракций — это глобальная схема, которая может быть положена в основу концептуального анализа структуры знаний любой предметной области. Лингвистический эквивалент иерархии — иерархия понятий, которую необходимо построить в понятийной структуре, формируемой инженером по знаниям (см. рис. 3.1).

Выбрав уровень представления, можно строить понятийную структуру, жестко задав ее границы. Следует подчеркнуть, что работа по составлению словаря и понятийной структуры требует лингвистического «чутья», легкости манипулирования терминами и богатого словарного запаса инженера по знаниям, так как зачастую аналитик вынужден самостоятельно разрабатывать словарь признаков. Чем богаче и выразительнее получается общий код, тем полнее база знаний.

Аналитик вынужден все время помнить о трудности передачи образов и представлений в вербальной форме. Полезными тут оказываются свойства многозначности слов естественного языка. Часто инженеру по знаниям приходится подсказывать слова и выражения эксперту, и такие новые лексические конструкции оказываются полезными.

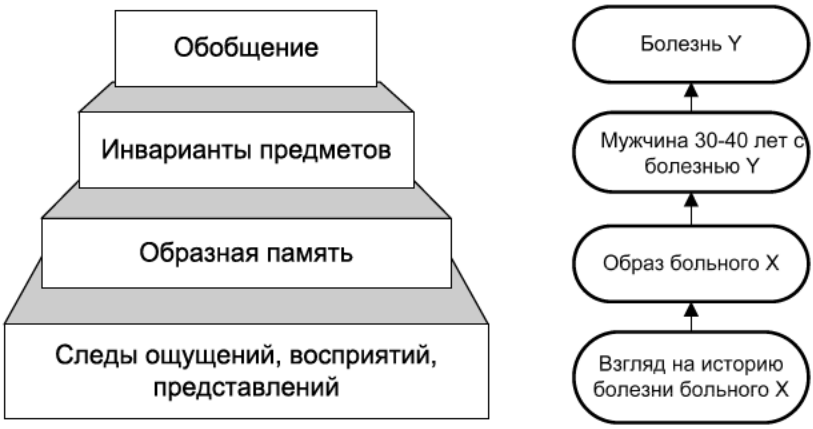


Рис. 3.1

Иерархия представлений

Способность к словесной интерпретации зависит и от гендерных отличий. Установлено, что традиционно женщины придают большую значимость невербальным компонентам общения, а в вербальных компонентах имеют более обширный алфавит признаков. К тому же существуют половые различия восприятия не только в бытовой сфере, но и в профессиональной. Следовательно, у эксперта-мужчины и у эксперта-женщины могут существенно отличаться алфавиты для вербализации признаков воспринимаемых объектов, а главное — критерии классификации (рис. 3.2 и 3.3).



Рис. 3.2

Иерархия понятий посуды (эксперт-женщина)

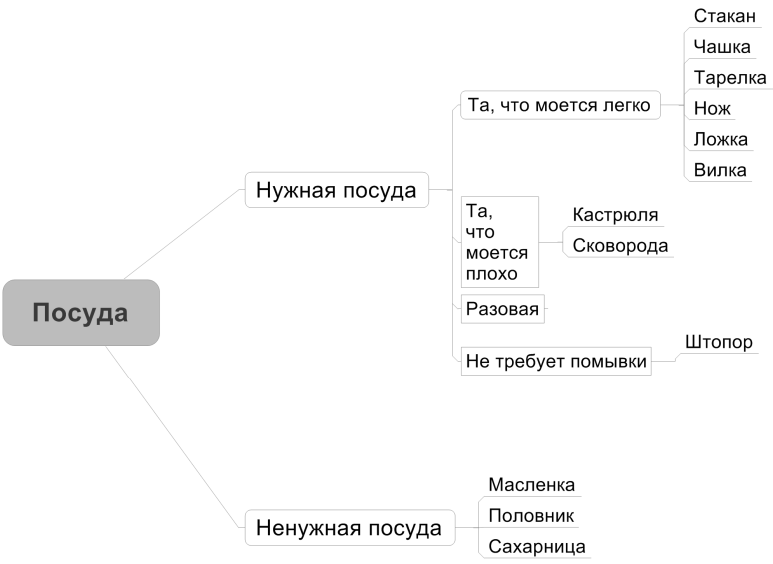


Рис. 3.3

Классификация посуды (эксперт-мужчина)

3.2. ВИЗУАЛЬНЫЕ МЕНТАЛЬНЫЕ МОДЕЛИ И ИХ КЛАССИФИКАЦИИ

При структурировании знаний и формировании иерархий полезным инструментом являются различные графические диаграммы (деревья, графы, сети и пр.). Визуализация всегда считалась мощным инструментами познания (mind tool), т. е. средством, предназначенным для организации и облегчения процесса познания. Любые графические модели от географической карты до модели бизнес-процесса являются инструментами познания, если они помогают сформулировать и объяснить природу и структуру явлений. Некоторые науки принципиально основываются на визуальных представлениях (геометрия, география, частично физика и химия). Активно используются графические представления в современных науках об информации [Загорулько, 2012; Романовский, Кознов, 2006; Рыбина, 2010; Тельнов, 2008]. Однако в гуманитарных науках они используются пока достаточно редко.

Исключением является менеджмент, где последние десятилетия активно внедряются языки моделирования бизнес-процессов, табличные представления и различные диаграммы. В менеджменте как междисциплинарной науке разные виды визуализации позволяют рассмотреть предметную область с различных сторон и точек зрения, используя на первый взгляд противоречивые, но, на самом деле, взаимодополняющие парадигмы.

На практике используются более сотни методов визуального структурирования знаний — от традиционных «и-или» графов, сетей Петри, диаграмм бизнес-процессов до модных схем «метро», «стратегических» карт (roadmaps), лучевых схем-пауков (spiders) и каузальных цепей. Такое многообразие обусловлено существенными различиями в природе, особенностях и свойствах знаний различных предметных областей. Сегодня любая попытка перечисления методов визуализации информации с целью структурирования практически обречена на провал. Всегда можно найти новый метод, не вошедший в список.

Хороший аналитик должен владеть широкой палитрой методов, всякий раз подбирая наиболее адекватный рассматриваемой предметной области. Все чаще визуальные языки моделирования используются в менеджменте [Lengler, Eppler, 2007], управлении знаниями [Кудрявцев, 2010], научных исследованиях [Хорошевский, 2012], учебном процессе [Гаврилова и др., 2011].

На сегодняшний день существует достаточно большое количество языков моделирования. Каждый из них имеет свою специфику и подходит для решения определенных задач визуализации. Однако палитра используемых на данный момент на практике языков моделирования ограничена, а выбор конкретного языка случаен и зачастую ошибочен. Пожалуй, в первую очередь это связано с отсутствием обобщающих принципов использования языков и их классификации. Существующая литература, а также программные продукты, предназначенные для создания визуальных моделей на компьютере, предлагают неоднородные и достаточно запутанные классификации языков. Как исключение можно упомянуть работу Бабкина и соавторов, в которой они проводят сравнительный анализ языков бизнес-моделирования [Бабкин и др., 2010; Бабкин и др., 2011]. Однако набор рассматриваемых языков в данных работах достаточно ограничен.

Данная ситуация ограничивает распространение языков моделирования вне узкого круга специалистов (таких как, например, бизнес-аналитики) и делает задачу выбора конкретного языка слишком сложной для широкого круга работников интеллектуального труда, в частности преподавателей.

Визуализация знаний играет важную роль при получении и передаче знаний [Кознов, 2004; Eppler, Burkhard, 2007; Margulies, Valenza, 2005]. Визуальные языки хорошо дополняют технологии онтологического (или концептуального) моделирования, делая содержимое онтологий понятным и наглядным не только для экспертов и аналитиков, но также и для новичков (подробнее про онтологии см. часть 2 книги).

Как отмечает ряд профессионалов в области дистанционного образования, визуализация играет значительную роль для улучшения усвоения материала [Vallano, 2012; Singh, 2011]. Визуализация материала не только упрощает его восприятие, но и повышает вовлеченность учеников в процесс обучения, что достаточно сложно достигнуть при отсутствии прямого контакта с преподавателем.

На рисунке 3.4 представлена одна из многочисленных попыток классификации методов визуализации, выполненная Ленглером и Эпплером [Lengler, Eppler, 2007].

Их подход внешне выглядит систематическим, напоминая таблицу Менделеева, однако, по сути, он просто описывает методы по нескольким важным признакам:

А. Природа объектов визуализации (данные, информация, концепты, стратегии, метафоры и комплексы) выделены цветом фона ячейки таблицы.

Б. Характер объекта (структура или процесс) показан цветом двухбуквенной метки.

В. Уровень грануляции (обзорный или детальный) отмечен символами ☼и ◘.

Г. Характер мышления (дивергентный и конвергентный) помечен символами < > и > <.

Последний признак требует пояснений: под дихотомией «дивергентное / конвергентное мышление» понимается разница в конкретном (конвергентном) и креативном (дивергентном) способах мышления [Hudson, 1967].

Другой критерий классификации, основанный на неформальном описании типов знаний, был предложен в работе [Гаврилова, Гулякина, 2008], позднее он был развит в работах [Гаврилова, Кудрявцев и др., 2013; Kudryavtsev, Gavrilova, 2016].

Эта классификация впервые устанавливает взаимосвязь между типом знаний, которые нужно представить, и видом модели.

При анализе любой сложной предметной области требуется взгляд с разных точек зрения — необходимо представить ЧТО-, КАК-, ЗАЧЕМ-, ПОЧЕМУ-, КТО-, ГДЕ- и КОГДА-знания [Гаврилова, Яшин, 1995]. Новая классификация помогает делать обоснованный выбор определенного языка для наиболее адекватного представления знаний. Данная задача актуальна как для преподавателей и для широкого круга профессионалов из различных областей как естественных, так и гуманитарных наук — моделирования предприятия (enterprise modeling), аналитиков бизнес-архитетуры (business architecture), дистанционного обучения (distance learning).

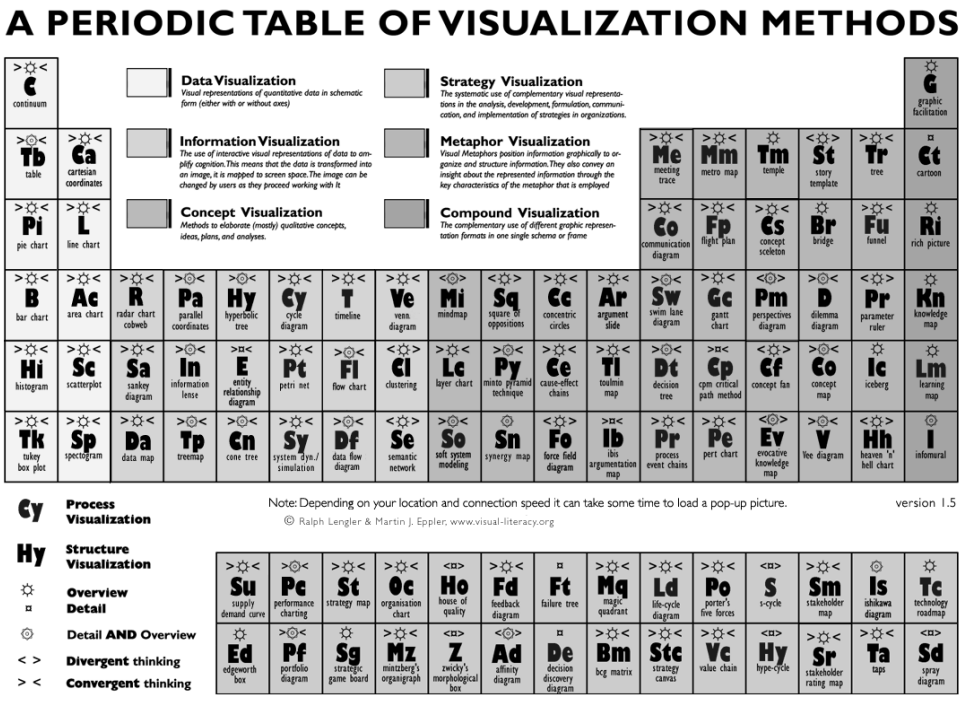


Рис. 3.4

Таблица Ленглера и Эпплера

[Lengler R., Eppler M., 2007. http://www.visual-literacy.org/periodic_table/periodic_table.html]

Также она полезна для широкого круга специалистов бизнес-информатики и ИКТ.

Для неформального описания типов знаний предлагается использовать вопросы проверки компетентности (competency questions) [Gruninger, Fox, 1995].

Рисунок 3.5 иллюстрирует предлагаемую новую классификацию вопросов по типу знаний. Список вопросов может быть существенно расширен применительно к конкретной задаче. Рисунок 3.6 уточняет, какие понятия и типы связей между ними относятся к тому или иному типу знаний, а также присутствуют в соответствующих визуальных языках моделирования. Классификация на рисунке 3.7 представляет расширение классификации из работы [Гаврилова, Гулякина, 2008] и сопоставляет наиболее распространенные в бизнес-приложениях

визуализации с предложенной типологией знаний. Данная классификация была получена путем семантического анализа метамоделей языков моделирования с использованием вопросов проверки компетентности и шаблонов проектирования онтологий [Kudryavtsev, Gavrilova, 2011].

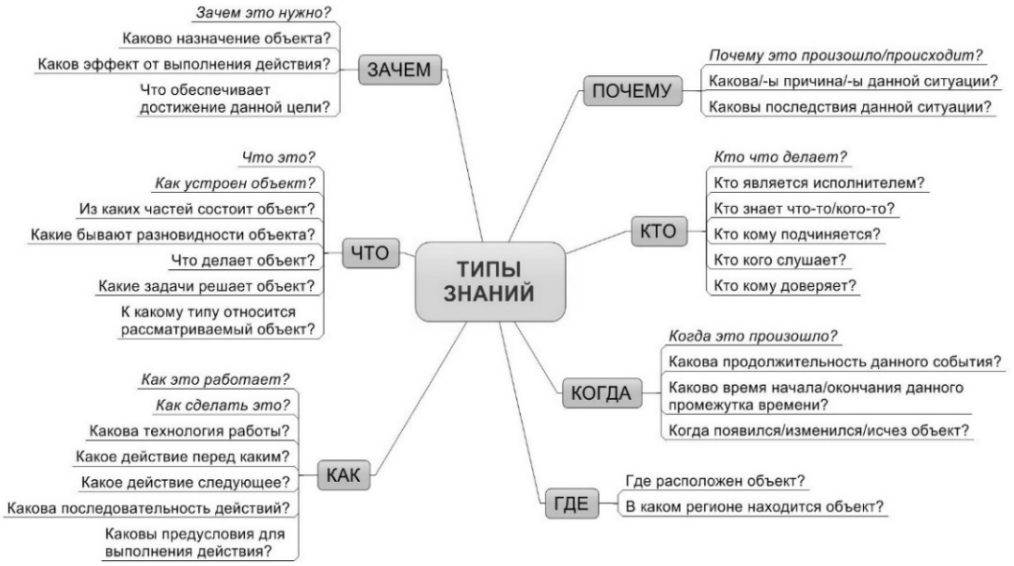


Рис. 3.5

Описание типов знаний с помощью вопросов проверки компетентности

Для успешного выбора метода визуализации необходимо ответить на несколько вопросов [Eppler, Burkhard, 2007].

- Какой тип знаний необходимо визуализировать?
- Какова цель визуализации знаний?
- Для кого необходимо визуализировать знания?
- В каких условиях будет использоваться визуализация: место размещения, канал передачи?
- В каком виде/формате необходимо представить знания?

Классификация на рисунке 3.7 устанавливает связь между ответом на первый из вышеуказанных вопросов и методом визуализации. Рассмотрим сценарий: любой преподаватель или специалист (не профессиональный бизнес-аналитик или инженер по знаниям) ищет визуальный язык моделирования, обладая неформализованным пониманием своих потребностей и желая доходчиво и наглядно отобразить свои знания или проверить знания и компетенции других людей. В таком случае можно предложить практический рецепт использования данного подхода.

А) Из рисунка 3.5 выбрать, какой вопрос наиболее точно отражает потребность и определить тип знаний, к которому относится данный вопрос. Уточнить выбор типа знаний можно с помощью рисунка 3.6, оценив какие взаимосвязи и между какими понятиями хочется отразить.

Б) Определить какие визуальные языки можно использовать для представления требуемого типа знаний по рисунку 3.7.

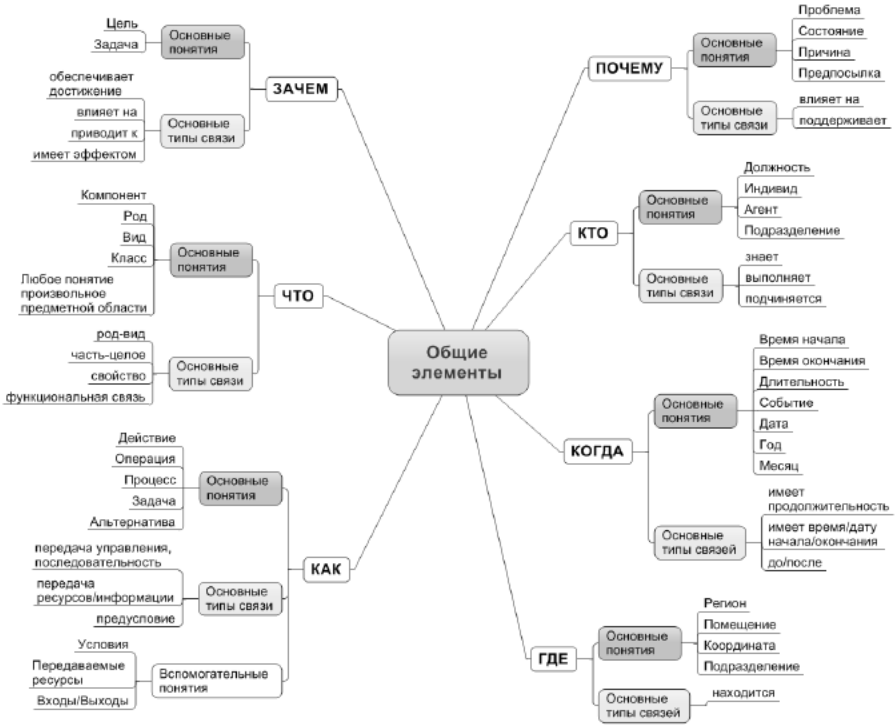


Рис. 3.6

Общие элементы для языков моделирования и типов знаний

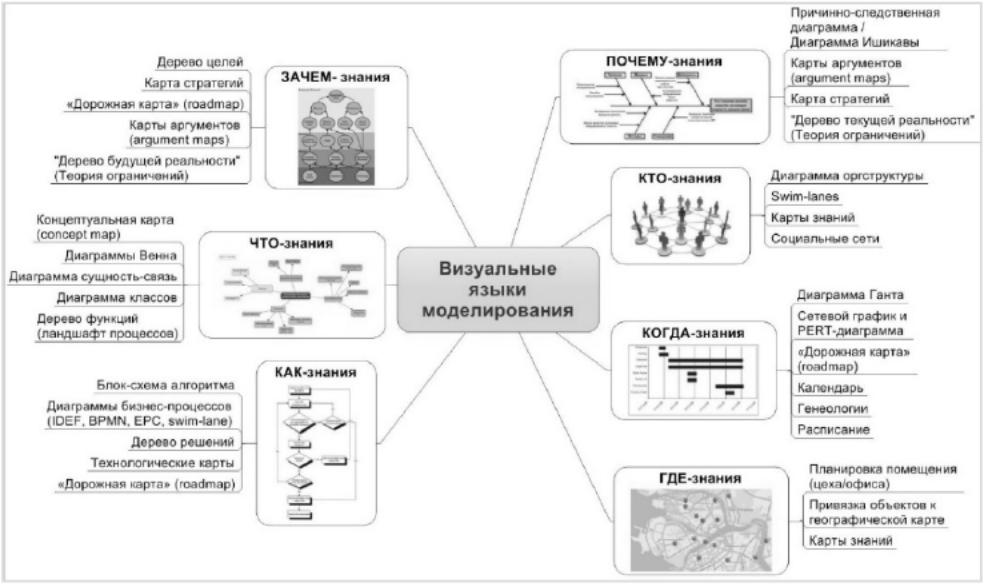


Рис. 3.7

Распределение визуальных языков моделирования по типам знаний

3.3. ТИПЫ ЗНАНИЙ И ВИДЫ ДИАГРАММ

ЧТО-знания

Практически во всех задачах ЧТО-знания являются определяющими и первоочередными. Особенно это важно в преподавании. Классический способ преподавания предполагает, что сначала дается определение изучаемого понятия, затем изучаются его свойства и лишь затем — взаимодействие с другими объектами или понятиями. Именно поэтому для визуализации ЧТО-знаний придумано множество языков. Рассмотрим основные из них.

Интеллект-карты

Интеллект-карты (mindmaps) — один из наиболее привлекательных и простых способов отображения понятийных структур. Хотя автором идеи интеллект-карт считается Тони Бьюзен [Бьюзен, Бьюзен, 2003], в России похожий метод успешно использовался в педагогике уже в 20-е годы [Глаголева, 1926] (рис. 3.8). Интеллект-карты — это иерархические диаграммы, используемые для представления идей, проектов, заданий, которые связаны с центральным ключевым понятием и организованы радиально вокруг него. Рисунки 3.9–3.12 иллюстрируют понятие интеллект-карты.

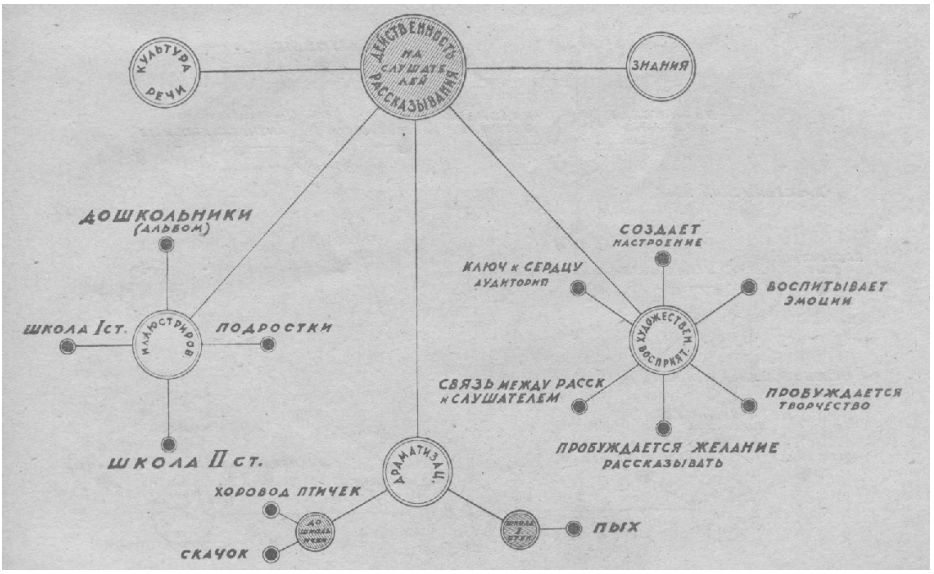


Рис. 3.8

Российская предшественница интеллект-карт

Идея интеллект-карт (и-карт) заключается в использовании и совмещении функции левого и правого полушарий для достижения целостного и наглядного представления рассматриваемого понятия. Фактически, это переход от последовательного (текстового) изложения к сетевому (образному). Бьюзен использовал также некоторые идеи из теории межполушарной асимметрии [Sperry, 1980] и гештальт-психологии [Вертгеймер, 1987].

Сперри обнаружил разделение функций левого (логического) и правого (образного) полушария. Основные интеллектуальные функции (по Сперри) представлены в таблице 3.1.

Позднее было установлено, что полушария могут иногда подменять друг друга и в целом все не так однозначно [Спрингер, Дейч, 1983; Черниговская, Деглин, 1986; Брагина, Доброхотов, 1988].

Таблица 3.1

Интеллектуальные функции полушарий головного мозга

Левое полушарие	Правое полушарие
Речь, чтение,	Пространственная ориентация
Анализ, логика	Гештальт (целостное восприятие)
Линейные представления	Трехмерное восприятие
Операции с последовательностями	Воображение, мечты
Операции с числами	Ритм, цвет

И-карта — графическое выражение процесса радиантного мышления. И-карты имеют 4 отличительные черты [Бьюзен, 2002]:

- объект внимания/изучения кристаллизован в центральном образе;
- основные темы, связанные с объектом изучения, расходятся от центрального образа в виде ветвей, которые поясняются ключевыми словами или образами;
- вторичные идеи также ветвятся;
- ветви формируют связанную узловую структуру.

Бьюзен сформулировал идею еще в 1970-е гг. в качестве компактного средства организации конспектов, позднее он понял, что метод гораздо шире и может использоваться как мощное орудие мышления применительно к научной работе, инновациям, бизнес-идеям, политическим дискуссиям, мозговому штурму, педагогике и пр.

Идея и-карт сильна еще и тем, что имеет серьезный нейрофизиологический базис. Человеческий мозг обладает выраженной сетевой структурой [Анохин, 1979]. Естественные нейронные сети мозга включают триллион нейронов, каждый из которых может связываться примерно с 10 000 ближайших соседей. И хотя природа их взаимодействия исследована далеко не полностью, так называемый «радиантный» характер передачи возбуждения от центра на периферию доказан.

И-карты применяются, с одной стороны, с целью генерирования, визуализации, структурирования и классификации идей, с другой стороны, с целью облегчения (ускорения) процесса обучения [CheiChang, 2008; Гаврилова, Лещева, Страхович, 2012], процесса разрешения проблемы или принятия решения [Роэм, 2009; Hverele, 2009; Jeffery, Maes & Bratton-Jeffery, 2005], а также при составлении и написании различных документов [Margulies, Valenza, 2005].

Их можно использовать как для объяснения, так и для проверки усвоенного материала. В этом случае структура и-карты служит критерием понимания изучаемого предмета. Среди и-карт, созданных студентами, встречаются диаграммы, демонстрирующие частичное (или полное) непонимание дисциплины, что является хорошей обратной связью для преподавателя.

Благодаря упрощенному и ясному изображению идей в графическом (радиальном и нелинейном) виде и-карты иногда используют для активизации «мозгового штурма» при решении организационных задач и осуществлении планирования [Lim & Klein, 2006]. В образовательном процессе данная модель применения и-карт может быть полезна, например, при выполнении учениками групповых проектов, анализе кейсов и т. д.

Создание и-карты начинается с объекта изучения, который располагается в центральном узле. Затем основные темы, связанные с объектом изучения, расходятся от центрального образа в виде ветвей, при этом иерархически детализируются, формируя связную смысловую структуру.

Важными элементами и-карт являются графические образы, цвет и различная величина шрифтов на разных уровнях иерархии, — все это позволяет четче и яснее отразить структуру карты, ускорить и упростить чтение и-карты. На рисунке 3.9 представлен пример и-карты для понятия «Окно» (предположим, это продукция компании), описывающий виды окон, их конструктивные элементы, предназначение и характеристики.

Поскольку и-карты позволяют представлять древовидные структуры произвольного содержания, они также могут использоваться не только для работы с ЧТО-знанием, но и для представления знаний других типов.

За последние 5 лет на рынке программного обеспечения появились десятки средств для быстрого и красивого мгновенного формирования и-карт (майнд мэппинга), например:

- Freeplane;
- IMindmap (by Tony Buzan);
- MindMapper Pro;
- MindGenius Business;
- Visual Mind;
- Mind Pad;
- Mind manager;
- Freemind;
- The Brain;
- Mind Meister;
- Conception и др.

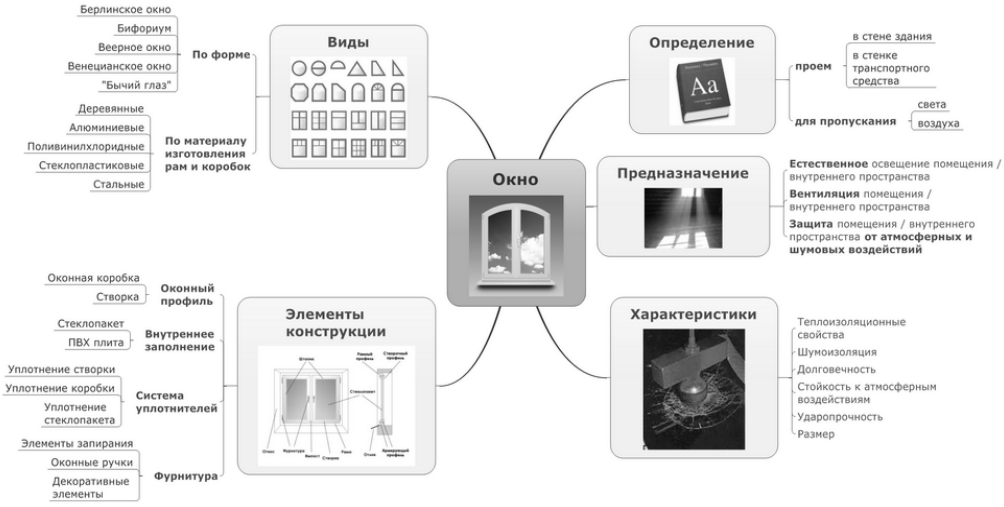


Рис. 3.9

Пример интеллект-карты

Пример и-карты, сформированной сотрудником одной из компаний (на тренинге в Москве) при помощи инструмента Mind Manager ©, представлен на рисунке 3.10.



Рис. 3.10

И-карта понятия «малое предприятие»

Хорошую и-карту нарисовать может только эксперт, у новичков обычно получаются весьма запутанные и невнятные карты. Можно сформулировать правило «хорошей и-карты».

Понятия, лежащие на одном уровне иерархии, должны быть однородны.

Однородность означает принадлежность одному уровню абстракции (так однородны «одежда–обувь» и неоднородны «мебель–стул») или детализации понятий. Так, например, все признаки или атрибуты понятия, или все его компоненты, или ассоциации. Так, на рисунке 3.11 понятия нижнего уровня неоднородны. На рисунке 3.10 также можно увидеть эту ошибку, тем не менее, приблизительную структуру знаний о малом предприятии она отражает.

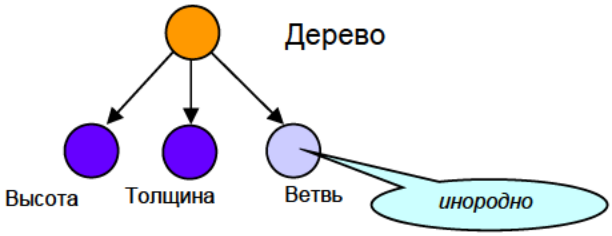


Рис. 3.11

Пример ошибки в и-карте

Концептуальные карты

Если и-карты показывают связи и древовидную структуру произвольных фрагментов знаний, то концептуальные карты (concept maps) позволяют глубже рассмотреть предметную область и включают отношения между понятиями или концептами. Такие концептуальные карты (к-карты) состоят из узлов и направленных поименованных отношений, или связей, соединяющих эти узлы. Связи могут быть различного типа, например, «является», «имеет свойство» и т. п. Концепты и связи имеют универсальный характер для некоторого класса понятий предметной области. Поэтому любая разработка к-графа подразумевает анализ структурных взаимодействий между отдельными понятиями предметной области.

В процессе создания к-карты эксперт и аналитик анализируют структуру отношений предметной области, что помогает им глубже понимать ее природу. Зачастую приходится генерировать новые, ранее невербализованные связи. Результатом этого является более осмысленное поле знаний.

Впервые к-карты были предложены Новаком в начале 70-х гг. [Novak, 1990] при изучении детского мышления и формирования первых научных понятий. Это исследование использовало идеи Дэвида Асубеля [Asubel, 1968] о формировании понятийного мышления. К-карты оказались эффективным инструментом отображения понятийной системы человека.

Визуальные спецификации в форме концептуальных карт широко применяются в обучающих системах (e-learning) и в традиционном преподавании [Kinchin, Hay, Adams, 2000], а также при моделировании корпоративных баз знаний [Загорулько, 2012].

Поскольку концептуальные карты, как и интеллект-карты, не накладывают ограничений на семантику элементов диаграммы, то они также могут использоваться не только для работы с ЧТО-знанием, но и для представления знаний других типов.

Визуальные спецификации в форме к-карт могут использоваться не только при разработке баз знаний. Они широко используются в обучающих системах (E-learning) и в традиционном обучении в классе. И студенты, и преподаватели могут применять к-карты в качестве инструментов для оценки изменений, произошедших в их мышлении. Козма [Kozma, 1992], один из разработчиков программы организации к-карт — Learning Tool, считает, что эти средства являются инструментами познания (mind tool), усиливающими и расширяющими познания человека. Разработка визуальных понятийных сетей требует от обучающихся:

- умения реорганизации знаний;
- навыков описания понятий и связей между ними;
- глубокого понимания знаний, что способствует лучшему запоминанию и извлечению из памяти знаний, а также повышает способности применять знания в новых ситуациях;
- способности связывания новых понятий с существующими понятиями и представлениями для улучшения понимания.

Полезность к-карт, пожалуй, лучше всего демонстрируется их возможностью отображать формы мышления высшего порядка. Их применяют для формальных обоснований в химии [Schlenker & Abegg, 1991] и для аргументации высказываний в биологии [Mikulecky, 1987]. Также было показано, что к-карты полезны при описании процессов проведения научных исследований [Goldsmith, Johnson & Acton, 1991].

В простейшем случае построение к-карты сводится к [Novak, 1990; Novak & Cacas, 2006]:

- определению контекста путем задания конкретного фокусирующего вопроса (focus question), определяющего главную тему и границы к-карты;
- выделению концептов — базовых понятий данной предметной области (обычно не более 15–20 понятий);
- построению связей между концептами — определению соотношений и взаимодействий базовых понятий;
- упорядочению графа — уточнению, удалению лишних связей, снятию противоречий.

«Хороший» граф обычно получается после 2–3 итераций. Типичные ошибки:

- целые предложения вместо отдельных концептов в узле;
- линейные карты;
- слишком много пересекающихся связей;
- слишком много концептов;
- неверно определенные типы отношений.

Строя к-карты в процессе создания базы знаний или экспертной системы, эксперт вместе с инженером по знаниям получают наиболее полное представление о предметной области (в то время как пользователи этой базы или системы получат «сухие выжимки», готовые факты без всего многообразия связей и путей их получения). Похожие идеи обсуждаются подробно далее в книге.

Стоит еще раз подчеркнуть, что к-карты — не только цель, но и инструмент структурирования. В процессе построения, т. е. при взаимодействии семантических связей нашей памяти с визуальной информацией, связи перестраиваются, порождая, в свою очередь, новые знания.

Можно строить к-карты на бумаге, на доске, в любом графическом или текстовом редакторе, но удобнее всего использовать специальный свободно распространяемый на сайте http://cmap.ihmc.us/ инструментарий IHMC CmapTools [Cacas et al., 2003].

Концептуальная карта представляется в виде графа, узлы которого отображают понятия (объекты или концепты), а направленные поименованные дуги, соединяющие эти узлы, — отношения (связи). Могут использоваться связи различного типа, например, «является», «имеет свойство» и т. п. Некоторые концепты и связи имеют универсальный характер для различных классов понятий и предметных областей. На рисунке 3.12 представлена концептуальная карта для системы образования города, разработанная в рамках Системного проекта г. Москвы [Костырко и др., 2012].

Диаграммы Венна

Диаграммы Венна в узком смысле — это геометрические представления множеств. Но в более широком смысле их можно рассматривать как метод визуализации и способ графической иллюстрации, широко используемых в различных областях науки. Данный тип визуализации позволяет показать все возможные теоретико-множественные или логические отношения между множествами, событиями или понятиями. Названы они по имени Джона Венна, который ввел их в 1881 г. в своей работе «Символьная логика». Пример диаграммы Венна показан на рисунке 3.13.

Концептуальные ER-диаграммы

Эти диаграммы или диаграммы сущность-связь пришли из семантического моделирования структур баз данных. Они были предложены Петером Ченом еще в 1976 г. [Чен, 1995]. ER-диаграммы используют графическое изображение сущностей предметной области, их свойств (атрибутов) и взаимосвязей между сущностями.

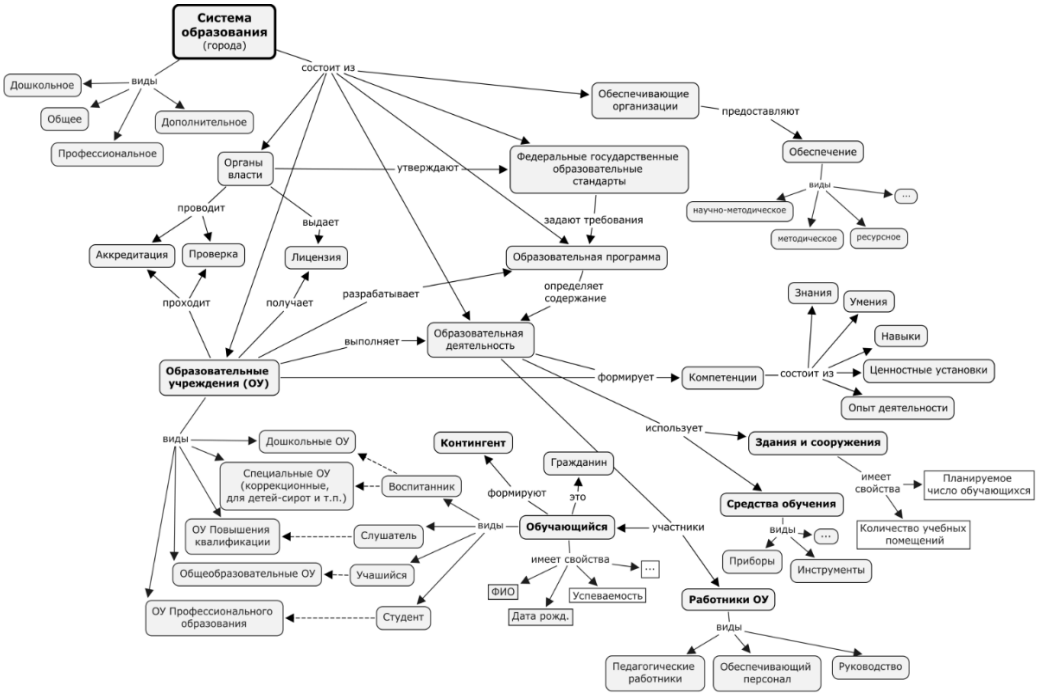


Рис. 3.12

Пример концептуальной карты



Рис. 3.13

Диаграмма Венна, иллюстрирующая представления Канта о формах государства

[Составлено по: Иммануил Кант. *Сочинения в шести томах. Том 6*. — М., 1966, 743 с.]

При построении ER-диаграмм оперируют такими понятиями, как сущность (класс объектов), экземпляр сущности (представитель этого класса), атрибут сущности (т. е. некоторое свойство), ключ сущности (набор атрибутов, значения которых в совокупности являются уникальными для каждого экземпляра сущности) и связь. Связи бывают различных типов (один-к-одному, один-ко-многим, многие-ко-многим) и иметь одну из двух модальностей (может, должен), причем связь может иметь различную модальность с разных концов.

Все это показывается на диаграмме с помощью определенных типов фигур и линий в зависимости от выбранной нотации. Пример диаграммы сущность-связь в нотации Баркера показан на рисунке 3.14.

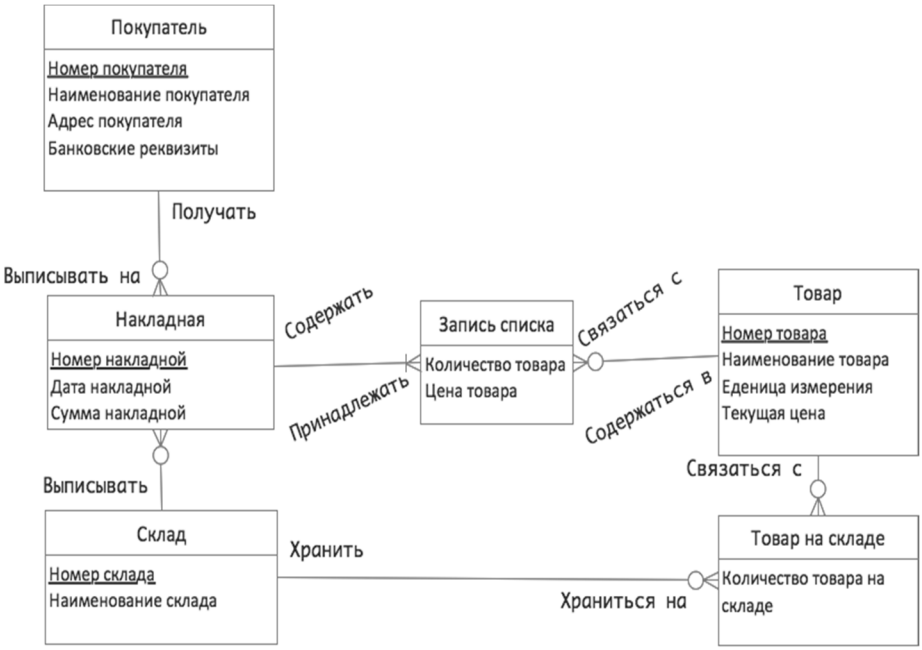


Рис. 3.14

ER-диаграмма системы заказов некоторой оптовой фирмы

ПОЧЕМУ-знания

Анализ причин, которые привели к текущей ситуации, относятся к ПОЧЕМУ-знаниям.

Диаграммы Ишикавы

Самый распространенный способ их визуализации — причинно-следственные диаграммы Ишикавы (или «рыбы») [Ishikawa, 1976]. В области контроля качества диаграмма Ишикавы — это диаграмма, которая показывает отношение между показателем качества и воздействующими на него факторами. В частности, рисунок 3.15 заимствован из стандарта по качеству Северной железной дороги — филиала ОАО «РЖД» «Методика восьми шагов решения проблем качества технических средств железнодорожного транспорта».

Кроме того, данный тип диаграммы может служить для визуального описания истории компании (продукта, технологии и т. п.), разбивая ее ход на определенные этапы и отмечая некоторые ключевые события.

Аналогичные вопросы приводят к построению **дерева текущей реальности**, разработанного в рамках теории ограничений, автором которой является Э. М. Голдратт [Новак, 2008]. Но, как следует из названия, визуальное представление несколько отличается от диаграммы Ишикавы. Цель дерева текущей реальности — выявить основную причину любого фактора, который препятствует достижению цели. При построении дерева текущей реальности обычно начинают с наблюдения нежелательных эффектов (*undesirable effects*, *UDE*). Далее в обратном порядке сопоставляются

причины и следствия, пока не обнаруживаются первопричины всех нежелательных эффектов, определенных в начале построения дерева.

Кроме того, можно выделить также и **причинно-следственную цепь** (causal chain), которая помимо ответа на вопрос «почему это произошло», позволяет описать предполагаемые последствия центрального события. Если ранее описанные модели направлены в первую очередь на анализ текущей ситуации, то причинно-следственная цепь удобна для анализа редких событий [Christianson и др., 2011]. В процессе обучения данная модель может эффективно использоваться как раз для анализа редких событий, таких как экономические и финансовые кризисы, внедрение прорывных инноваций (конкретных) и т. п.

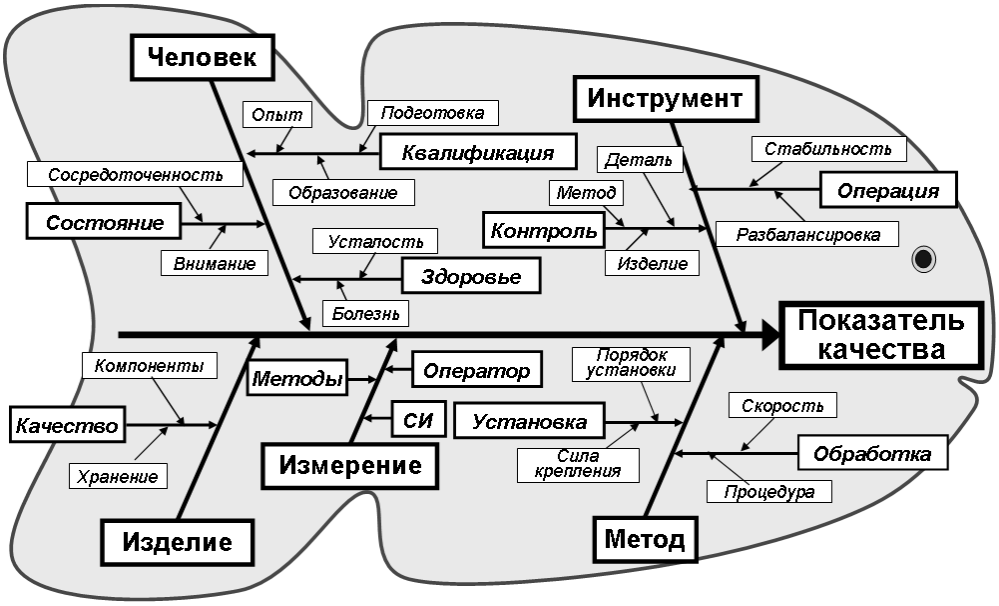


Рис. 3.15

Пример диаграммы Ишикавы «Причины хорошего или плохого качества результата»

Еще одним способом объяснения «зачем» является **карта аргументов**, заимствованная из неформальной логики. Пример простейшей карты аргументов приведен на рисунке 3.16.

Рассмотрим языки, позволяющие визуализировать ответы на вопросы «Как достичь цели?» или «Как решить проблему/задачу?»

Блок-схемы

Самым простым способом является графическое описание алгоритма решения задачи в виде блок-схемы, которое используется при обучении программированию. Это же представление можно использовать для визуализации процессов или любой последовательности действий и их взаимосвязи. Пример блок-схемы показан на рисунке 3.17.

Для моделирования бизнес-процессов разработано множество нотаций (например, IDEF, BPMN, EPC), заимствующих идею блок-схем. У каждой нотации есть свои особенности и области применения, но их обсуждение выходит за рамки данного обзора.



Рис. 3.16

Пример карты аргументов

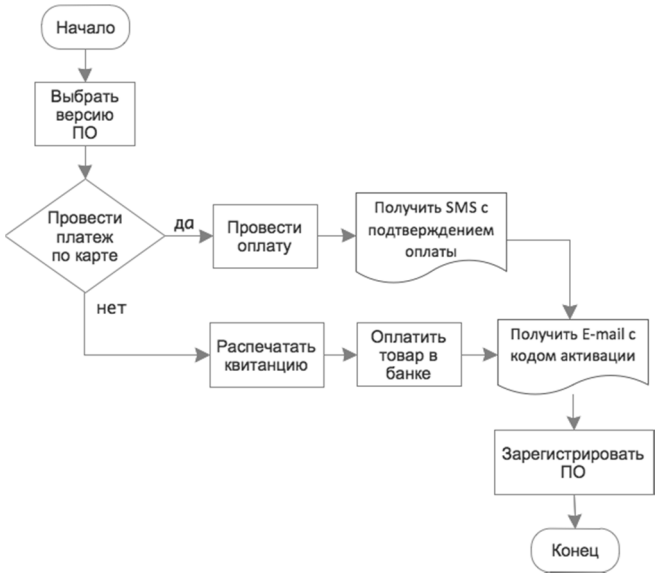


Рис. 3.17

Блок-схема процесса «Приобретение ПО через web-сайт»

КАК-знания

Иногда при моделировании процесса бывает полезно разбить состояния деятельности на группы, каждая из которых представляет собой роль, элемент организации, определенный ресурс или местоположение. Визуально каждая группа отделяется от соседних чертой (см. рис. 3.18), это похоже на плавательные дорожки в бассейне, поэтому такие диаграммы называют *swim lanes* (плавательные дорожки) [Леоненков, 2001].

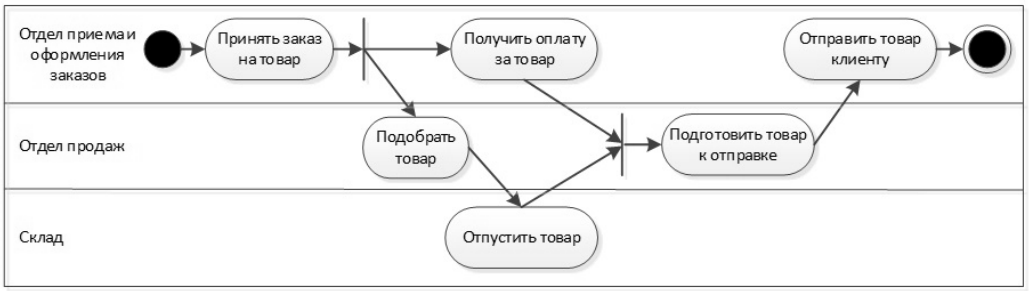


Рис. 3.18

Схема бизнес-процесса, на которой операции разнесены по отделам

Swim lane является визуальным элементом, используемых в процессе технологических схем, которые описывают, что или кто работает на определенной части процесса.

«Плавательные дорожки» расположены либо по горизонтали, либо по вертикали и используются для группировки процессов или задач в соответствии с обязанностями этих ресурсов, ролей или отделов.

Дерево целей — это структурированная, построенная по иерархическому принципу совокупность целей (например, экономической системы, программы, плана), в которой выделены генеральная цель («вершина дерева»); подчиненные ей подцели первого, второго и последующего уровней («ветви дерева»), которые позволяют достигнуть основной цели [Райзберг и др., 2007]. Концепция «дерева целей» впервые была предложена Ч. Чёрчменом и Р. Акоффом в 1957 г. [Churchmanetal, 1957]. Важность подцелей по отношению к друг другу показывают коэффициенты значимости, которые отражают их вклад в достижение конечной цели. Пример дерева целей показан на рисунке 3.19.

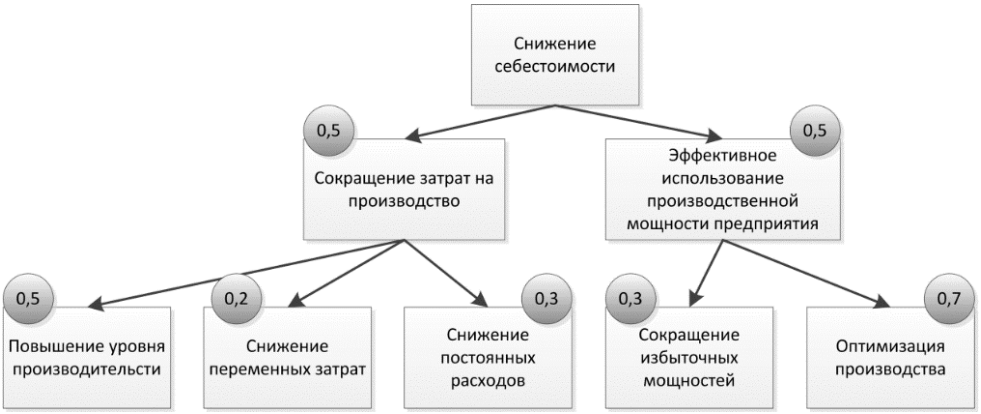


Рис. 3.19

Дерево целей: снижение себестоимости

Дерево решений рассматривает не один-единственный способ решения некоторой задачи, а помогает осуществить выбор из совокупности альтернативных способов. Сложная задача принятия решения разбивается на несколько подзадач, чтобы получить возможность решать их по очереди.

В «корне» дерева решений размещается задача, которую необходимо решить. На следующих уровнях иерархии помещаются взаимоисключающие альтернативы, среди которых происходит выбор пути решения. На последнем уровне — результат, к которому приведет выбор той или иной альтернативы, а также положительные и отрицательные стороны этого результата. Таким образом, можно сформировать понимание, как решить конкретную задачу. Пример дерева решений представлен на рисунке 3.20.

Помимо рассмотренных выше диаграмм КАК — знания часто удобно представлять в виде обычных таблиц. Следующий параграф посвящен таблицам решений.

ЗАЧЕМ-знания

В результате стратегического анализа получаются ЗАЧЕМ-знания о предметной области.

Для визуализации ЗАЧЕМ-знаний могут применяться методы, предложенные ранее для визуализации КАК-знаний. В частности, **дерево целей** (рис. 3.19), при рассмотрении его снизу вверх (от целей низших уровней, к основной цели) будет давать ответ не столько на вопрос, как достичь основной целей, сколько показывать зачем необходимо достигать «подчиненных» целей. В образовательном процессе данная модель может быть особенно полезна для описания курса, давая более четкое осознание необходимости той или иной части материала для освоения всего курса. Это полезно, как для учеников, так и для преподавателей.

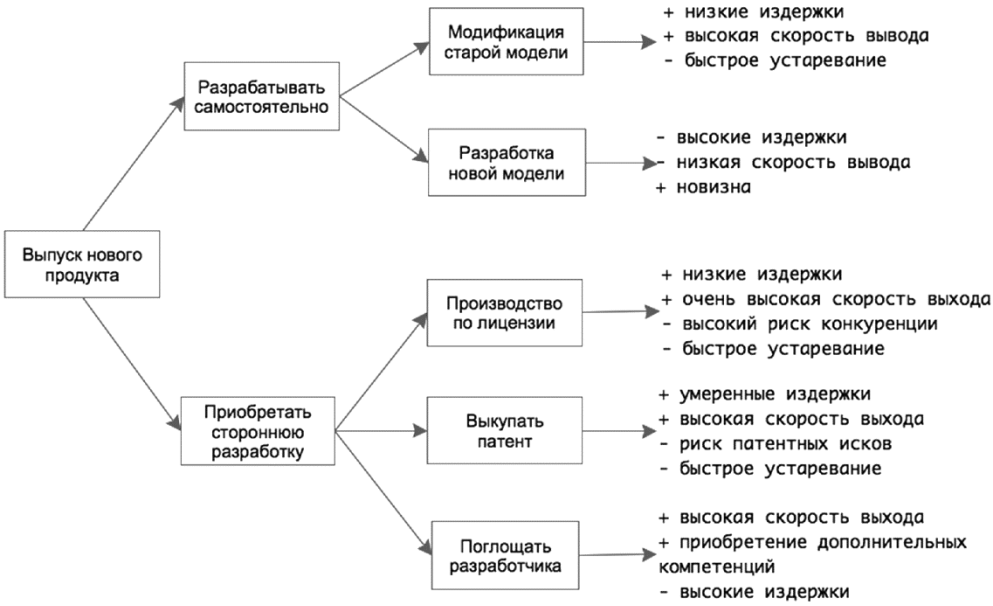


Рис. 3.20

Пример дерева решений «Выпуск нового продукта»

Дерево решений (рис. 3.20) также может помочь ответить на вопрос «зачем».

КТО-знания

Знания о том, кто за что отвечает, традиционно отображают в виде **организационной диаграммы**, представляющей собой иерархию структурных единиц

предприятия с учетом их взаимосвязей и подчинения. Но если в организационной диаграмме подразумевается, что знания о том, чем занимается каждый отдел, вытекают из его названия, то в **матрице ответственностей**, заимствованной из управления проектами, задачи прописаны явным образом.

Также на вопрос «Кто обладает знаниями в организации?» помогают ответить «плавательные дорожки» (см. рис. 3.18) и карты знаний. **Карта знаний** представляет собой графическое изображение местоположения знаний и информации, необходимых для работы организации и ее подразделений. Она связывает три типа знаний — ЧТО-, ГДЕ- и КТО- в единую модель. Карта знаний показывает возможность интеграции разъединенных знаний (скрытых и явных), находящихся в базах знаний, в головах экспертов, внутри компании и даже за ее пределами (у клиентов, партнеров, поставщиков). Это также инструмент, предоставляющий возможность обнаружения необходимых ресурсов и услуг, например, информационных, и доступа к ним [Мариничева, 2008].

Интересный способ визуализации КТО-знаний предлагает направление современной компьютерной социологии SNA (Social network analysis — анализ социальных сетей), которое занимается описанием и анализом возникающих в ходе социального взаимодействия и коммуникации связей (сетей) различной плотности и интенсивности. Цель визуализации — выявление групп, а также тех, кто является связующим звеном между группами; тех, кто собирает группу вокруг себя; и тех, кто по каким-то причинам плохо вписывается в коммуникационную сеть. На рисунке 3.21 [Whelan, 2011] можно видеть представителей всех вышеперечисленных классов — участников взаимодействия в организации. Такие сети отражают сетевой характер современного бизнеса и наглядно представляют сложность коммуникаций в организации.

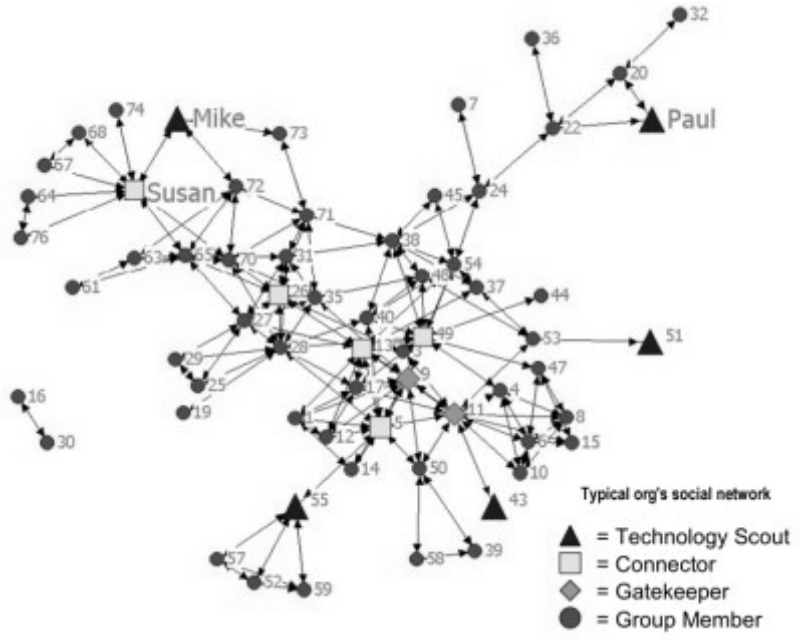


Рис. 3.21

Визуализация связей типичной организации [Whelan, 2011]

КОГДА-знания

Традиционно информация о времени проведения мероприятий узнается из различного рода календарей и расписаний. Но они не дают возможности показать взаимосвязь большого количества событий. Для этого служат сетевые графики (или PERT-диаграммы).

Сетевые графики

Сетевой график — это граф, который служит для отображения последовательности работ, их связей (одни работы не могут начаться до завершения других) и длительностей. Работы могут обозначаться как дугами (тогда вершины графа — этот состояния проекта), так и узлами (тогда дуги — это связи между работами). Но сетевой график не привязан к календарю и длительности работ никак не влияют на длины дуг. То есть сетевой график дает ответ на вопрос, в каком порядке должны осуществляться действия, но не очерчивает временных рамок их реализации.

Решить обе вышеназванные задачи можно с помощью диаграммы Ганта, объединившей в себе календарь и сетевой график. По сути **диаграмма Ганта** — это наложенный на календарь сетевой график. Задачи в диаграмме Ганта традиционно показывают прямоугольниками, причем длина этого прямоугольника соответствует длительности задачи. Стрелками обозначаются связи между задачами. Связи могут быть различных типов. Для наглядности введены специальные задачи нулевой длительности — вехи. Пример учебной диаграммы Ганта с отмеченным критическим путем показан на рисунке 3.22. В образовательном процессе диаграммы Ганта могут применяться для планирования курса преподавателем, для планирования групповых проектов студентами и для решения других схожих задач.

Кроме того, для визуализации КОГДА-знаний можно использовать диаграмму Ишикавы. На рисунке 3.23 отражена история компании Polaroid. Данный вариант использования позволяет комбинировать ПОЧЕМУ-знания и КОГДА-знания. Такой подход особенно полезен при изучении истории организаций, стран, развития отдельных событий и т. д.

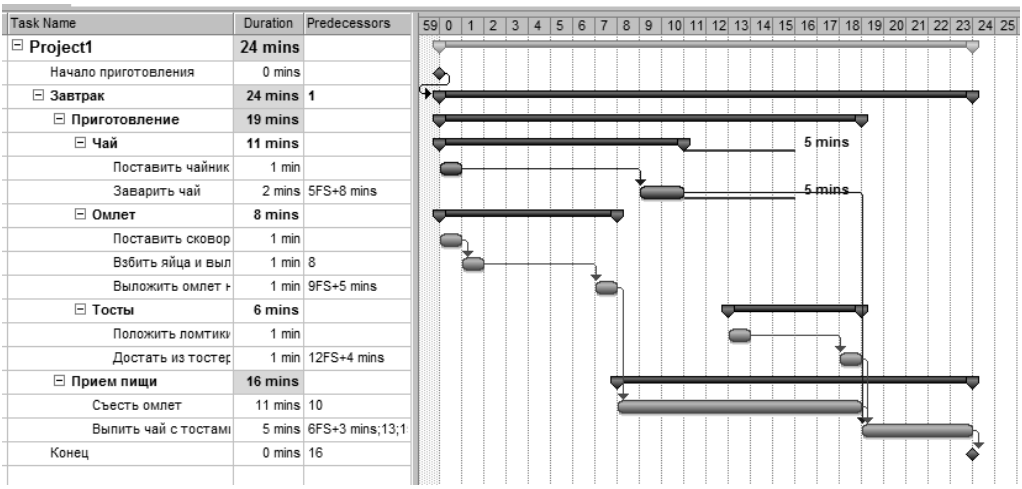


Рис. 3.22

Пример расширенной диаграммы Ганта

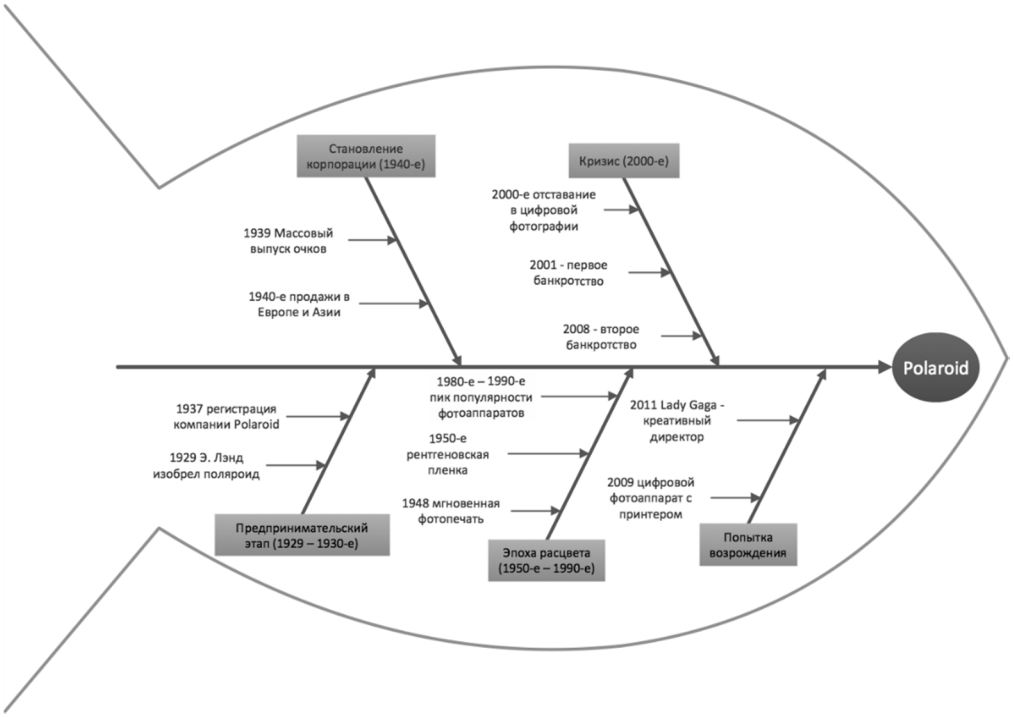


Рис. 3.23

Пример применения диаграммы Ишикавы для КОГДА-знаний

ГДЕ-знания

Первые карты появились задолго до возникновения письменности. Поэтому неудивительно, что способы визуализации информации о местоположении объектов столь широко распространены и развиты. Карты местности, планы расположения магазинов в торговом центре, примеры расстановка мебели в офисе — вот краткий перечень приложений этих способов. Визуализация ГДЕ-знаний также активно используется в гуманитарных науках: исторические карты, карты из экономической географии. Особенно актуален этот тип знаний также в современных ГИС (геоинформационных системах).

Предложеннная классификация языков визуального моделирования позволяет выбрать тип диаграммы в зависимости от типа знаний, которые необходимо представить. Также важен выбор программного инструмента для формирования визуальной модели. Сейчас имеется огромный выбор программ — от классического Visio до коммерческих ARIS и SmartDraw.

Для того чтобы упорядочить множество программных средств, можно предложить набор признаков для их классификации (рис. 3.24).

Из всего вышесказанного следует, что нет большого смысла говорить о лучшем инструменте, можно говорить о конкретном выборе согласно задаче, типу информации и квалификации аналитика. Как справедливо заметил ведущий специалист по моделированию бизнес-процессов Г. Н. Калянов, «при выборе лучшим является тот язык моделирования, на котором ты хорошо умеешь работать».

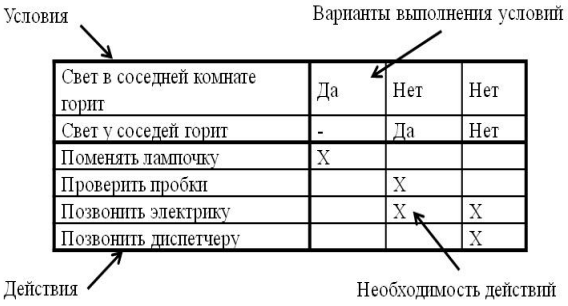


Рис. 3.24

Характеристики инструментов визуального структурирования

3.4. ТАБЛИЧНЫЕ МЕТОДЫ СТРУКТУРИРОВАНИЯ ЗНАНИЙ

Таблица решений (таблица принятия решений) — способ компактного представления КАК-знаний и подходит для отображения моделей со сложной логикой. Аналогично условным операторам в языках программирования, она устанавливают связь между условиями и действиями. Но, в отличие от традиционных языков программирования, таблицы решений в простой форме могут представлять связь между множеством независимых условий и действий. Таблицы принятия решений, как правило, разделяются на четыре квадранта: *Условия* — список возможных условий, *Варианты выполнения условий* — комбинация из выполнения и/или невыполнения условий из этого списка, *Действия* — список возможных действий, *Необходимость действий* — указание надо или не надо выполнять соответствующее действие для каждой из комбинаций условий (рис. 3.25).



Свет в соседней комнате горит	Да	Нет	Нет
Свет у соседей горит	-	Да	Нет
Поменять лампочку	X		
Проверить пробки		X	
Позвонить электрику		X	X
Позвонить диспетчеру			X

Рис. 3.25

Пример таблицы решений

Например, для ситуации «свет погас», но «свет в соседней комнате горит» необходимо «поменять лампочку». Вариантов выполнения условия может быть не два: да или нет, а несколько, например, цвет может быть красным, оранжевым, синим. В более сложных таблицах может применяться нечеткая логика. Действия могут быть элементарными или ссылаться на другие *таблицы принятия*

решений. Необходимость выполнения действий может быть неупорядоченной, как в данном примере, или упорядоченной. В последнем случае если при определенной комбинации выполнения условий возможно выполнение нескольких действий, то в таблице решений указывается их приоритет.

Другим примером табличного метода структурирования знаний является QFD (акроним от англ. Quality Function Deployment), или структурирование (развертывание) функции качества. Данный метод помогает трансферу знаний и принятию решений при разработке продуктов или услуг. По мнению создателей, QFD может помочь организации сосредоточить внимание на важнейших характеристиках новых или существующих продуктов или услуг с точки зрения отдельного клиента, сегмента рынка, компании, или технологии развития. Результатами применения методики являются понятные схемы и матрицы, которые могут быть повторно использованы для будущих товаров либо услуг. QFD трансформирует потребности клиентов (голос клиента) в инженерные характеристики продукции, расставляет приоритеты для каждого продукта / услуги и одновременно определяет задачи в области развития продукции или услуги.

Основа QFD — построение фигурной матрицы, названной в соответствии со своей формой «Дом качества» (рис. 3.26), в рамках которой фиксируется информация о качестве продукта и принимаемых решениях.

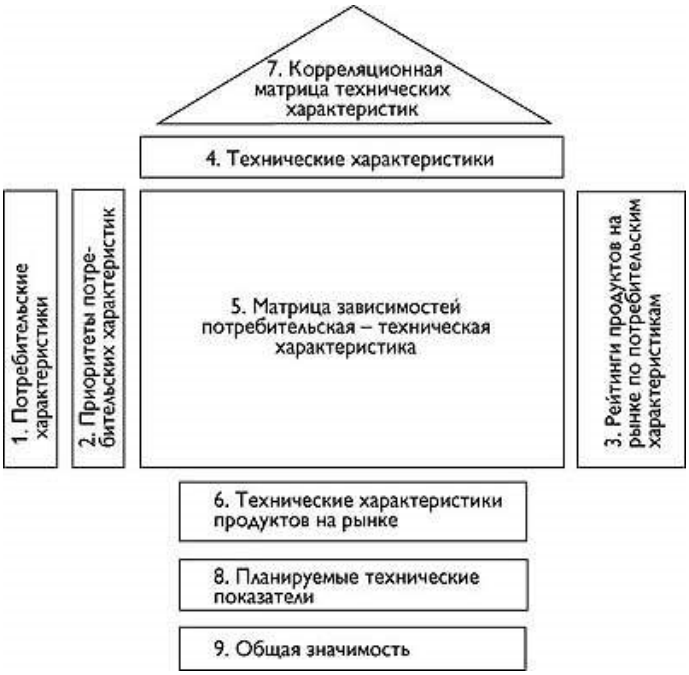


Рис. 3.26

Схема дома качества

Центральная часть дома — это таблица, столбцы которой соответствуют техническим характеристикам, а строки — потребительским. В клетках отмечается уровень зависимости, если она есть. Крышу дома представляют сведения о корреляции между техническими характеристиками. Левое крыло — столбец приоритетов пользовательских характеристик. Правое крыло — таблица рейтингов потребительских

характеристик (с точки зрения пользовательского восприятия) для существующих на рынке подобных продуктов. Подвал дома содержит результаты анализа технических характеристик конкурирующих продуктов, результаты выработки стратегии изменения технических характеристик своего продукта (планируемые показатели для первоначальной разработки), оценки абсолютной и относительной важности.

Предпосылками QFD являются маркетинговые исследования, определяющие, что хочет пользователь, насколько важны те или иные качества (левое крыло, шаги 1 и 2), а также, как решают подобные проблемы другие поставщики (правое крыло, шаг 3). Каждому продукту, включая свой текущий, конкурентов, свой перспективный по каждому требованию при сваивается рейтинг. После определения набора технических характеристик (шаг 4), заполняется центральная часть дома — определяются зависимости между потребительскими и техническими характеристиками (шаг 5). На шестом шаге анализируется уровень реализации в конкурирующих продуктах. После анализа взаимной корреляции технических характеристик (шаг 6), исходя из полученных сведений, формируются целевые показатели для разрабатываемого продукта (шаг 7). Наконец для сопоставления значимости технических характеристик (шаг 8) формируется общая значимость, как сумма, в которой каждой непустой клетке в столбце матрицы зависимостей сопоставляется произведение приоритета соответствующего пользовательского показателя на уровень зависимости (рис. 3.27).

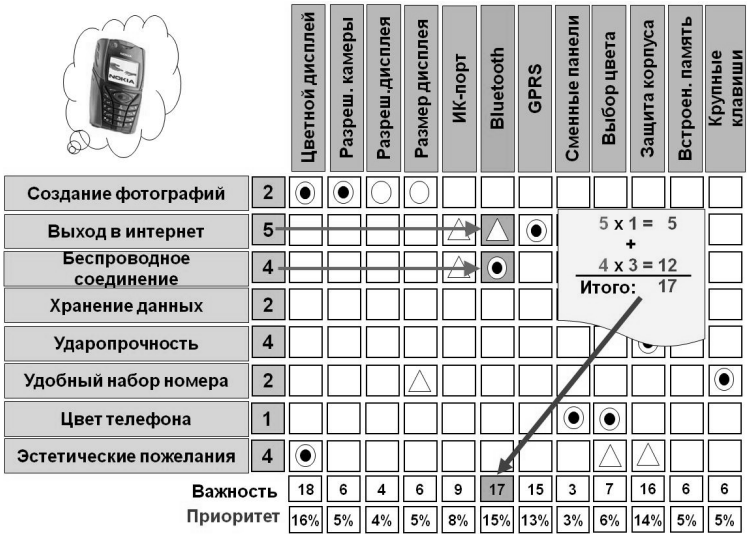


Рис. 3.27

Определение важности технических характеристик мобильного телефона с помощью QFD

QFD позволяет в компактной форме представить данные о разнообразных характеристиках продукта, а также отследить их влияние на принимаемые технические решения. В развернутом виде QFD включает четыре фазы, и на каждой из них строится свой дом качества. После преобразования потребительских характеристик в технические, последние преобразуются в характеристики компонентов, и далее: в операции процессов, а затем в параметры технологических процессов производства (рис. 3.28).

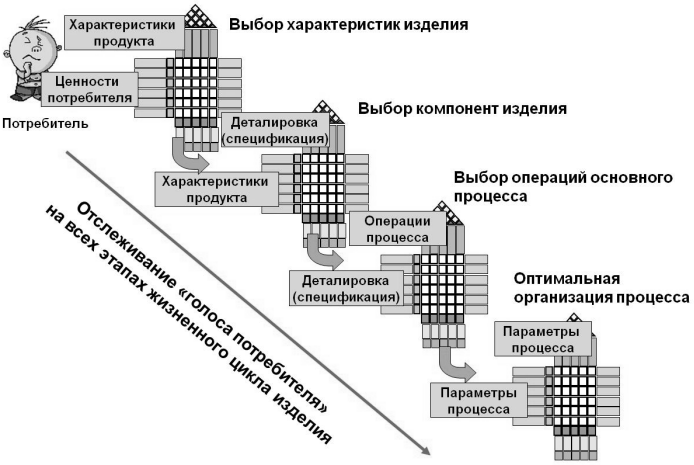


Рис. 3.28

Последовательное развертывание функции качества

3.5. СТРАТЕГИИ СТРУКТУРИРОВАНИЯ

После представления множества возможных форм для формирования поля знаний, в качестве простейшего прагматического подхода к его формированию, начинающему инженеру по знаниям можно предложить следующий алгоритм для «чайников» (рис. 3.29).

1. Определение входных $\{X\}$ и выходных $\{Y\}$ данных, если они имеются. Этот шаг совершенно необходим в случае разработки систем рекомендаций или советующих систем, так как он определяет направление движения в поле знаний — от X к Y. Кроме того, структура входных и выходных данных существенно влияет на форму и содержание поля знаний. На этом шаге определение может быть достаточно размытым, в дальнейшем оно будет уточняться.

2. Составление словаря терминов и наборов ключевых слов N. На этом шаге проводится текстуальный анализ всех протоколов сеансов извлечения знаний, затем выписываются все значимые слова, обозначающие понятия, явления, процессы, предметы, действия, признаки и т. п. При этом следует попытаться разобраться в значении терминов. Важен осмысленный словарь.

3. Выявление объектов и понятий $\{A\}$. Производится «просеивание» словаря N и выбор значимых для принятия решения понятий и их признаков. В идеале на этом шаге образуется полный систематический набор терминов из какой-либо области знаний.

4. Выявление связей или отношений между понятиями. Но определить, как направлены связи, что ближе, а что дальше, необходимо на этом этапе. Таким образом, связи в сети ассоциаций только намечены, но пока не поименованы. Например, понятия «день», «ночь», «утро» и «вечер» явно связаны, связаны также и понятия «красный флаг» и «красный галстук», но характер связей тут различен.

5. Выявление метапонятий и детализации понятий. Связи, полученные на предыдущем шаге, позволяют инженеру по знаниям структурировать понятия, а также выявлять понятия более высокого уровня обобщения (метапонятия), детализировать на более низком уровне.

6. Построение «картинки» или пирамиды знаний. Под пирамидой знаний мы понимаем иерархическую лестницу понятий, подъем по которой означает углубление понимания и повышения уровня абстракции (обобщенности) понятий. Количество уровней в пирамиде зависит от особенностей предметной области, профессионализма экспертов и инженеров по знаниям.

7. Выбор диаграммы, наиболее подходящей к отображению полученных знаний.

8. Определение отношений $\{RA\}$. Отношения между понятиями выявляются как внутри каждого из уровней пирамиды, так и между уровнями. Фактически на этом шаге даются имена тем связям, которые обнаруживаются на шагах 4 и 5, а также обозначаются причинно-следственные, лингвистические, временные и другие виды отношений.

9. Определение стратегий принятия решений (Sf). Определение стратегий принятия решения, т. е. выявление цепочек рассуждений, связывает все сформированные ранее понятия и отношения в динамическую систему поля знаний. Именно стратегии придают активность знаниям, именно они «перетряхивают» модель M в поиске от X к Y.



Рис. 3.29

Стадии структурирования знаний — алгоритм для «чайников»

Однако на практике при использовании данного алгоритма можно столкнуться с непредвиденными трудностями, связанными с ошибками на стадии извлечения знаний и с особенностями знаний различных предметных областей. Тогда возможно привлечение других, более «прицельных» методик и приемов структурирования. При этом на разных этапах структурирования знаний возможно использование различных методик.

Дуальная стратегия структурирования

Традиционно при проектировании сложных систем и при структурировании больших объемов информации использовался иерархический подход [Месаро-

вич, Такахара, 1972] как методологический прием расчленения формально описанной системы на уровни (или блоки, или модули). На высших уровнях иерархии используются наименее детализованные представления, отражающие только самые общие черты и особенности проектируемой системы. На следующих уровнях степень подробности возрастает, при этом система рассматривается не в целом, а отдельными блоками.

Одно из преимуществ иерархического подхода состоит в том, что сложная задача большой размерности разбивается на последовательно решаемые группы задач малой размерности. На каждом уровне вводятся свои представления о системе и элементах. Элемент k-го уровня является системой для $k-1$ уровня. Продвижение от уровня к уровню имеет строгую направленность, определяемую стратегией проектирования — сверху-вниз или снизу-вверх.

Описанный выше объектно-структурный подход позволяет объединить две обычно противопоставляемые стратегии проектирования — нисходящую или дедуктивную STRtd (top-down) с последовательной декомпозицией объектов и процессов сверху вниз и восходящую или индуктивную STRbu (bottom-up) с постепенным обобщением понятий и увеличением степени абстрактности описаний снизу-вверх.

Синтез этих стратегий, а также включение возможности итеративных возвратов на предыдущие уровни обобщений позволили создать дуальную концепцию, предоставляющую аналитику широкую палитру возможностей на стадии структурирования знаний как для формирования концептуальной структуры предметной области *Sc*, так и для функциональной структуры *Sf*.

Рисунок 3.30 иллюстрирует дуальную концепцию при проектировании *Sc* для ЭС помощи оператору энергетического блока.

Основанием для прекращения агрегирования и дезагрегирования является полное использование словаря терминов, которым пользуется эксперт, при этом число уровней является значимым фактором успешности структурирования.

Неконтролируемый и неуправляемый рост информации и стремительные техногенные изменения вызвали появление новых направлений в науках по изучению процессов восприятия, обучения и обработки информации, связанных с визуализацией (инфографика, когнитивная графика, виртуальная реальность, компьютерная графика и т. д.) [Рапуто, 2011]. Основная цель этих новых видов графической подачи информации — сделать сложные и громоздкие информационные массивы более простыми и понятными для восприятия. Фактически, структурирование знаний также направлено на решение задачи по пониманию и компрессии информации.

Все больше современных специалистов понимают важность проблемы структурирования и наглядного представления знаний в различных сферах деятельности. Визуализация позволяет наглядно показать изучаемые процессы, явления и понятия в тех случаях, когда непосредственно восприятие затруднено. Графические представления как метод компактной организации информации могут использоваться как мощное орудие мышления применительно ко всем областям профессиональной интеллектуальной деятельности, включая моделирование предприятий, сложных организационных структур, рынка, а также процессам бизнес-обучения.

Необходимо признать, что структурирование знаний с использование разнородных визуальных языков моделирования имеет свои сложности, ограничения и область применения. Одним из проблемных вопросов в использовании графических языков является семантическая интероперабельность — различные пользователи могут по разному интерпретировать значения символов на диаграмме,

различные языки моделирования могут иметь элементы с похожими названиями, но с разным смыслом, а также, наоборот, при схожести смысла понятий могут различаться по формулировкам. Для устранения данной проблемы используются унифицированные метамодели [Heidarietal, 2013] или онтологии [Grüningeretal, 2000; Anderssonetal, 2006]. На практике в данном случае еще полезны непротиворечивые демонстрационные примеры диаграмм, выполненных в соответствии с требуемой нотацией. Чем их больше, тем лучше. Пользователи понимают значения элементов визуального языка через примеры (экстенсиональные определения понятий).

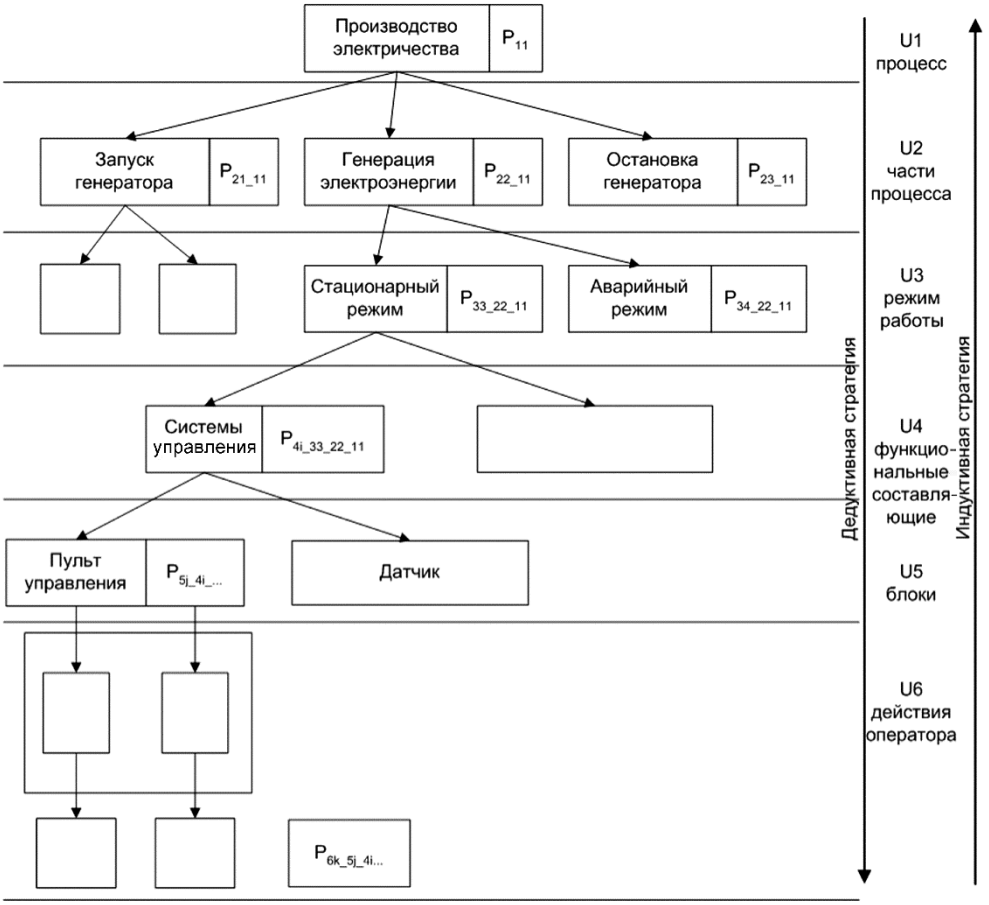


Рис. 3.30

Дуальная стратегия проектирования

Кроме того, для решения управленческих задач обычно нужна не отдельная диаграмма, а комплекс диаграмм, охватывающих разные типы знаний. Например, при оптимизации деятельности с помощью теории ограничений используются 5 взаимосвязанных типов диаграмм [Детмер, 2007], а при проектировании ИС — UML [Буч и др., 2006] или Archimate [Iacobetal, 2012].

Наконец, не всегда диаграммы — это оптимальный способ представления знаний. В статьях [Ghoniem et al., 2005; Keller, 2006] было произведено сравнение матричных (табличных) представлений и наиболее популярных диаграмм вида «узел–связь». Гоним с со-авторами [Ghoniem et al., 2005] показал, что матрицы превосходят диаграммы в случае больших сильносвязанных графов в контексте нескольких задач чтения, за исключением задачи «нахождения пути» (pathfinding). Такие выводы поддерживаются экспериментами Келлера и его коллег [Keller, 2006], в результате которых было выявлено, что диаграммы «узел-связь» нагляднее для небольших и несложных структур, однако они становятся сложными при визуализации моделей больших систем.

Несмотря на вышесказанное, диаграммы являются полезным средством для представления знаний, однако нуждаются в поддерживающих механизмах. В частности, методы визуального моделирования необходимо сочетать с онтологическим инжинирингом и метамоделированием [Кудрявцев, 2006] для обеспечения семантической интероперабельности. Иначе их можно комбинировать с инженерией методов (method engineering) [Henderson-Sellers, Ralyté, 2009] и отраслевыми лучшими практиками для интеграции диаграмм со способами решения задач в различных областях, а также с неграфическими методами представления информации в ситуациях, когда последние более эффективны [Grigoriev, Kudryavtsev, 2013; Григорьев, Кудрявцев, 2012].

Вся инфографика нацелена на взаимосвязь между единицей информации (не имеет значения какой), количественным значением и общим положением данной информации в системе. По сути, основная задача визуализации — это графическая интерпретация информации, данных и знаний, позволяющая «сжимать» информацию, освобождая ее от словесного «балласта».

3.6. О ЯЗЫКАХ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

Поле знаний — это только эскиз будущей базы знаний, но не формальное ее описание. Описанные выше диаграммы (например, и-карты и к-карты) являются инструментом для подготовки поля знаний. Теперь необходимо перейти к этапу формализации (см. главу 4) и создать модель представления знаний.

Модель представления знаний позволит описать объекты со свойствами и закономерности предметной области. Или, другими словами, эта модель должна отражать взаимосвязи между имеющимися данными. Существуют десятки моделей или языков представления знаний (ЯПЗ) как универсальных, так и специализированных для различных предметных областей. Многие ЯПЗ появились уже в начале 1970-х гг. при разработке первых интеллектуальных систем. Тогда же появились первые языки программирования, ориентированные на представление знаний. Наиболее известные и распространенные из них — языки Prolog и Lisp — с успехом применяются до сих пор. В 1980-х гг. получили развитие многие формальные компьютерные языки и системы представления и обработки обобщенных знаний, записанных на естественном языке, например, проект Cyc (сокр. от Encyclopedia). Наконец, с развитием электронного документооборота, появились языки, явно содержащие структуру документа. К ним можно отнести HTML, SGML, XML и пр. Однако при всем разнообразии ЯПЗ большинство из них основано на использовании следующих моделей:

- таблицы решений и операторов;
- семантические сети и концептуальные графы;
- фреймы и сценарии;

- продукционные правила;
- объектно-ориентированные и объектно-структурированные языки;
- логические модели.

При разработке новой интеллектуальной системы может быть выбран один из ЯПЗ или использоваться сочетание нескольких моделей. Во многом от выбранного ЯПЗ зависит успех экспертной системы и эффективность ее работы. Каждый из языков представления знаний обладает своими преимуществами при решении одних задач и малоэффективен для других. Наиболее важными критериями, пожалуй, можно назвать следующие:

- качество решения, предоставляемого экспертной системой (т. е. процент ошибок, скорость получения решения и пр.);
- простоту понимания и модификации базы знаний.

При выборе метода представления знаний можно руководствоваться общепринятыми для данной предметной области подходами. Так, например, в задачах проектирования распространены табличные описания, в медицине, как правило, постановка диагноза и лечение болезней описываются в свободной форме, с использованием неформальных правил. Решение игровых задач, доказательство теорем могут быть описаны с помощью языка логики.

Но ошибочно считать, что представление знаний — это простое кодирование информации.

С учетом сделанных замечаний можно дать следующее определение термину *представление знаний*:

> Представление знаний — это описание понятий и действий, необходимых для решения проблемы, на формальном языке, пригодном для построения компьютерной системы.

Однако не следует смешивать понятие модели представления знаний и среду ее выполнения. В качестве среды реализации механизмов представления знаний может служить какой-либо язык программирования. Например, специализированные языки Prolog, Lisp или универсальные Scala, C#, Java или Python. Средой также могут быть средства представления в таких инструментариях, как Exsys или G2.

После выбора языка представления знаний и программного средства следует процесс формализации или переноса поля знаний на этот язык. Существуют два основных подхода к формализации. Первый (модульный) рассматривает и моделирует независимые фрагменты (элементы) знаний, второй (сетевой) делает акцент на взаимосвязанных конструкциях. Рисунок 3.31 иллюстрирует это деление. Если вспомнить классификацию знаний из рисунке 3.2, то следует заметить, что модульный подход ориентирован на моделирование поверхностных знания, а сетевой — глубинных. На сегодняшний день большинство современных ЭС аккумулируют поверхностные знания.

Шире и подробнее модели представления знаний будут описаны далее. При этом будет расширена классификация на рисунке 3.31. Там же рассматриваются языки представления знаний, применявшиеся в первых экспертных системах: таблицы операторов, семантические сети и фреймы. Все эти модели позволяют структурировать данные и знания для эффективного решения интеллектуальных задач. Онтологии как иерархические семантические сети будут обсуждаться во второй части книги.



Рис. 3.31
Языки представления знаний

3.7. МЕТАДАННЫЕ

В век «больших данных» (Big Data) специалистам бывает довольно трудно разобраться в имеющихся разнообразных и многочисленных информационных ресурсах. Для представления сведений о ресурсах (в частности, о документах и базах данных) было предложено понятие метаданных, описывающих содержимое ресурса в виде специального краткого набора данных, описывающих также и связи с другими ресурсами. Метаданные используются для автоматизированного анализа содержимого ресурса, построения поисковых индексов и позволяют обеспечить достаточно высокую точность и эффективность поиска разнородной информации.

Данный термин принято толковать как «данные о данных», однако его значение распространяется помимо описания состава данных, их структуры (формата) представления, места хранения и других признаков описания, также на поддерживающие их информационные системы, технологии, пользователей, методы доступа и т. д. Особенно широко этот термин стал использоваться в последние годы в связи с развитием электронных библиотек, поскольку метаданные стали важнейшим средством обеспечения навигации и поиска информации.

Существуют различные категории метаданных, например, описательные метаданные (descriptive metadata) (в том числе библиографические данные); метаданные о структурах и форматах (structural metadata); административные метаданные (administrative metadata), содержащие данные для управления доступом; идентификационные метаданные (identifier metadata), которые однозначно идентифицируют объекты внешнего мира и т. п.

Различие между данными и метаданными часто условно и зависит от контекста. Например, реферат с точки зрения терминологии электронных библиотек относится к метаданным, хотя в электронном каталоге или реферативной базе данных его содержание рассматривается как данные. Существуют различные концепции, стандарты и системы представления метаданных. Ниже приводятся несколько самых известных из них.

Dublin Core (DC) — «Дублинское ядро»: разработка ведется с 1995 г. рабочей группой с одноименным названием. Ею предложена простая структура описания документов, которая, по мнению разработчиков, должна заменить сложные системы существующей каталогизации документов. Текущая версия (DC 1.1) спецификации «Дублинского ядра» была принята в июле 1999 г.

В соответствии с нею структура описания данных включает 15 элементов.

1. Заголовок (Title) — название, присвоенное ресурсу его создателем.

2. Создатель (Creator) — лицо, организация или служба, ответственная за подготовку ресурса (автор, исполнитель, фотограф).

3. Предмет (Subject) — тема ресурса, выраженная ключевыми словами, элементами тематического справочника или короткой фразой.

4. Описание (Description) — текстовое описание ресурса.

5. Издатель (Publisher) — лицо, организация или служба, обеспечивающая доступ к ресурсу.

6. Участник создания материала (Contributor) — человек или организация, которые не являются авторами, однако внесли в создание ресурса значительный интеллектуальный вклад помимо указанного в разделе «Создатель».

7. Дата (Date) — дата, указывающая на создание или появление ресурса в доступном для использования виде.

8. Тип (Type) — жанр, категория ресурса, например, домашняя страница, роман, статья и т. п.

9. Формат (Format) — способ представления ресурса, например, тип программного обеспечения и персонального компьютера, необходимых для отображения ресурса.

10. Идентификатор (Identifier) — например URL.

11. Источник (Source) — сведения о первичном источнике, из которого был продуцирован данный ресурс.

12. Язык (Language) — язык представления ресурса.

13. Связь (Relation) — ссылка на ресурс, связанный с данным; идентификатор вторичного ресурса и его связь с настоящим ресурсом, например, издание книги и глава книги.

14. Охват (Coverage) — области времени, пространства и т. п., к которым относится содержание ресурса.

15. Авторские права (Rights) — права интеллектуальной собственности на ресурс.

Как указывается в RFC2413 (см. RFC2413, Dublin Core Metadata for Resource Discovery), эти элементы можно условно разбить на три группы (табл. 3.2).

Таблица 3.2

Три группы элементов метаданных «Дублинского ядра»

Содержание	Интеллектуальная собственность	Экземпляр
Название	Создатель	Дата
Тема	Издатель	Формат
Описание	Участник создания материала	Идентификатор
Тип	Авторские права	Язык
Источник		
Связь		
Охват		

1. Содержание (Content) — элементы, в основном относящиеся к содержанию ресурса.

2. Интеллектуальная собственность (Intellectual Property) — элементы, в основном рассматриваемые с позиции интеллектуальной собственности.

3. Экземпляр (Instantiation) — элементы, в основном относящиеся к данному экземпляру ресурса.

В настоящее время ведется разработка версии DC 2.0. Рабочая группа Dublin Core работает в контакте с разработчиками языка RDF.

LOM — модель учебного объекта — система метаданных для образовательных ресурсов, разработанная Международной ассоциацией IEEE, принятая в качестве стандарта в России для Единой информационно-образовательной среды.

GEM — версия «Дублинского ядра» метаданных, доработанная для образовательных информационных ресурсов, принятая в качестве стандарта для образовательных ресурсов США.

VCard — система метаданных для представления сведений о физических лицах и организациях, разработанная Международной ассоциацией IEEE.

MARC — машиночитаемый каталог — семейство форматов, широко распространенных для представления библиографической информации, главным образом в мировом библиотечном сообществе. В полном варианте включает до 800 элементов данных. Является стандартом де-факто для российских библиотечных сетей.

ONIX — формат описания информационных товаров в системах электронной торговли (книги, периодика, компактные диски, аудио- и видеокассеты и др.). Основан на системе стандартов ЭДИФАКТ. GILS — Глобальная (правительственная) служба поиска информации. Система метаданных.

GILS представляет собой расширение форматов MARC для описания информационных ресурсов широкого профиля.

EDIFACT — одна из старейших международных систем метаданных, содержащих правила и структуру описания торгово-транспортных и других коммерческих документов. С 1998 г. поддерживается языком XML. Реализация применительно к торговле книгами и другими информационными товарами — система ONIX.

Формат Государственного регистра баз и банков данных. Российский стандарт, действующий с конца 1980-х гг. и содержащий систему метаданных для баз данных и других электронных наборов данных. Утвержден Правительством РФ, поддерживается НТЦ «Информрегистр», применяется в ряде отраслей и регионов, а также в странах СНГ для ведения баз метаданных.

Имеется еще несколько десятков менее известных систем метаданных, в той или иной степени претендующих на широкое применение. Среди большого количества разнообразных систем метаданных, используемых в информационных ресурсах в сфере науки, культуры и образования и рассмотренных выше, наиболее распространенными и признанными являются системы метаданных семейства MARC, DC, а также быстро распространяющаяся в информационно-образовательной среде система метаданных LOM. Поэтому при создании сводных каталогов электронных ресурсов разработчикам чаще всего приходится иметь дело с интеграцией описаний ресурсов, представленных в этих системах метаданных.

3.8. ВИДЫ ИНФОРМАЦИОННО-ПОИСКОВЫХ ЯЗЫКОВ

Отдельно следует выделить структурирование и поиск библиотечной информации. Для этого существуют специализированные информационно-поисковые языки (ИПЯ). Тема ИПЯ имеет многовековую историю в рамках библиотековедения и научно-технической информации. Основные положения в данной области зафиксированы в виде системы стандартов по информации, библиотечному и издательскому делу (СИБИД), которая объединяет в себе общетехнические и организационно-методические стандарты, регламентирующие правила представления данных, описания документов, функционирования библиотечных фондов, оформления печатных и электронных изданий и многое другое (http://www.gsnti-norms.ru/norms/norms/stands2.htm).

Большинство представленных в данном параграфе понятий и определений основано на ГОСТ 7.74-96 «СИБИД. Информационно-поисковые языки. Термины и определения» (http://www.gsnti-norms.ru/norms/common/doc.asp?0&/norms/stands/7_74.htm). И сам параграф носит справочный характер.

Информационно-поисковый язык, ИПЯ (information retrieval language, indexing language) — искусственный язык, предназначенный для формализованного описания смыслового содержания документов, данных, отдельных понятий или терминов и обеспечения последующего их поиска в информационно-поисковых массивах. Формализация лексики и создание различных ИПЯ вызвано необходимостью устранения «избыточности» и «недостаточности» естественного языка для целей информационного поиска, а также ликвидации присутствующих в нем синонимии и омонимии для реализации однозначности информационного поиска.

Поисковый образ документа (document description, search document image) — описание содержания документа на *ИПЯ*, отражающее важные признаки его содержания и вида для реализации поиска данного документа.

Индексирование: выражение содержания документа и/или смысла информационного запроса на ИПЯ. В исследованиях по семантическим технологиям данному термину соответствует *аннотирование (annotating)*.

ИПЯ делятся на классификационные, предметизационные и дескрипторные [Сукиасян, 2004, 2006, 2009].

Классификационный ИПЯ (Classificational information retrieval language) — ИПЯ, предназначенный для индексирования документов (частей документов) и информационных запросов посредством понятий и кодов какой-либо классификационной системы.

Предметизационный ИПЯ (язык предметных рубрик, Subject headings language) — ИПЯ, предназначенный для индексирования документов (частей документов) и информационных запросов посредством предметных рубрик, представляющих собой краткую формулировку темы на естественном языке.

Дескрипторный ИПЯ (Descriptor language) — ИПЯ, предназначенный для координатного индексирования документов и информационных запросов посредством дескрипторов и/или ключевых слов.

В соответствии с видами ИПЯ различаются процессы индексирования.

Классификационное индексирование, библиографическая систематизация, классификационная систематизация или просто систематизация: присвоение данным или документам классификационных индексов в соответствии с правилами какого-либо классификационного ИПЯ. Классификационный индекс: поисковый образ, построенный средствами классификационного ИПЯ.

Предметное индексирование, предметизация: индексирование предметного содержания документов.

Координатное индексирование: индексирование, предусматривающее многоаспектное выражение основного смыслового содержания документа или смыслового содержания информационного запроса множеством ключевых слов или дескрипторов.

Классификационные ИПЯ[1]

Описание и применение классификационных ИПЯ строится на основе следующих понятий.

Классификационная система (КС) — средство формализованного представления содержания документов, данных и информационных запросов посред-

[1] Дальнейший обзор ИПЯ заимствован из [Гиляревский и др., 2008].

ством кодов или описаний классов логически упорядоченного множества понятий, где: *код класса* — это обозначение класса средствами нотации (индексации) классификационной системы; *описание класса, наименование классификационного деления* — это обозначение класса на естественном языке. *Классификационная запись, классификационное деление:* совокупность элементов, обозначающая в классификационной таблице класс классификационной системы и состоящая из кода класса, описания класса и методических указаний.

Классификационный индекс — это поисковый образ, построенный средствами классификационного ИПЯ. Классификационные индексы бывают простыми, комбинированными и сложными. *Простой индекс* — классификационный индекс, содержащий один код класса без сочетания с кодами других классов. *Комбинированный индекс* — классификационный индекс, образованный из двух или более кодов класса основной таблицы классификации. *Сложный индекс* — классификационный индекс, образованный из сочетания кода класса основной таблицы классификации с кодами классов вспомогательных таблиц классификации.

По содержанию различаются КС универсальные, охватывающие все области знаний, и отраслевые для отдельных предметных областей.

По структуре КС можно разделить на перечислительные, комбинированные и фасетные. До конца XIX — начала XX вв. создавались КС перечислительного типа, деления которых располагались в простой последовательности единственного иерархического ряда.

Наиболее известны среди них «Десятичная классификация» М. Дьюи (1876), «Классификация Библиотеки Конгресса США» (1904). С появлением метода типизации, в соответствии с которым вне основной классификационной таблицы создавались стандартизированные типовые деления, отражающие форму издания, географические, хронологические и др. признаки, на смену перечислительным системам пришли системы комбинационного типа, позволяющие включать в индекс несколько классификационных характеристик.

В России популярность завоевала Универсальная десятичная классификация (УДК). Дальнейшее развитие теории и практики КС привело к возникновению аналитико-синтетических, или фасетных, КС. Теория их построения разработана индийским ученым и библиотековедом Ш. Р. Ранганатаном («Классификация двоеточием», 1933). В аналитико-синтетических КС отсутствуют готовые индексы, классификационная схема предусматривает лишь основные деления и дополняется огромным количеством отдельных таблиц, разработанных для каждой отрасли знания в соответствии с характерными для нее категориями (объектами, процессами и т. п.). Присоединяя к основным делениям индексы таблиц категорий, можно получить обозначения для самых сложных понятий. Состав категорий и их последовательность в каждой отрасли знаний регламентируются фасетной формулой.

Фасетная классификация вместо единого ряда делений в каждом основном классе имеет несколько «фасетов», соответствующих аспектам классифицируемого понятия или предмета. Все существенные термины данного класса распределяются по фасетам и образуют их «фокусы». При индексировании документов их содержание выражается цепочкой фокусов, последовательность которых определяется специальной «фасетной формулой». Примером построения индексов по системе Ш. Ранганатана могут служить фасеты и фокусы из области медицины, представленные в таблице 3.3. По этому фрагменту фасетной классификации индекс документа по диагностике инфекционных заболеваний кишечника — 23:41:3, по лечению туберкулеза легких — 41:411:4.

Таблица 3.3

Фрагмент фасетной классификации по медицине

ФАСЕТЫ		
Органы тела	**Проблемы медицины**	**Уход и лечение**
1. Органы в целом 2. Органы пищеварения 21. Пищевод 22. Желудок 23. Кишечник 3. Кровеносная система 4. Органы дыхания 41. Легкие	1. Общие проблемы 2. Морфология 3. Физиология 4. Болезни 41. Инфекционные 411. Туберкулез	1. Питание 2. Этиология 3. Диагностика 4. Лечение

Преимущества этого вида классификаций в том, что они облегчают многоаспектное индексирование документов, позволяя собирать в одном месте все аспекты рассмотрения какого-либо предмета или темы, они легче поддаются изменениям при введении новых понятий, допускают большую глубину индексирования при более коротких индексах.

Их применение особенно эффективно при поиске в небольших по объему узкоспециализированных собраниях документов. Видный английский специалист по информационным технологиям Д. Фоскетт так обосновал достоинства фасетных классификаций: «От схемы не требуется более, чтобы она указывала *место* для каждого документа, включая любой термин или набор терминов в явном виде в классификационные таблицы по каждой предметной области. Эти схемы могут задать набор правил или рабочих процедур, с помощью которых такие контексты можно, по мере надобности, формулировать на основе тех же самых схем».

Предметизационные ИПЯ

В библиотеках распространены ***предметные*** каталоги, сущность которых заключается в том, что содержание документа кратко формулируется при помощи одного или нескольких типовых ключевых слов, получивших название ***предметных рубрик***, а затем предметные рубрики располагаются по алфавиту, и под каждой из них как под заголовком[1] собираются библиографические описания документов. Составление предметных рубрик и распределение по ним документов называется ***предметизацией***. Задача предметизации обычно состоит в том, чтобы указать главный объект рассмотрения в документе и, может быть, основные его аспекты и основные отношения к другим предметам.

В отличие от КС, предметизация распределяет документы по предметам или понятиям, не соотнося их с какими-либо областями знания. Это различие делает классификационный и предметизационный принципы организации документов независимыми, дополняющими друг друга, предназначенными для поиска документов по разным типам запросов. Предметизация дает возможность собирать в одном месте документы по таким комплексам, как конкретный материал, свойство, изделие, явление природы или общества, род деятельности, географическое понятие и т. п., собирая под каждым предметным заголовком весь массив знаний, безотносительно к тому, какой области науки эти знания принадлежат.

[1] Предметные рубрики также называются предметными заголовками (subject headings), и этот вариант термина лучше отражает природу термина, выбранного для обозначения содержания ряда документов в каталоге. Собственно предметными рубриками следует называть совокупности документов, объединенные одним предметным заголовком.

Другим отличием предметизации от классификации является то, что заранее составленный список предметных заголовков не ограничивает подробности анализа содержания документа. Если документ посвящен вопросу, не отраженному в списке предметных заголовков, всегда имеется возможность сформулировать новый заголовок самостоятельно. Обычно же при выборе предметной рубрики для документа руководствуются заранее составленным списком предметных заголовков. Но не представляет труда внести в него вновь образованную предметную рубрику, которая займет свое надлежащее место, определяемое алфавитным порядком. Этого обычно не удается делать в языках классификационного типа, где введение новых классов зачастую влечет преобразование большой части классификационных связей.

У каждого предметного заголовка могут в принципе быть подзаголовки, делящие документы в рубрике на подрубрики. В некоторых случаях к предметному заголовку могут быть даны ссылки на другие рубрики, где могут находиться документы по сходному предмету. Таким образом, в списке предметных заголовков одна запись может иметь довольно сложный характер. Вот пример фрагмента словаря предметных рубрик:

. . .
самовар
самолеты
– военные
– гражданские
– грузовые
– пассажирские
см. также авиалайнеры
саморезы
см. винты-саморезы
самоходные баржи
см. теплоходы грузовые речные

Традиционная каталожная техника не позволяет раскрывать содержание документа предметными рубриками с достаточной полнотой. Предметные заголовки отражают только основной предмет документа, и даже документы многопланового, обзорного характера могут быть отражены в каталоге лишь в ограниченном числе предметных рубрик. Недостаток места, трудоемкость составления картотечных каталогов заставляли библиографов разрабатывать довольно сложные правила оптимального выбора используемых предметных заголовков. Эти правила вместе со словарями рекомендуемых предметных заголовков составляют ИПЯ предметных рубрик, предметные ИПЯ.

Языки предметных рубрик используются не только как основа для предметного каталога документов, но также и как вспомогательное средство для пользователя систематическим каталогом, основанным на тематической классификации. В тех случаях, когда пользователь знает предмет своего интереса, но не знает, к какой отрасли знания он относится, он может обратиться к алфавитно-предметному указателю, в котором для каждой предметной рубрики указывают подходящий раздел (разделы) тематической классификации, в котором собраны знания по данному предмету. В этом случае мы имеем дело с языком предметных рубрик в функции поиска не документов, а тематических разделов систематического каталога, или, что то же самое, для поиска классов тематической классификации.

ИПЯ координатного индексирования

Современная компьютерная техника снимает ограничения по объему каталогов и снижает трудоемкость их составления. Поэтому получила распространение

идея приписывать документам все термины, существенные для выражения содержания документа, и в электронном каталоге иметь инверсный файл записи адресов документов, использовавших каждый такой термин, получивший наименование «ключевое слово».

Под ключевыми словами в данном случае понимаются наиболее существенные для выражения содержания документа полнозначные слова и словосочетания, обладающие назывной (номинативной) функцией. Поиск документа при этом должен происходить, как правило, не по одному ключевому слову (не по одной предметной рубрике, как в случае языка предметных рубрик), а по формулировке поисковой потребности, содержащей ряд ключевых слов, полно описывающих тему поиска. В процессе поиска по записям инверсного файла, соответствующим ключевым словам запроса, должны производиться логические операции над множествами адресов документов в соответствии с указаниями поискового предписания.

Ключевые слова образуют новый способ описания и поиска документов — язык ключевых слов (ИПЯ ключевых слов). Если в поисковой системе ключевые слова всей предметной области поиска сведены в словарь, где они связаны смысловыми отношениями так, что выражают структуру данной области знания, то такой словарь называют информационно-поисковым тезаурусом (см., например, общественно-политический тезаурус русского языка, используемый в УИС России, http://uisrussia.msu.ru/docs/ips/n/techno/index.htm), а термины, включенные в него, называют *дескрипторами*.

Дескрипторные информационно-поисковые системы открыли принципиально новую возможность поиска документов и содержащейся в них информации по любому сочетанию заранее не предвиденных признаков. В настоящее время производятся попытки сочетания уже имеющихся поисковых средств со вновь создаваемыми. Отсюда вытекает и желание найти общие черты в этих разных системах и лежащих в их основе ИПЯ: языке предметных рубрик и дескрипторном языке, что обычно сочетается с поисками путей их совместимости.

Всегда можно найти такую удаленную позицию, такое основание деления, при которых эти языки попадут в один общий класс. В ряду искусственных языков они принадлежат к классу информационных, в ряду информационных — к подклассу информационно-поисковых. В них используются в качестве индексов слова естественного языка. При построении этих языков применяются внешне схожие приемы: перечень предметных заголовков и словарная часть тезауруса упорядочиваются в алфавите слов. Тем не менее, учитывая эти общие и сходные черты, нельзя забывать и о принципиальных различиях данных языков. Основной словарный состав языка предметных рубрик — это имена сложных классов, построенных до индексирования документов, т. е. этот язык принадлежит к типу *предкоординируемых.* Дескрипторный же язык является *посткоординируемым,* т. е. строится из имен простых классов, которые образуют необходимые понятия при их пересечении (логическом умножении) в момент индексирования и/или поиска документов, т. е. после создания координатной сетки описания документов.

Для того, например, чтобы индексировать статью о производстве и экспорте вычислительных и пишущих машин в США, Японии и Великобритании достаточно дескрипторов *производство, экспорт, компьютер, пишущая машина* и названий трех этих стран. Тогда при поиске мы сможем сразу (за один шаг) найти этот документ при любом порядке перечисления дескрипторов (а таких комбинаций может быть $1\times2\times3\times4\times5\times6\times7 = 5040$, т. е. число перестановок из семи признаков). Если же пользоваться языком предметных заголовков, то в его словаре должны быть предусмотрены все осмысленные комбинации этих терминов,

чтобы был возможен тот же результат. Число осмысленных комбинаций, конечно, меньше указанного выше, но оно все равновелико, и в реальные словари предметных рубрик включается только малая часть их. Это, в свою очередь, приводит к необходимости дополнительного поиска именно той комбинации, которая включена в словарь, а чаще влечет за собой значительные поисковые потери. Так в реальный словарь в соответствии с данной тематикой реально могут быть включены предметные заголовки типа:

компьютеры — производство — Великобритания
компьютеры — производство — США
компьютеры — производство — Япония
пишущие машины — экспорт — Великобритания
пишущие машины — экспорт — США
пишущие машины — экспорт — Япония.

Однако трудно предположить, что в нем будет представлен хотя бы один предметный заголовок со всеми шестью терминами. Предметные заголовки, наподобие желаемого нами «компьютер — пишущая машина — производство — экспорт — Великобритания — США — Япония», не только практически нереальны, но и теоретически недопустимы в языках предметных рубрик.

Координатное индексирование в том и состоит, что для характеристики содержания документа или запроса перечисляются такие ключевые слова или дескрипторы, пересечение (логическое умножение) которых выражает основное смысловое содержание (главную тему, предмет) этого документа или запроса, тогда как в предметизации для данной цели используются заранее сформулированные по определенным правилам заголовки и подзаголовки.

Таблица 3.4

Обзор принципов индексирования [Сукиасян, 2006]

Принцип индексирования	**Классификационый**	**Предметизационный**	**Координатного индексирования**
ИПЯ	Классификационный	Предметизационный	Дескрипторный
Представление ИПЯ	Классификационная система	Словарь (список) предметных рубрик	Тезаурус
Термины индексирования	Классификационные индексы	Предметные рубрики	Дескрипторы
Характер поиска	Систематический (по иерархии понятий классификационной системы)	Алфавитный (по алфавиту предметных рубрик)	Многоаспектный (по основной, алфавитной, части тезауруса и по указателям)
Название процесса	Систематизация	Предметизация	Координатное индексирование
Название специалиста	Систематизатор	Предметизатор	Индексатор

Существует отдельный стандарт на данную тему — ГОСТ 7.25-2001 ТЕЗАУ

При индексировании, т. е. выражении основного смыслового содержания документа в терминах ИПЯ, процессы информационного анализа и синтеза совершаются в два этапа. Первый этап является общим для всех языков. Содержание документа анализируется как с позиций того, какие идеи и факты заложены в него автором, так и с позиций научных и практических интересов большинства его потенциальных читателей. (Если не иметь в виду узкоспециальные интересы,

то обе точки зрения чаще всего совпадают.) Результаты этого анализа синтезируются в виде субъективного представления индексатора об основном содержании документа. Второй этап зависит от языка индексирования. Если это предкоординированный алфавитно-предметный язык, то свое представление о содержании документа индексатор сверяет с потенциальными запросами читателей, отраженными в перечне предметных заголовков. Для посткоординируемого дескрипторного языка аналогичному анализу подвергается тезаурус (не связанный непосредственно с потенциальными запросами потребителей) и сам текст индексируемого документа. Синтез в данном случае выражается в выборе соответствующих предметных заголовков или дескрипторов (ключевых слов).

Информационно-поисковые тезаурусы

Теория и практика информационного поиска свидетельствуют, что наиболее перспективными и универсальными являются ИПЯ, основанные в той или иной мере на тезаурусе понятий, который отражает структуру онтологии предметной области поиска.

Информационно-поисковый тезаурус (ИПТ) — это контролируемый словарь терминов на естественном языке, явно указывающий отношения между терминами и предназначенный для информационного поиска. ИПТ относится к ИПЯ координатного типа.

Основными целями разработки традиционных ИПТ являются следующие:

- обеспечение перевода естественного языка документов и пользователей на контролируемый словарь, применяемый для индексирования и поиска;
- обеспечение последовательного использования единиц индексирования;
- описание отношений между терминами;
- использование как поискового средства при поиске документов.

РУС ИНФОРМАЦИОННО-ПОИСКОВЫЙ ОДНОЯЗЫЧНЫЙ. Правила разработки, структура, состав и форма представления.

Разные типы ИПЯ имеют свои достоинства и ограничения, которые делают их особо пригодными для решения разных поисковых задач. Возможности дескрипторного языка эффективно реализуются при узкотематическом поиске по произвольной комбинации признаков. Широкий тематический поиск по традиционным отраслям знаний и поиск по конкретным предметам, дисциплинам и их разделам в фондах документов за многие годы и в условиях одновременного обращения к ним большого числа читателей по-прежнему хорошо обеспечиваются библиотечными каталогами, основанными на иерархических и алфавитно-предметных классификациях. Выпуск информационных изданий требует разработки специальных рубрикаторов с небольшим числом уровней иерархии и подвижной, быстро меняющейся рубрикацией.

Глава 4

ПРЕДСТАВЛЕНИЕ ЗНАНИЙ

4.1. ТАБЛИЦЫ РЕШЕНИЙ И ТАБЛИЦЫ ОПЕРАТОРОВ

Представление знаний в табличной форме применялось еще в Вавилонском царстве, где правила умножения клинописных чисел наносились на глиняные таблички для изучения (рис. 4.1). Табличная форма записи легка для восприятия человеком. Правила в такой форме легко поддаются обработке или исполнению.

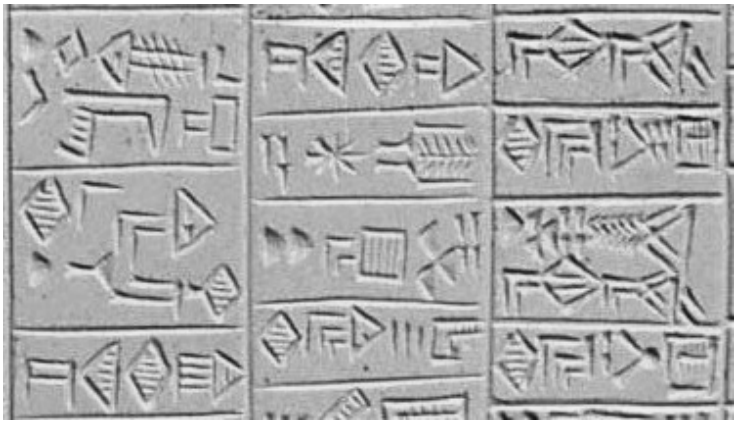


Рис. 4.1

Глиняные таблички в Вавилонском царстве

Табличное представление знаний в памяти компьютера было впервые применено в конце 1950-х гг. компаниями General Electric, The Sutherland Corporation, The United States Air Force (пример на рисунке 4.2) и получила название таблицы решений [Люгер, 2003]. Эти организации работали над проектом комплексного ведения файлов. Попытки поставить задачу с использованием традиционных блок-схем, структурных схем и комментариев закончились неудачей после шести лет работы, и это было типично для большинства больших проектов в те времена. Тогда в 1958 г. четыре аналитика, используя таблицы решений, смогли успешно поставить задачу менее чем за четыре недели, после чего она была решена в кратчайшие сроки.

Компания The Sutherland Corporation также использовала таблицы решений для составления спецификаций программ, после чего программистам оставалось лишь перевести данные из таблицы в машинные коды. General Electric, обнаружив, что блок-схемы и комментарии оказались полностью неприменимы для задач со сложной логикой, автоматизировала процесс непосредственной генерации программного кода из таблиц. В начале 1960-х гг. таблицы решений широко применяются для обобщения программного кода и разработки алгоритмов оптимизации. Название таблицы решений становится общепринятым названием данного метода. В 1970 г. канадская ассоциация по стандартизации вводит стандарт для таблиц решений (DECISION TABLES, CSA Standard Z243.1,1970).

В настоящее время таблицы решений широко используются в задачах, связанных с логической обработкой данных. На рынке представлено множество компьютерных программ, основанных на использовании данной парадигмы. В качестве

примера непосредственной реализации таблиц решений можно привести программу LogicGem фирмы Catalyst Development.

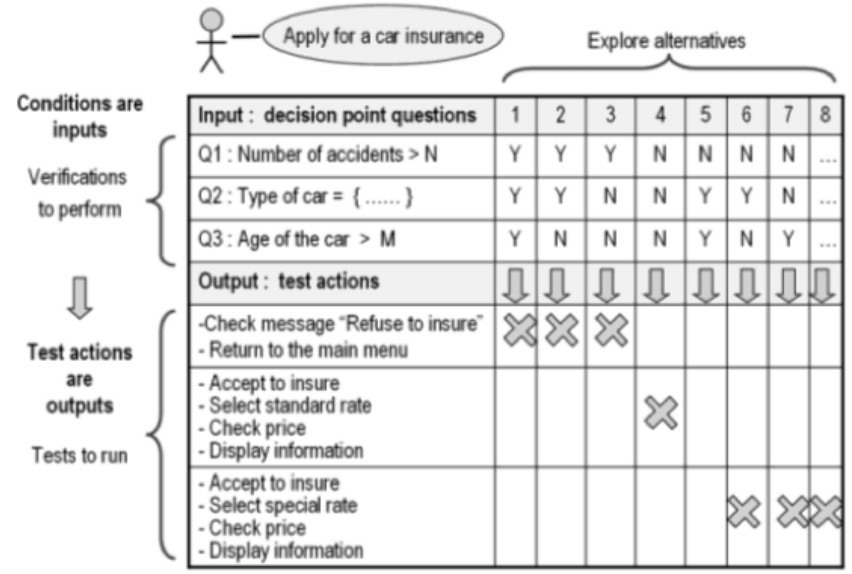


Рис. 4.2

Представление одной из первых таблиц решений

4.1.1. Общая структура и принципы таблицы решений

Таблицы решений (decision table), или как их еще называют, логические таблицы, отображают соотношения между комбинациями условий и комбинациями действий. В различных приложениях таблицы решений могут иметь различную форму (см., например, таблицу решений в главе 3, рис. 3.10). Одной из возможных форм является та, где каждое условие может быть сопоставлено с какой-либо определенной ситуацией или объектом (табл. 4.1).

Таблица 4.1

Структура таблицы решений

Условия:	**Ситуации:**			
	С1	*С 2*	...	*С N*
У1	1	0	...	1
...	...	...	...	...
УМ	1	1	...	0
Действия:				
Д1	0	1	...	1
...	...	...	...	...
ДК	1	0	...	1

Если некоторое условие или действие имеет место для данной ситуации, то в соответствующую ячейку таблицы заносится «1», в противном случае используется символ «0». Также можно задать последовательность действий, используя вместо символа «1» числа 1, 2, 3..., показывающие очередность выполнения действий.

Пример 4.1. Простая таблица решений

Рассмотрим неформальную задачу: предположим, необходимо ответить на вопрос «Чем заняться сегодня?» Одна из возможных таблиц решений для данной задачи представлена ниже (табл. 4.2).

Таблица 4.2

Пример таблицы решений со всеми возможными сочетаниями вариантов условий

Чем заняться сегодня?								
	1	2	3	4	5	6	7	8
Сегодня рабочий день?	*1*	*1*	*1*	*1*	*0*	*0*	*0*	*0*
Сегодня выходной?	*0*	*0*	*1*	*1*	*1*	*1*	*0*	*0*
Идет дождь?	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*0*
Отправиться на работу	*1*	*1*						
Поехать на пикник				*1*		*1*		*1*
Смотреть телевизор			*1*		*1*		*1*	

В данном примере представлена таблица с сочетаниями вариантов всех возможных условий (табл. 4.2). Если некоторое условие или действие имеет место для данной ситуации, то в соответствующую ячейку таблицы заносится «1», в противном случае используется символ «0». Также можно задать последовательность действий, используя вместо символа «1» числа 1, 2, 3..., показывающие очередность выполнения действий.

Каждый столбец позволяет сформировать одно простое правило поведения. Для столбца номер 1 это будет: «Если сегодня рабочий день и сегодня не выходной и идет дождь, то отправиться на работу». Однако достаточно беглого взгляда на таблицу чтобы определить главный недостаток данного примера — избыточность. Первая и вторая ситуация рекомендуют нам одно и то же — пойти на работу. Отличаются эти ситуации только одним условием: идет или не идет дождь. Следовательно, можно объединить эти ситуации в одну, поставив при этом против условия «идет дождь» символ «–» (не имеет значения). Аналогично можно поступить с ситуациями 3 и 5, 4 и 6. Таким образом, таблица решений примет следующий вид (табл. 4.3):

Таблица 4.3

Пример улучшенной таблицы решений

Чем заняться сегодня?								
	1	2	3	4	5	6	7	8
Сегодня рабочий день?	*1*	—	—	*0*	*0*			
Сегодня выходной?	*0*	*1*	*1*	*0*	*0*			
Идет дождь?	—	*1*	*0*	*1*	*0*			
Отправиться на работу	*1*							
Поехать на пикник			*1*		*1*			
Смотреть телевизор		*1*		*1*				

Первые три столбца таблицы теперь представляют составные правила для ситуаций, что позволяет значительно сократить размер таблицы.

В таблице решений выделяют четыре компонента: перечень условий, перечень действий, матрицу соответствия условий и матрицу возможных действий (табл. 4.4). Обычно в ячейках матрицы соответствия условий используются сим-

волы 1/0, Да/Нет, Y/N. Если наличие или отсутствие данного условия не критично для выбранной ситуации, можно также использовать символ «—», обозначающий «не имеет значения». Однако в общем случае в ячейках таблицы решений могут использоваться любые символы, отражающие суть условия или действия: значения и диапазоны значений, коды и номера, символьные данные и т. п. В качестве действия может выступать также набор инструкций, подлежащих выполнению в случае выполнения условий.

Таблица 4.4

Элементы таблицы решений

Перечень условий	Матрица соответствия условий
Перечень действий	Матрица возможных действий

Структура таблицы модифицируется под конкретную задачу. Например, можно добавить столбец с кодом операции, которую необходимо выполнять для проверки заданного условия. Возможно использование упрощенного варианта таблицы, состоящей из пар «условие–действие» без указания каких-либо объектов или ситуаций. Не представляет труда и расширение таблицы на новые классы объектов — необходимо лишь добавить в конец таблицы недостающие столбцы. А добавление новых строк, напротив, представляется несколько более сложным, так как потребуется внести изменения для всех существующих записей (столбцов).

Алгоритм поиска решения по таблице решений тривиален и прост в реализации. Интерпретирующая программа получает на вход конкретизированный набор условий для заданной ситуации и осуществляет последовательный перебор, подыскивая столбцы, где этот набор выполняется. Если такие столбцы находятся, то выполняется действие, указанное для этой ситуации.

Основным достоинством таблиц решений является высокая степень формализации, наглядности процесса принятия решений. Они строятся регулярным образом и могут наращиваться практически до бесконечности, т. е. являются универсальным средством решения задач, для которых возможно описание ситуаций с помощью ограниченного набора условий, например, проектирование деталей, представляющих собой поверхности вращения. Однако если различные ситуации характеризуются разными условиями, то таблицы решений становятся сильно разреженными и делают данный подход малоэффективным.

4.1.2. Методика формализации знаний с помощью таблиц решений

Таблицы решений позволяют сделать процесс формализации знаний достаточно простым, сводя его к нескольким типовым шагам. Для любой задачи эти шаги включают:

1. Описание всех требуемых действий.
2. Выявление перечня условий, необходимых для выполнения указанных действий.
3. Определение ключевого набора условий при решении проблемы.
4. Поочередное описание ситуаций для частных решений проблемы.

Содержание этих шагов можно рассмотреть на простом примере: пусть необходимо принять решение о пересечении улицы в зависимости от всевозможных факторов.

Описание всех требуемых действий. В данном примере возможны всего два действия: «перейти улицу» и «стоять на месте». В действительности же количество возможных действий никак не связано со степенью сложности решаемой задачи. Не следует вводить избыточные действия там, где можно упростить описание. Например, в ситуации с переходом улицы можно было бы ввести действия:

«перейти на зеленый свет светофора», «перейти на красный свет светофора», «перейти при любом сигнале светофора» и т. д. Очевидно, что такая избыточность не влияет на качество решения, так как в конечном итоге требуется принять решение идти или не идти. А вот практическое использование системы и ее поддержка в этом случае значительно усложняются.

Выявление перечня условий, необходимых для выполнения действий. Для выявления перечня возможных условий потребуются знание о предметной области, так как эти условия формулируются в терминах решаемой проблемы. В нашем примере мы можем ввести две группы условий: объективные и субъективные. Первые относятся к внешним факторам по отношению к лицу, принимающему решение. В данном примере это «свет светофора», «наличие движущихся машин» и «расстояние до них». Вторая группа условий характеризует некоторые внутренние характеристики проблемы. Для ситуации с пересечением дороги можно учесть, например, насколько спешит переходящий ее человек и насколько он склонен к риску.

Определение ключевого набора условий при решении проблемы. От ключевого набора условий зависит глобальное решение проблемы, т. е. при нарушении этих условий получить решение невозможно в принципе. Очевидно, при переходе дороги таким условием является наличие вблизи движущейся машины. Если машина есть, то дорогу переходить нельзя.

Поочередное описание ситуаций для частных решений проблемы. При описании частных решений не обязательно рассматривать все возможные варианты комбинаций условий. Как было показано выше, можно комбинировать условия, используя неопределенные значения в ячейках таблицы. Также можно использовать логическое отрицание, которое в ряде случаев упрощает описание ситуации. Например, вместо перечисления различных сигналов светофора, при котором действие разрешено, можно указать то, при котором оно запрещено «не красный» или «не зеленый» и т. п.

Одна из возможных таблиц решений для примера с пересечением улицы может выглядеть так, как показано в таблице 4.5.

Таблица 4.5

Пример таблицы решений для ситуации с пересечением улицы

Можно ли переходить дорогу?							
	1	2	3	4	5	6	...
Есть движущаяся машина (1 — да, 0 — нет)	*1*	*1*	*1*	*0*		*1*	
Расстояние до машины (1 — близко, 2 — далеко)	*1*	*2*	*2*			*2*	
Свет светофора (1 — зеленый, 2 — желтый, 3 — красный)			*3*	*не 1*	*1*	*2*	
Пешеход спешит (1 — да, 0 — нет)				*1*			
Пешеход склонен к риску (1 — да, 0 — нет)		*1*					
Стоять на месте	*1*		*1*			*1*	
Перейти дорогу		*1*		*1*	*1*		

После формирования таблицы можно приступить к ее оптимизации и отладке. Для этих целей удобно использовать специализированные программные решение, например, LogicGem, специализированные языки программирования или макросы для электронных таблиц. Наконец, после того, как таблица полностью сформирована, на ее основе можно сформулировать набор продукционных правил для экспертной системы (см. ниже).

Рассмотрим еще один пример построения таблицы решений на примере выбора оптических элементов в задаче синтеза оптической системы (объектива) в зависимости от значений параметров классификации по техническим характеристикам: J — светосила системы, W — угловое поле, F — фокусное расстояние, L — хроматический диапазон, Q — качество изображения (степень аберрационной коррекции), S — задний фокальный отрезок, D — положение входного зрачка.

1. Описание всех требуемых действий. В данном примере возможны всего четыре действия: рекомендовать элемент из списков базовых (B), коррекционных (C), светосильных (T) или широкоугольных (Y). Как уже было сказано, не следует вводить избыточные действия там, где можно упростить описание. Например, в данной ситуации можно было бы ввести действия для всех возможных разновидностей оптических элементов с учетом формы поверхности и толщины. Очевидно, что такая избыточность не влияет на качество решения, поскольку в конечном итоге требуется принять решение о выборе конкретного элемента из списков B, C, T или Y, поэтому достаточно просто указать его номер или код.

2. Выявление перечня условий, необходимых для выполнения указанных действий. Для выявления возможных условий потребуются знания о предметной области, поскольку эти условия формулируются в терминах решаемой проблемы. В нашем примере можно ввести три группы условий: по оптическим характеристикам (параметры J, W, F), по назначению (L, Q) и по конструктивным особенностям (S, D). Каждое из перечисленных условий разделяется на три диапазона значений, обозначенных индексами «0», «1» и «2», которые и будут записываться в таблице решений. Это значительно упрощает восприятие таблиц решений, так как вместо различных единиц измерения и различных диапазонов используются унифицированные обобщенные индексные значения.

3. Определение ключевого набора условий при решении проблемы. В данном примере один из параметров классификации по техническим характеристикам является ключевым, а именно D — положение входного зрачка.

4. Поочередное описание ситуаций для частных решений проблемы. Повторимся, что при описании частных решений необязательно рассматривать все возможные варианты комбинаций условий и можно комбинировать условия, используя неопределенные значения в ячейках таблицы, а также логическое отрицание. Таким образом, для каждого оптического элемента достаточно указать одно или несколько условий, а не все.

Один из возможных вариантов решений для описываемого примера приведен в таблице 4.6.

Таблица 4.6

Пример таблицы решений для выбора элементов оптической системы

	1	2	3	8
J — светосила системы	*не 2*	—	—	
W — угловое поле	*не 2*	*не 0*	—	
F — фокусное расстояние	—	—	—	
L — хроматический диапазон	—	—	—	
Q — качество изображения	—	—	—	
S — задний фокальный отрезок	—	—	*не 2*	
D — положение входного зрачка	*не 2*	—	*не 1*	
Оптический элемент B	*1P1A*			
Оптический элемент K				
Оптический элемент C			*303I*	
Оптический элемент Y		*2P2P*		

4.1.3. Программное обеспечение для работы с таблицами операторов

В настоящее время таблицы решений широко используются в задачах, связанных с логической обработкой данных. На рынке представлено множество компьютерных программ, основанных на использовании данной парадигмы. В качестве примера непосредственной реализации таблиц решений можно привести программу LogicGem фирмы Catalyst Development (рис. 4.3).

Демонстрационную версию этой программы можно найти на сайте разработчика http://download.cnet.com/LogicGem/3000-2383_4-75848455.html. Ссылка на методическое пособие http://books.ifmo.ru/file/pdf/497.pdf.

Одним из самых современных решений для работы с таблицами решений является модуль Drools Expert в системе Drools. Этот модуль позволяет редактировать базу знаний с помощью табличного интерфеса (рис. 4.4). Ссылка на сайт разработчика http://www.jboss.org/drools/drools-expert.html. Методическое пособие расположено по ссылке http://window.edu.ru/resource/741/79741/files/itmo1080.pdf.

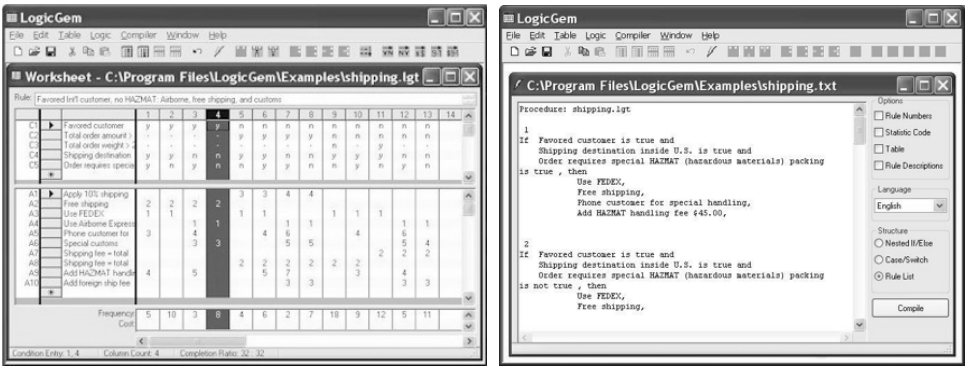


Рис. 4.3

Интерфейс программы LogicGem

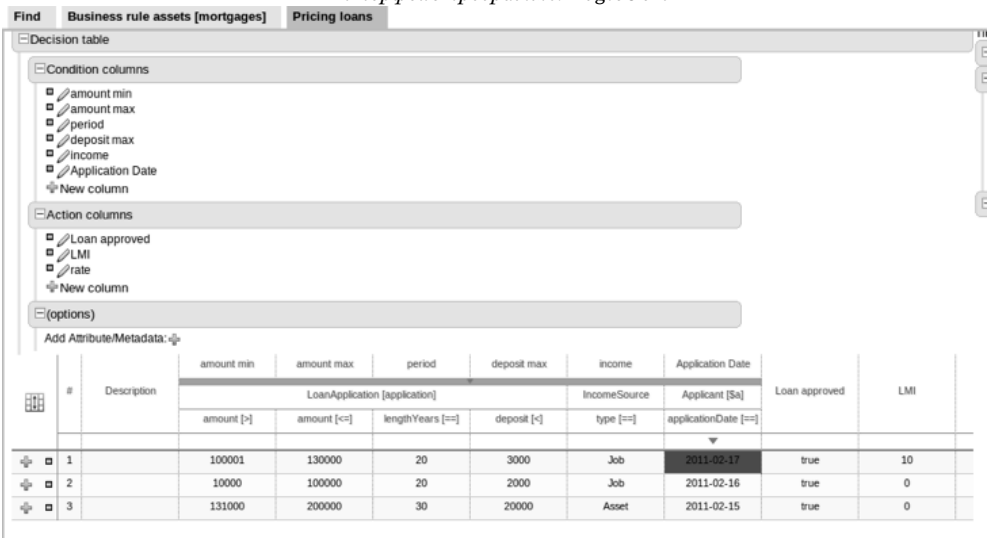


Рис. 4.4

Интерфейс Drools Expert

4.1.4. Таблицы операторов в системе STRIPS

Подход, во многом близкий к описанному выше, был предложен в системе STRIPS (сокр. от STanford Research Institute Problem Solver). Эту систему разработали в 1971 г. два исследователя — Ричард Файкс (Fikes) и Нильс Нильсон (Nilsson) [Fikes, Nilsson, 1971]. Программа предназначалась для решения проблемы формирования плана поведения мобильного робота Shakey (рис. 4.5), перемещающего предметы через множество комнат. Текущее состояние окружающей среды — помещений и предметов в них — представляется набором выражений *предикат–аргумент*, которые в совокупности образуют модель мира. Поиск решения в этой системе был основан на методологии анализа целей и средств (mean–ends analysis) системы General Problem Solver, а для проверки истинности предварительных условий действия в текущем состоянии применялась методология доказательства теорем из системы QA3.

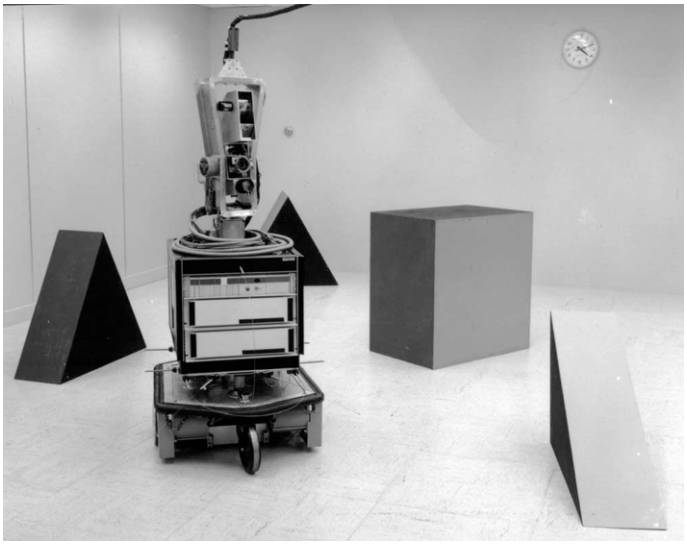


Рис. 4.5
Робот Shakey

Общепринятая для ИИ конструкция *предикат–аргумент*, определяется следующим образом:

<предикат> ::= <предикатный символ> (<аргумент$_1$>, ..., <аргумент$_n$>).

Различают одноместные или *n*-местные предикаты. В случае одноместного предиката считается, что аргумент обладает свойством, выраженным предикатным символом. *N*-местный предикат описывает отношение между объектами, которые заданы аргументами.

Примером ситуации, описываемой с помощью двухместного предиката, может быть нахождение робота в определенной комнате:

at(robot, roomA).

Данный предикат означает, что объект *robot* находится в комнате *roomA*. Порядок следования аргументов в предикате определяется разработчиком, но должен быть неизменным везде, где этот предикат используется. Можно было бы

написать *at(roomA, robot)* и придать этой записи аналогичный смысл, но использовать одновременно в пределах одной базы знаний записи *at(robot, roomA)* и *at(roomA, robot)* нельзя.

С помощью множества предикатов можно описать текущую модель мира, т. е. набор конкретных объектов, их свойств и отношений. Например, исходная ситуация может описываться следующим множеством:

$$W1 = \{ at(robot, roomA), at(box1, roomB), at(box2, roomC) \}.$$

Конечная ситуация также задается множеством предикатов:

$$WK = \{ at(box1, roomA), at(box2, roomB) \}.$$

Для описания действий, которые может выполнять робот, используются операторы, применяемые к текущей модели мира. Эти операторы позволяют добавлять или изымать некоторые факты из текущей модели. Например, действие

«Переместить робота из комнаты А в комнату В»

в модели W^1 приводит к формированию новой модели W^2. При этом факт *at(robot, roomA)* будет изъят из модели, а *at(robot, roomB)* добавлен.

Множество допустимых операций, таких как перемещение робота или перенос предметов, кодируются в таблице операторов (табл. 4.7), которая близка по структуре к таблице решений, но предполагает обязательное наличие модели мира.

Таблица 4.7

Пример таблицы операторов

Оператор	*move(X, Y)*	*push(X, Y, Z)*
Предварительные условия	*{ at(robot, X) }*	*{ at(robot, Y), at(X, Y) }*
Список удалений	*{ at(robot, X) }*	*{ at(robot, Y), at(X, Y) }*
Список добавлений	*{ at(robot, Y) }*	*{ at(robot, Z), at(X, Z) }*

Как уже было отмечено, задачей системы STRIPS является формирование плана действий робота для достижения цели. Таким образом, результатом работы системы должна быть последовательность операторов, применение которых к исходной модели мира позволяет достичь целевой модели. Зная целевое состояние среды, можно было бы перебирать последовательно или случайно комбинации операторов, пока цель не будет достигнута. Но экспоненциальный рост количества вариантов при каждой новой проверке делает такой подход неприемлемым на практике.

Для предотвращения экспоненциального роста вариантов возможных решений в качестве основы для работы системы был предложен метод «анализ целей и средств», идея которого состоит в том, чтобы с каждой новой операцией отличие между текущим состоянием и целевым уменьшалось. Это предполагает наличие меры оценки «расстояния» в пространстве решений. Например, если очередная подцель сформулирована в виде предиката:

$$at(box1, roomA),$$

а ящик *box1* находится в комнате *roomB*, то перемещение робота из комнаты *roomA* в комнату *roomC* не приблизит текущее состояние к целевому (для модели W^1). А перемещение из комнаты *roomA* в комнату *roomB*, наоборот, уменьшит расстояние между текущим и целевым состоянием, так как позволит на следующем шаге переместить ящик *box1* в комнату *roomA*.

Алгоритм поиска требуемых операторов системы STRIPS основан на сопоставлении очередной подцели и списков добавления в таблице операторов. Для того чтобы узнать, применимо ли действие в некотором заданном состоянии, мы должны проверить истинность формулы в предварительном условии, т. е. доказать, что эта формула является логическим следствием из аксиом модели мира данного состояния. Если предварительное условие истинно, то действие применимо. Новые подцели выбираются из списка предварительного условия найденного оператора. Так, например, цель *at(box1, roomA)* соответствует элементу *at(X, Z)* в списке добавлений оператора *push(X, Y, Z)*. Унификация этих двух предикатов (*X* соответствует *box1*, а *Z* — *roomA*) позволяет выбрать из предварительного условия оператора новые подцели:

at(robot, Y), at(box1, Y).

Далее необходимо найти в модели мира предикат, содержащий объект (в данном случае это комната), который может конкретизировать символ *Y*. Таким предикатом может быть, например, *at(box1, roomB)*. Поставив в соответствие с символом *Y* объект *roomB*, можно окончательно сформулировать очередные подцели:

at(robot, roomB) и at(box1, roomB).

Теперь первый элемент в этом списке указывает желаемое (целевое) положение робота, а второй элемент уже присутствует в модели мира *W1*. Следовательно, после применения оператора *push* к модели мира *W1*, необходимо добавить *at(robot, roomB)* и удалить *at(robot, roomA)*. В результате получится новая модель:

W2 = { at(robot, roomB), at(box1, roomB), at(box2, roomC) }.

Отметим, что если формула из списка добавления уже присутствует в текущем состоянии модели мира (состоянии, к которому применяется действие), то эта формула не будет добавлена вторично. Аналогично, если формула из списка удаления уже отсутствует в текущем состоянии, то ее удаление игнорируется. Все формулы, которые не упоминаются в таблице оператора, остаются неизменными в новом состоянии. Это последнее утверждение в литературе называют *предположением STRIPS* (STRIPS assumption). Описанная процедура повторяется до тех пор, пока очередная модель не будет соответствовать целевой.

Иллюстрация работы описанного алгоритма представлена на рисунке 4.6.

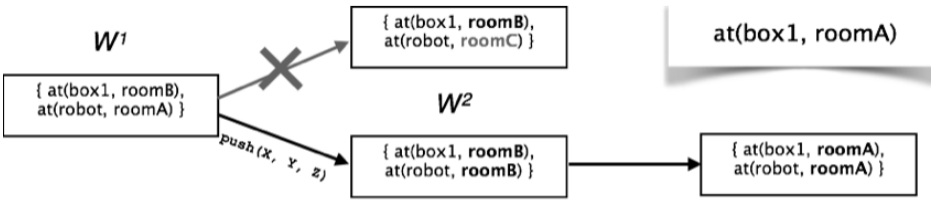


Рис. 4.6

Иллюстрация работы алгоритма «анализа целей и средств»

Так как таблица операторов, модель мира и цели представлены с помощью одного и того же синтаксиса в виде конструкций *предикат–аргумент*, то, применяя описанную выше схему сопоставления, программа довольно просто находит, какие именно операции нужно выполнить для достижения поставленной цели.

Все, что нужно для этого сделать, — просмотреть списки добавлений в описании операторов и найти в них элемент, соответствующий заданной цели.

Однако существуют ситуации, в которых система STRIPS оказывается неспособной решать поставленные задачи. Одна из таких ситуаций получила название аномалия Зюссмана (Sussman Anomaly).

4.2. ПРОДУКЦИОННЫЕ СИСТЕМЫ И СПОСОБЫ ОРГАНИЗАЦИИ РАССУЖДЕНИЙ

Продукционная система (production system) — это модель представления, основанная на *продукционных правилах* (production rule), позволяющая описывать знания о решении задач в виде правил «ЕСЛИ *условие*, ТО *действие*» (IF... THEN...). Понятие «продукционные системы» является частным случаем *систем, основанные на знаниях* (knowledge based systems). Впервые идея представления знаний в виде продукций появилась в работе Эмиля Поста (Emil Leon Post) в 1943 г. в ходе исследований нормальных исчислений, являющихся частным случаем канонической системы Поста [Post, 1943]. Формальные вычисления на множестве продукций основаны на теории синтаксического анализа. А управление продукционными системами основано на Марковских алгоритмах, представляющих собой упорядоченную группу продукций, применяемых в порядке приоритета к входной строке.

Пост доказал, что любая система в математики или логике может быть представлена в виде продукционных правил. Другими словами, продукционная система эквивалентна машине Тьюринга, что делает данный подход принципиально пригодным для решения широкого класса задач. Таким образом, продукционная система является общей моделью вычислений, которая может быть запрограммирована для решения любых задач. Но наиболее эффективным применением этой модели являются интеллектуальные задачи.

Правила обеспечивают формальный способ представления рекомендаций, указаний или стратегий. Они идеально подходят в тех случаях, когда знания предметной области возникают из эмпирических ассоциаций, накопленных за годы работы по решению задач в данной области. Представления знаний в виде продукций наиболее распространено в экспертных системах, так как запись знаний фактически ведется на подмножестве естественного языка. Следствием этого является то, что правила легко читаются, их просто понять и модифицировать, эксперты без труда могут сформулировать новое правило или указать на ошибочность какого-либо существующего.

Интересным и важным приложением метода продукционных систем является программа General Problem Solver (GPS), разработанная Гербертом Саймоном (Herbert Simon) и Алланом Ньюэллом (Allen Newell) в 1960–1970-х гг. в Carnegie Institute of Technology [Newell, Shaw, Simon, 1959]. Система GPS осуществляла моделирование решение различных задач с помощью так называемых *протоколов решения задач*. На основе этих протоколов строился *граф поведения задачи*, поиск по которому осуществлялся с помощью продукционной системы.

4.2.1. Синтаксис продукционной модели и архитектура продукционной системы

Основными компонентами архитектуры продукционной системы являются (рис. 4.7):

- БЗ продукционных правил;
- рабочая память;
- цикл управления «распознавание–действие».

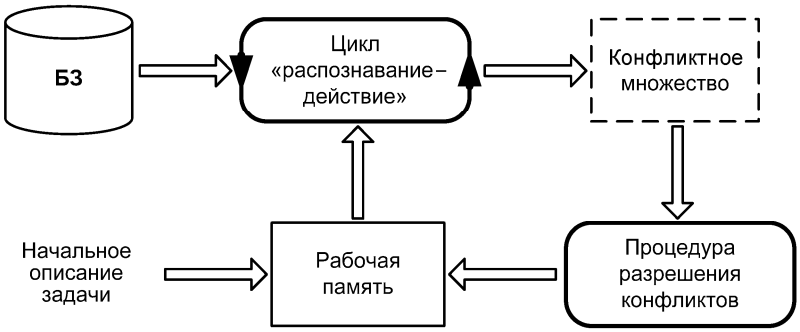


Рис. 4.7
Архитектура продукционной системы

Моделирование решения задачи основано на процессе *сопоставления с образцом* (pattern matching), в ходе которого текущее состояние решения сравнивается с имеющимися знаниями для определения дальнейших действий.

В БЗ содержится множество *продукционных правил* или просто *продукций* (productions), являющихся парами условие–действие, которые определяют элементарные шаги решения задачи. *Условная часть* (IF–part) правила является шаблоном (образцом), по которому можно определить, в какой момент необходимо использовать (активировать) данное правило для выполнения очередного этапа решения задачи. *Часть действия* (THEN–part) описывает соответствующий шаг решения. Условную часть правила также называют *антецедентом* (antecedent), а часть действия — *консеквентом* (consequent).

Рабочая память (working memory) содержит текущее описание модели мира в процессе рассуждений. В этой модели содержится набор образцов, инициализируемый начальным описанием задачи.

В управляющем цикле *распознавание–действие* (the recognize–act cycle) осуществляется сравнение образцов из рабочей памяти с условными частями правил в БЗ (см. рис. 4.8). Если условие какого-либо правила соответствует образцу, то это правило помещается в *конфликтное множество* (conflict set). Продукции, содержащиеся в конфликтном множестве, называют *допустимыми*, так как они согласованы с текущим состоянием рабочей памяти. После того, как закончит работу цикл «распознавание–действие», осуществляется процесс *разрешения конфликтов* (conflict resolution), в ходе которого выбирается и активизируется (возбуждается) одна из допустимых продукций. Наконец, в соответствии с частью действия активированного правила осуществляется модификация рабочей памяти. Весь этот процесс повторяется до тех пор, пока образцы в рабочей памяти не будут соответствовать ни одному из правил БЗ.

Стратегии разрешения конфликтов отличаются в различных реализациях продукционной модели и могут быть достаточно простыми. Например, выбирается первое из допустимых правил. Однако многие системы допускают использование сложных эвристик для осуществления выбора из конфликтного множества. Например, в системе OPS5 поддерживаются следующие стратегии разрешения конфликтов [Brownston, Farrel, Kant, Martin, 1985]:

1. *Рефракция* (refraction) для предотвращения зацикливания: после активизации правила оно не может быть использовано снова, пока не изменится содержимое рабочей памяти.

2. *Новизна* (recency) позволяет сосредоточить поиск на одной линии рассуждения: предпочтение отдается правилам, в условии которых встречаются факты, добавленные в рабочую память последними.



Рис. 4.8

Работа управляющего цикла «распознавание–действие»

3. *Специфичность* (specifity) отдает предпочтение более конкретным правилам перед более общими: одно правило более специфично (конкретно), чем другое, если оно содержит больше фактов в условной части.

В современной системе DROOLS имеется значительное количество реализованных стратегий, а также можно определить свою.

1. Приоритет («выпуклость», Salience) является формой приоритета. Каждое правило имеет целочисленный атрибут Salience, по умолчанию нулевой. Правила с более высокими значениями получают более высокий приоритет в очереди активации.

2. Новизна (recency) учитывает счетчик, изменяемый при каждой активации правила. Приоритет отдается правилам с более высоким значением счетчика.

3. Первенство (Primacy) является противоположной стратегией по отношению к «новизне». Приоритет отдается правилам с более низким значением счетчика.

4. FIFO (First In First Out) — стратегия в глубину. Только что активированные правила помещаются вверх списка конфликтного множества.

5. LIFO (Last In First Out) — стратегия в ширину. Только что активированные правила помещаются вниз списка.

6. Сложность (Complexity) — это стратегия, основанная на специфичности и принимающая во внимание сложность конфликтующих правил.

7. Простота (Simplicity) — это также стратегия, основанная на специфичности, принимающая во внимание простоту конфликтующих правил.

8. Порядок загрузки (LoadOrder) — правила с большим числом LoadOrder размещаются в верхней части списка.

9. Случайность (Random) — активируется случайно выбранное из конфликтного множества правило.

Чистая продукционная модель не предусматривает выхода из тупиковых ситуаций в процессе поиска. Работа продолжается, пока не исчерпаны все возможные варианты. Гораздо более эффективным является модификация цикла управления с механизмом *возврата* (cycle back) в предыдущее состояние модели мира, позволяющего находить более точные решения задачи.

Продукционная модель позволяет имитировать различные аспекты поведения человека при решении задач (рис. 4.9). БЗ продукций соответствует навыкам решения задач в *долговременной памяти*. Подобно знаниям в долговременной памяти продукционные правила не изменяются в процессе решения задачи. Новые навыки могут быть просто добавлены в БЗ при необходимости. Рабочая память соответствует *кратковременной памяти*. В ходе решения задачи «фокус внимания» переходит от одного шага к другому, при этом после получения решения содержимое рабочей памяти не сохраняется.

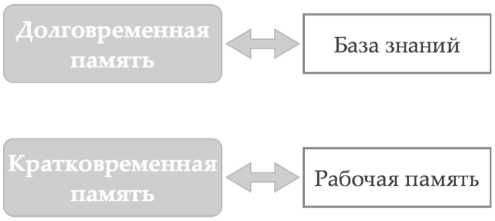


Рис. 4.9

Аналогия продукционной модели и человеческой памяти

В качестве *условия* и *действия* в правилах может быть, например, предположение о наличии того или иного свойства, принимающее значение *истина* или *ложь*. При этом термин *действие* следует трактовать широко: это может быть директива к выполнению какой-либо операции, рекомендация или модификация базы знаний — предположение о наличии какого-либо производного свойства.

Примером продукции может служить следующее выражение:

П1: *ЕСЛИ* клиент работает на одном месте более двух лет,
ТО клиент имеет постоянную работу.

Как условие, так и действие правила могут учитывать несколько выражений, объединенных логическими связками *И*, *ИЛИ*, *НЕ*:

П2: *ЕСЛИ* клиент имеет постоянную работу
И клиенту более 18 лет
И клиент НЕ имеет финансовых обязательств,
ТО клиент может претендовать на получение кредита.

Помимо продукционных правил база знаний должна включать и простые *факты*, поступающие в систему через интерфейс пользователя или выводимые в процессе поиска решения задачи. Факты являются простыми утверждениями типа «клиент работает на одном месте более двух лет». Факты, являясь истинными утверждениями, копируются в рабочую память для использования в цикле «распознавание–действие».

Последовательное активирование правил порождает *цепочку вывода* (reasoning). Цепочка вывода, полученная в результате последовательного выполнения правил П1 и П2, показана ниже (рис. 4.10). Эта цепочка показывает, как на основании правил и исходных фактов выводит заключение о возможности получения кредита. В ходе первой итерации вывода управляющий цикл, сопоставляя факт «клиент работает на одном месте более двух лет» и правило П1, добавляет в рабочую память новый факт (образец) «клиент имеет постоянную работу».

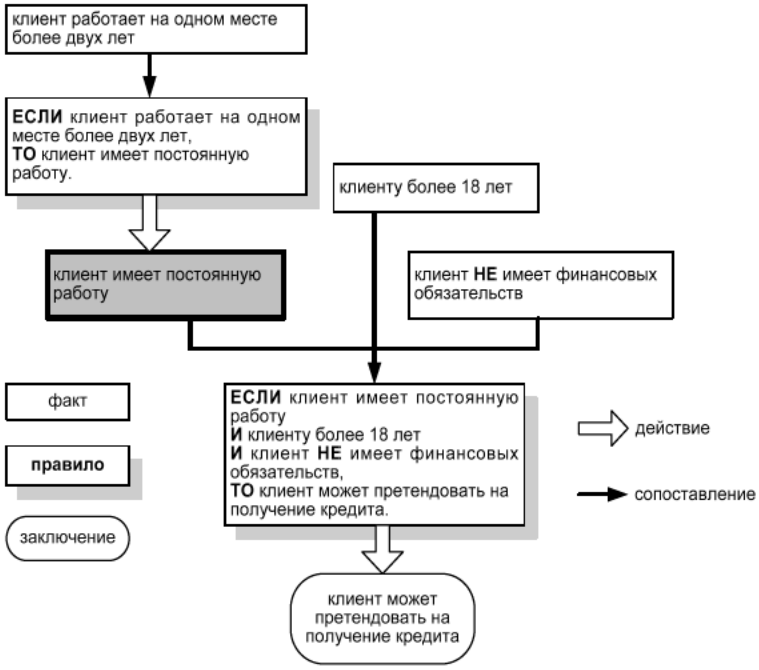


Рис. 4.10

Пример цепочки вывода в продукционной системе

Далее на основании содержимого рабочей памяти и правила П2 выводится новый факт «клиент может претендовать на получение кредита», который и является окончательным решением, так как цикл «распознавание–действие» останавливается и процесс вывода на этом завершается.

Монотонным выводом в продукционных системах называют вывод, при котором факты не удаляются из рабочей памяти. *Немонотонный вывод* допускает удаление фактов из рабочей памяти. При немонотонном выводе существенную роль играет порядок применения продукционных правил.

Монотонным выводом в продукционных системах называют вывод, при котором факты не удаляются из рабочей памяти. *Немонотонный вывод* допускает удаление фактов из рабочей памяти. При немонотонном выводе существенную роль играет порядок применения продукционных правил.

Цепочки вывода экспертной системы могут быть предъявлены пользователю и помогают понять, как было получено решение. Также визуализация этого процесса может быть полезна в процессе отладки БЗ, так как эксперт получает возможность «увидеть» неправильный ход рассуждений. Большинство реализаций продукционных систем имеют встроенные средства для интроспекции БЗ и мониторинга работы машины вывода.

Вывод решения может осуществляться в двух направлениях: от данных задачи к цели и в обратном направлении от цели к исходным данным. *Вывод на основе данных* (data–driven search), процесс решения задачи начинается с исходных фактов. Затем, применяя допустимые правила, осуществляется переход к новым фактам. И так до тех пор, пока цель не будет достигнута. Это процесс также называют *прямой цепочкой вывода* (forward chaining). *Вывод от цели* (goal–

directed strategy) начинается от одной из допустимых целей, и рассматриваются пути, ведущие к достижению этой цели. Таким образом, определяется последовательность правил, позволяющих найти решение. Процесс повторяется для всех заданных в задаче целей. Такой способ поиска называют также *обратной цепочкой вывода* (backward chaining). Выбор направления поиска зависит от конкретной задачи, структуры исходных данных и целей, способов реализации алгоритмов поиска. Следует отметить, что, как правило, в реальных системах используются комбинации обоих способов вывода.

4.2.2. Прямая цепочка рассуждений на основе данных

Прямая цепочка рассуждений применяется в задачах, где на основании имеющихся фактов необходимо определить тип (класс) объекта или явления, выдать рекомендацию, определить диагноз и т. п. К таким задачам относятся, например, интерпретация данных, проектирование, классификация и пр. (см. главу 2). Вывод на основе данных применим к решению задач в следующих случаях:

Все или большинство данных заданы в пространстве задачи. Например, задача интерпретации состоит в выборе этих данных и в представлении их для использования в системах интерпретации более высокого уровня.

Существует большое количество потенциальных целей, но всего лишь несколько способов представления и применения исходных фактов. Примером является экспертная система DENDRAL, предназначенная для исследования молекулярных структур органических соединений на основе формул, данных масс-спектрографа и знаний из химии. Для любого органического соединения существует чрезвычайно большое число возможных структур. Однако данные масс-спектрографа позволяют программе оставить лишь небольшое число таких комбинаций.

Сформировать цель или гипотезы очень трудно в силу избыточности исходных данных или большого числа конкурирующих гипотез.

Таким образом, при поиске на основе данных исходные факты в постановке задачи используются для формирования возможных путей ее решения. Алгоритм прямого вывода обычно основан на *стратегии поиска в ширину*: в рабочую память помещаются исходные факты и далее ее содержимое последовательно сравнивается с предпосылками каждого правила в БЗ (рис. 4.11). Если содержимое рабочей памяти приводит к активизации какого-либо правила, то после модификации рабочей памяти анализируется следующее правило. Когда первый проход по правилам завершается, процесс повторяется, начиная с первого правила.

Модификацией рассмотренной стратегии является так называемый *оппортунистический поиск* (opportunistic reasoning). Это простая, но более эффективная стратегия поиска: при активизации правила, переход осуществляется к правилу, содержащему в условной части вновь выведенные факты.

Пример 4.2. Прямая цепочка рассуждений

Рассмотрим пример миниатюрной экспертной системы для фондовой биржи из книги [Levine, Drang, Edelson, 1988]. БЗ подобной ЭС может включать, например, следующие продукционные правила:

ЕСЛИ *Процентные ставки падают*, ТО *Уровень цен на бирже растет*.

ЕСЛИ *Процентные ставки растут*, ТО *Уровень цен на бирже падает*.

ЕСЛИ *Валютный курс доллара падает*, ТО *Процентные ставки растут*.

ЕСЛИ *Валютный курс доллара растет*, ТО *Процентные ставки падают*.

ЕСЛИ *Процентные ставки федерального резерва падают* И *Средства федерального резерва добавлены*, ТО *Процентные ставки падают*.



Рис. 4.11

Алгоритм прямого вывода

Предположим, пользователь сообщил системе, что валютный курс доллара падает по отношению к основным валютам других стран, и попросил консультацию в области биржевых операций. Цель в данном случае заключается в выборе правильного поведения на бирже, но останется ли при этом клиент в выигрыше, зависит от пока еще не определенных условий. На основании запроса пользователя инициализируется исходное состояние рабочей памяти путем добавления в нее факта:

Валютный курс доллара падает.

Значит, исходное состояние системы будет выглядеть следующим образом (рис. 4.12):

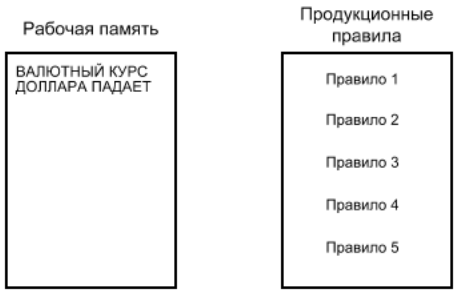


Рис. 4.12

Исходное состояние продукционной системы

Для приведенного примера необходимо, чтобы в условной части какого-либо правила содержалось бы условие:

Валютный курс доллара падает.

Такое условие содержится только в правиле 3:

ЕСЛИ *Валютный курс доллара падает*, ТО *Процентные ставки растут*.

В соответствии с этим правилом можно сделать вывод о росте процентных ставок. О «Валютном курсе доллара» упоминается также в правиле 4. Но условие, записанное в этом правиле, звучит следующим образом:

ЕСЛИ Валютный курс доллара растет, ТО...

Это условие не соответствует исходному состоянию падения валютного курса доллара, и поэтому правило 4 в дальнейших рассуждениях не будет участвовать. Следовательно, единственное изменение на этом шаге будет произведено после возбуждения правила 3, и в рабочую память добавится новый факт:

Процентные ставки растут.

Состояние системы станет следующим (рис. 4.13):

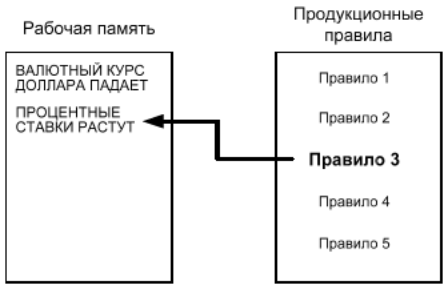


Рис. 4.13

Первый шаг прямого вывода

Рассуждения еще не закончены, так как необходимо проверить, не приведет ли добавление в рабочую память нового факта к другим выводам. В условной части правила 2 как раз и используется данный факт:

ЕСЛИ *Процентные ставки растут*, ТО *Уровень цен на бирже падает*.

Соответственно, после активизации этого правила в рабочую память системы добавляется факт:

Уровень цен на бирже падает.

Теперь система будет выглядеть так, как на рисунке 4.14.

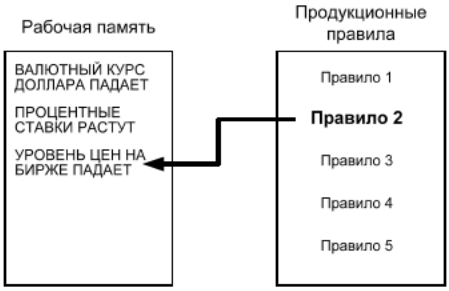


Рис. 4.14

Второй шаг прямого вывода

Рассуждения продолжаются подобным образом, пока не будут рассмотрены все возможные ситуации. Если после выполнения проверки всех правил, но ни в одном из них в условной части не упоминается уровень цен на бирже, то на этом рассуждения заканчиваются. Пользователю можно сообщить следующее: «Когда обменный курс доллара падает, растут процентные ставки и уровень цен на бирже падает». В реальной жизни такое заключение потребовало бы более сложных правил, однако система, реализующая прямую цепочку рассуждений, оперирует только теми данными, которые есть в базе знаний.

Факты для активизации правил могут выбираться не только из рабочей памяти, а например, после запроса «Что будет, если добавить в банковскую систему средства из федерального резервного банка?» Этот запрос соответствует добавлению в рабочую память факта:

Средства федерального резерва добавлены.

Просмотрев список продукций, система определяет, что этот факт упоминается в условной части правила 5. Но в условии этого правила содержится также факт:

Процентные ставки федерального резерва падают.

Этот факт отсутствует в рабочей памяти и не может быть выведен из других правил. Следовательно, система должна запросить у пользователя является ли этот факт истинным? Обычно в продукционных системах факты связывают с вопросами типа «Растет или падает уровень процентных ставок Федерального резервного банка?» Если пользователь выбирает «Падает», то факт добавляется в систему и правило 5 может быть активировано. В противном случае система завершит работу без выдачи какой-либо рекомендации, так как в ее памяти недостаточно фактов для осуществления вывода.

Рассмотренный пример иллюстрирует работу типичной системы на основе прямой цепочки рассуждений:

1) система содержит описание ряда ситуаций;

2) для каждой ситуации система ищет в базе знаний правила, в условной части которых содержится соответствующее условие;

3) в соответствии с консеквентом (частью ТО) каждое правило может генерировать новые факты, которые добавляются к уже имеющимся в рабочей памяти;

4) система обрабатывает каждый вновь сгенерированный факт. При наличии хотя бы одного правила, в антецеденте (части ЕСЛИ) которого присутствует данный факт выполняются действия, начиная с пункта 2;

5) рассуждения заканчиваются, когда больше нет необработанных фактов и правил.

4.2.3. Обратная цепочка рассуждений на основе цели

Обратная цепочка рассуждений применяется в задачах, соответствующих процессу проверки гипотез при решении проблем человеком, — для заданной ситуации необходимо определить условия к ней приводящие. Обратная цепочка применима также, если требуется объяснить, как было получено решение. Процесс поиска от цели к данным рекомендован в следующих случаях.

- Цель поиска явно присутствует в постановке задачи или может быть легко сформулирована. Многие диагностические системы рассматривают возможные диагнозы, систематически подтверждая или опровергая некоторые из них способом поиска от цели.
- Имеется слишком большое число правил, которые на основе исходных фактов продуцируют возрастающее число заключений или целей. Своевременный отбор целей позволяет отсеять множество тупиковых ветвей, что сокращает пространство поиска.
- Исходные данные не приводятся в задаче, но подразумевается, что они должны быть известны или могут быть легко получены. Системы медицинской диагностики предлагают множество симптомов или результатов лабораторных анализов, подтверждающих или опровергающих диагноз.

Таким образом, при поиске от цели подходящие правила или шаги решения в основном применяются для исключения неперспективных ветвей поиска. Алгоритм обратного вывода обычно основан на *стратегии поиска в глубину*: в рабочую

память помещаются целевые факты, и далее ее содержимое последовательно сравнивается с заключением каждого правила в БЗ. Если содержимое рабочей памяти соответствует какому-либо правилу, то факты, указанные в условной части данного правила, помещаются в рабочую память и становятся новыми подцелями на следующей итерации работы продукционной системы. Система работает до тех пор, пока все подцели в рабочей памяти не будут подтверждены продукционными правилами. В этом случае исходная гипотеза будет истинной. В противном случае — ложной.

Пример 4.3. Обратная цепочка рассуждений

В качестве примера рассмотрим консультирующую экспертную систему для поддержки принятия решений директором некоторого предприятия при приеме нового сотрудника на работу (пример взят из книги [Levine, Drang, Edelson, 1988]). Поскольку в задаче необходимо выбрать один из нескольких возможных вариантов (должностей), для ее решения можно воспользоваться обратной цепочкой рассуждений. В действительности ответ уже существует. Перед директором сидит человек и всеми силами старается произвести на него хорошее впечатление. Если директора этот человек устраивает, для него нужно подобрать подходящую должность. Директору необходимо задать посетителю такие вопросы, ответы на которые дадут возможность сделать правильный выбор. Для описания подобных задач обычно используются диаграммы, которые называются деревьями решений. Деревья решений предоставляют необходимую наглядность и позволяют проследить ход рассуждений.

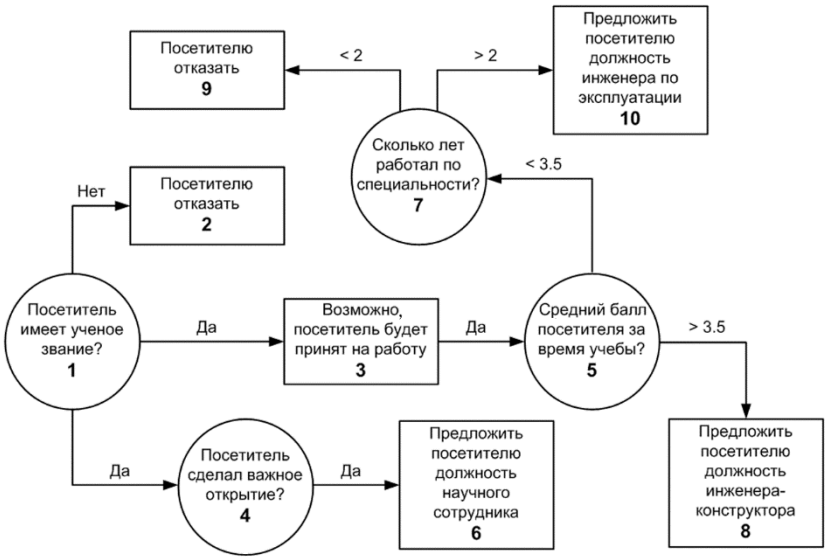


Рис. 4.15

Дерево решений для обратного вывода

Каждой вершине присваивается номер, по которым можно на них ссылаться. Вершины в форме окружностей, содержащие вопросы, называются вершинами решений. Прямоугольные вершины содержат цели диаграммы и соответствуют логическим выводам. Линии показывают направление рассуждений. Выбор ветви, выходящей из вершины, определяется проверкой условия, содержащегося

в вершине. В дереве решений обеим частям правила соответствуют связанные между собой вершина решения и вершина логического вывода. Условная часть содержит все вершины решения, находящиеся на пути к логическому выводу, т. е. каждая вершина решения на пути к выводу — это одно условие части ЕСЛИ. Вывод же составляет часть правила ТО. Таким образом, процесс формирования правил для всех возможных логических выводов состоит из следующих шагов:

1. Выбрать из дерева решений вершину вывода и зафиксировать ее.

2. Найти вершину решения, расположенную слева от выбранной вершины вывода и связанную с ней ветвью, и зафиксировать ее.

3. Повторять шаг 2 до тех пор, пока не будут рассмотрены все вершины решения, расположенные левее зафиксированной вершины вывода или не встретится новая вершина вывода. Если встретилась вершина вывода, то ее надо зафиксировать и прекратить выполнение шага 2. Выполнение также прекращается, если исчерпаны все вершины.

4. Каждая вершина решения, составляющая путь, — это одна из переменных части ЕСЛИ правила. Значение, связанное с ветвью, представляет собой условие. Переменные условной части правила объединяются логическим оператором И.

5. Выбранный логический вывод перенести в часть ТО правила.

Пример пути на дереве решений, из которого можно составить правило $1 \to 4 \to 6$:



Рис. 4.16

Пример пути на дереве решений

Для данного примера можно сформулировать следующие продукционные правила.

Путь $1 \to 2$: ЕСЛИ *Посетитель* НЕ *имеет ученое звание*, ТО *Посетителю отказать*.

Путь $1 \to 3$: ЕСЛИ *Посетитель имеет ученое звание*, ТО *Возможно, что посетитель будет принят на работу*.

Путь $1 \to 4 \to 6$: ЕСЛИ *Посетитель имеет ученое звание* И *Посетитель сделал важное открытие*, ТО *Предложить посетителю должность научного сотрудника*.

Путь $3 \to 5 \to 7 \to 10$: ЕСЛИ *Возможно, что посетитель будет принят на работу* И *Средний балл посетителя за время учебы* $< 3{,}5$ И *Опыт работы посетителя по специальности* > 2 лет, ТО *Предложить посетителю должность инженера по эксплуатации*.

Путь $3 \to 5 \to 7 \to 9$: ЕСЛИ *Возможно, что посетитель будет принят на работу* И *Средний балл посетителя за время учебы* $< 3{,}5$ И *Опыт работы посетителя по специальности* < 2 лет, ТО *Посетителю отказать*.

Путь $3 \to 5 \to 8$: ЕСЛИ *Возможно, что посетитель будет принят на работу* И *Средний балл посетителя за время учебы* $> 3{,}5$, ТО *Предложить посетителю должность инженера-конструктора*.

Рассмотрим работу обратной цепочки рассуждений на примере подтверждения или опровержения запроса пользователя. Предположим, что претендент на

работу хочет получить должность инженера-конструктора. В этом случае в рабочей памяти окажется следующая запись: «*Предложить посетителю должность инженера-конструктора*» (рис. 4.17).

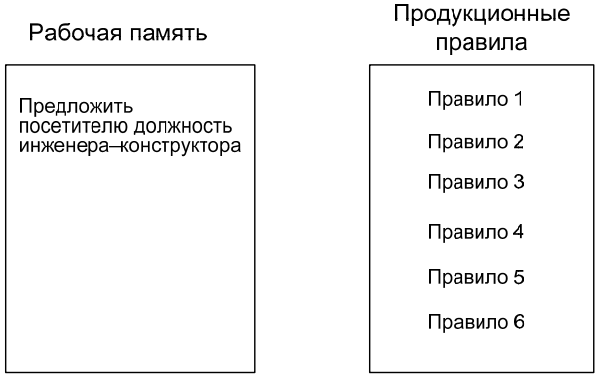


Рис. 4.17

Первый шаг обратного вывода

Правило 6 содержит в заключении данный факт, следовательно, система перейдет к рассмотрению новых подцелей (рис. 4.18):

Возможно, что посетитель будет принят на работу,
Средний балл посетителя за время учебы.

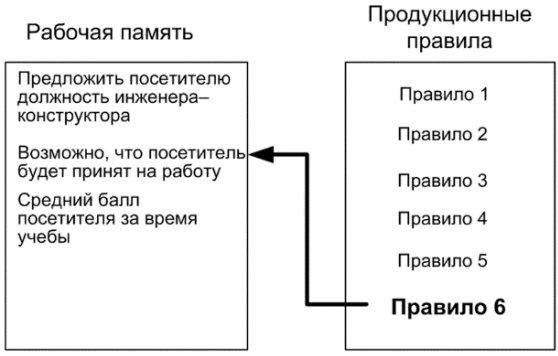


Рис. 4.18

Второй шаг обратного вывода

Второй факт может быть запрошен у пользователя, а первый факт необходимо попытаться вывести из БЗ. В заключении правила 2 указан этот факт, следовательно, следующей подцелью будет факт из условной части правила 2 (рис. 4.19):

Посетитель имеет ученое звание.

Факт, соответствующий новой подцели, можно выяснить у пользователя. Таким образом, для подтверждения или опровержения поставленной перед системой цели пользователю необходимо будет ответить на следующие вопросы:

Какой средний балл получен посетителем за время учебы?

Имеет ли посетитель ученое звание?

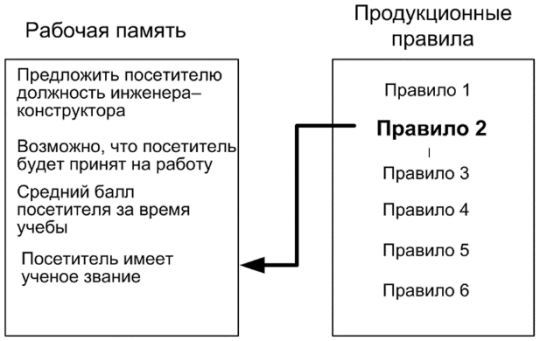


Рис. 4.19

Третий шаг обратного вывода

Если пользователь введет ответы, например, 4,5 и ДА, соответственно, то заданная цель будет подтверждена. А если ответы будут, например, 3,2 и НЕТ, то ответ системы на запрос пользователя будет отрицательный.

Данный пример демонстрирует простой, но очень типичный процесс, соответствующий обратной цепочке рассуждений. Этот процесс предусматривает следующие шаги.

1. Определить цель для логического вывода и выбрать ее в качестве текущей подцели.

2. В списке правил найти первое вхождение этой подцели. Если правило найдено, перейти к рассмотрению условной части найденного правила. Если правило не найдено, сообщить пользователю, что ответ найти невозможно.

3. Выбрать в качестве подцелей факты из условия из данного правила.

4. Если в списке подцелей имеются факты, истинность или значение которых могут быть запрошены у пользователя, то задать пользователю соответствующие вопросы.

5. Если очередная подцель может быть выведена, то перейти к шагу 2.

6. Если очередная подцель не может быть выведена или запрошена у пользователя, сообщить, что ответ получить невозможно.

7. Если все подцели подтверждены, то сообщить пользователю окончательный вывод.

4.2.4. Эвристические знания и метаправила

Разделение базы знаний и механизма вывода является сильной стороной экспертных систем. В процессе вывода решения все правила системы равнозначны и самодостаточны, т. е. все необходимое для активизации правила содержится в его условии, и одни правила не могут непосредственно вызывать другие. Работа машины вывода не зависит от предметной области, что делает ее универсальной. Но иногда для получения решения требуется вмешательство в стандартный процесс вывода. Для этих целей некоторые продукционные системы позволяют вводить в базу знаний специальные правила для управления процессом вывода — *метаправила*. Метаправила не принимают непосредственного участия в процессе формирования рассуждений, а определяют приоритет выполнения или исключают из рассмотрения обычных правила и выполняются в первую очередь. Таким образом, в базе знаний вводится определенное структурирование и упорядочивание правил. Например, предпосылки правила можно упорядочить таким

образом, что наиболее вероятные отрицательные ответы или простые подтверждения истины будут проверяться первыми. Ниже приведен пример подобного метаправила, сокращающего цепочку вывода:

МП1: *ЕСЛИ* кредитный рейтинг клиента высокий
И клиент является клиентом банка,
ТО сначала применить правила для льготных условий предоставления кредита.

Можно также сформулировать пример метаправила, касающегося общей стратегии вывода и не связанного с какой-либо конкретной предметной областью:

МП2: *ЕСЛИ* существуют правила, в условиях которых не упоминается текущая цель
И существуют правила, в условиях которых упоминается текущая цель,
ТО сначала следует активизировать первые из перечисленных правил.

Последний пример не связан с предметной областью, и поэтому подобные метаправила можно применять в системах различного назначения. Интерес к подобного рода обобщенным формулировкам знаний достаточно высок. Идея использования метаправил является весьма продуктивной, но, тем не менее, метаправилами следует пользоваться осмотрительно, учитывая возможные исключительные ситуации. Планирование порядка правил, определение стоимости отдельных проверок и т. п. — действия, эвристичные по своей природе. Эти подходы, с одной стороны, отражают действия человека-эксперта при решении проблемы, но, с другой стороны, могут давать и ошибочные результаты.

Эвристики в продукционных системах можно условно разделить на метаправила высокого уровня, такие как рефракция и оппортунистический поиск, и учет частных особенностей предметной области, таких как, например, в метаправиле МП1. Существуют алгоритмы оптимизации поиска потенциально полезных правил, такие как RETE [Forgy, 1982].

Rete — эффективный алгоритм сопоставления с образцом для продукционных систем, экспертных систем и баз знаний, созданный Чарльзом Форги из Университета Карнеги Меллона. Впервые был описан в рабочем документе 1974 г., затем в докторской диссертации 1979. Rete стал основой многих популярных экспертных систем, включая CLIPS, Jess, Drools, BizTalk Rules Engine и Soar (рис. 4.20).

Алгоритм Rete содержит обобщение логики функционала, ответственного за связь фактов и продукций в системах сопоставления с образцом. Алгоритм Rete имеет следующие характеристики:

- уменьшает или исключает избыточность условий за счет объединения узлов;
- сохраняет частичные соответствия между фактами при слиянии разных типов фактов. Это позволяет избежать полного вычисления (re-evaluation) всех фактов при любом изменении в рабочей памяти продукционной системы. Система работает только с самими изменениями (deltas);
- позволяет эффективно высвобождать память при удалении фактов;
- rete — это направленный ациклический граф из правил высшего порядка.

Обычно это сеть объектов в оперативной памяти, устанавливливающая связь условий правил и фактов (реляционных кортежей). Сеть Rete действует как процессор реляционных запросов, выполняя проекции, выборки и мультиплицирование небольшого числа записей.

По существу, алгоритм Rete интегрирует правила в сетевую структуру, позволяющую системе сопоставлять правила с данными, непосредственно указывающими на правило. Этот алгоритм значительно ускоряет выполнение поиска, особенно, если число правил велико.

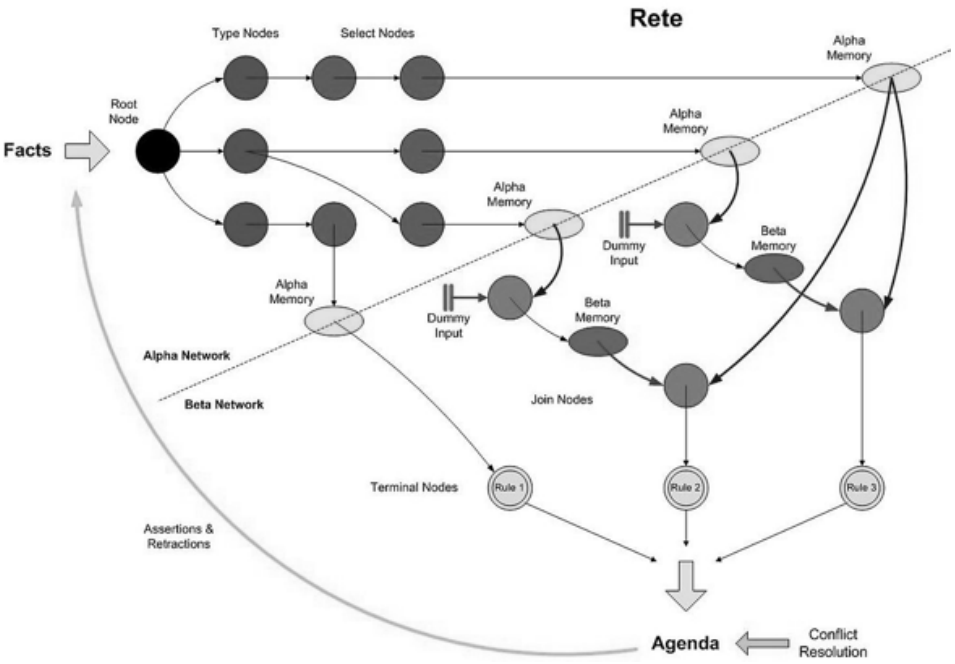


Рис. 4.20

Иллюстрация работы алгоритма Rete

Продукционные правила обеспечивают естественный способ описания процессов, управляемый сложной и изменяющейся средой. Через правила можно описать ход решения задачи, не имея заранее алгоритма этого решения. Более того, можно корректировать способ решения, добавляя новые правила, не изменяя существующих, что обеспечивает высокую модульность базы знаний. Однако, несмотря на то, что с помощью продукционных правил можно представить решение любой задачи, при большом количестве правил становится сложно отслеживать непротиворечивость базы знаний.

4.3. СЕМАНТИЧЕСКИЕ СЕТИ

Как следует из названия *семантическая сеть*, этот метод представления знаний позволяет описывать объекты, явления и понятия предметной области с помощью сетевых структур, основанных на теории графов (*семантика* — это наука, устанавливающая отношения между символами и объектами, которые они обозначают, или наука, определяющая смысл знаков; *сеть* — разновидность графа). Семантические сети являются машинным языком представления знаний и обладают определенной грамматикой. Это делает возможным трансляцию семантических сетей во внутренние структуры данных в памяти компьютера, на базе которых выполняется вывод знаний. В данном параграфе рассмотрены некоторые разновидности семантических сетей, начиная с первых графических языков. Данное изложение помогает понять содержание процесса формализации и вывода знаний с использованием графических ЯПЗ.

Одной из первых работ, оказавшей прямое влияние на современные семантические сети, считается система *экзистенциальных графов* (existential graph),

предложенная Чарльзом Пирсом (Charles Sanders Peirce) в 1909 г. [Roberts, 1973]. Эта система легла в основу *графической логики*, которую он называл «логикой будущего» (рис. 4.21).

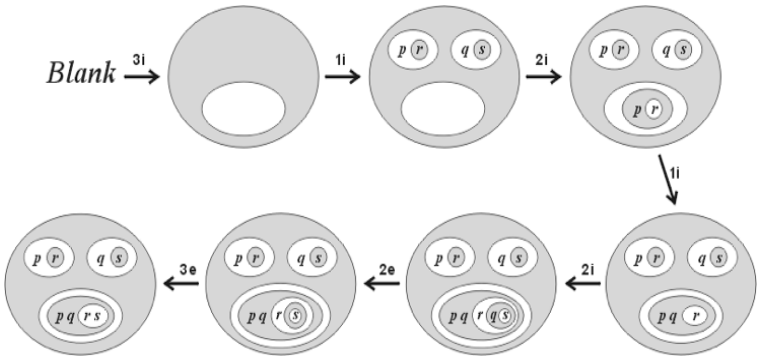


Рис. 4.21

Графическое доказательство теоремы Лейбница

Существовали и более ранние попытки использования сетевого представления. Так, еще в 1879 г. систему графических обозначений для логических выражений разработал Готлоб Фреге (Friedrich Ludwig Gottlob Frege). Несмотря на громоздкую, с современной точки зрения, нотацию, эти системы были весьма наглядны и эффективно использовались в теоретических разработках.

Достаточно давно графы используются в психологии для описания структуры понятий и ассоциаций, а также в области понимания естественных языков. Одна из первых работ в этом направлении была выполнена Отто Зельцем (Otto Selz) и опубликована в 1913 и 1922 гг. [Selz, 1913, 1922]. В качестве альтернативы распространенным в то время *ненаправленным ассоционистским теориям* (undirected associationist theories) он предложил теорию *схематического упреждения* (schematic anticipation) в качестве целенаправленного метода фиксации мыслительного процесса при поиске ассоциаций и обобщенных понятий. Идеи метода схематического упреждения Зельца оказали значительное влияние на работы по сетям и фреймам в ИИ. Пионеры ИИ Герберт Саймон (Herbert Simon) и Аллан Ньюэлл (Allen Newell) изучали работу Адриана де Гроота (Adriaan de Groot) [Simon, 1981] по применению теории Зельца для анализа вербальных описаний игры в шахматы начинающими игроками и гроссмейстерами. Впоследствии Ньюэлл и Саймон адаптировали метод Зельца для изучения процесса решения проблем человеком. Их ученик Росс Квиллиан (Ross Quillian) использовал комбинацию сетей Зельца и семантических сетей для построения системы машинного перевода.

Различные по структуре варианты семантических сетей объединяет принцип представления в сетевой форме понятий, объектов, категорий или типов и отношения между ними. Работы Шенка и Теслера [Schank, Tesler 1969] — «графы концептуальной зависимости», Брахмана [Brachman, 1979] — «сети структурного наследования» внесли значительный вклад в совершенствование семантических сетей, что сделало данную парадигму одним из самых распространенных способов представления знаний. Развитие идей сетевого представления привело к созданию языка концептуальных графов — формализма, позволяющего осуществлять представление смысла в задачах понимания естественного языка.

Одна из первых программных реализаций семантических сетей была выполнена бывшей студенткой Витгенштейна Маргарет Маштерман (Margaret Masterman) в 1961 г. при создании системы машинного перевода [Masterman, 1961]. Она определила структуру из 100 примитивных концептов (типов понятий), на базе которых был сформирован словарь из 15 000 понятий. Этот подход был использован в работе [Wilks, 1972] для построения системы понимания естественных языков. Хьючинс [Hutchins, 1986] показал, что подход, предложенный Маштерман, был наиболее эффективной реализацией семантических сетей для своего времени. Еще одним из пионеров в использовании семантических сетей является Силвио Цекатто (Silvio Ceccato), также занимавшийся созданием системы машинного перевода. Для построения *корреляционной сети* (*correlational nets*) он использовал 56 типов отношений, включая отношения тип–подтип, класс–экземпляр, часть–целое, логические, численные и ряд других [Ceccato, 1961]. На этапе разбора Цекатто использовал деревья зависимостей (dependency trees), «коррелирующие» с предопределенными сетями, для разрешения неоднозначностей. На этапе генерации использовались сети, управляющие выбором слов. Наиболее значительные из ранних работ представлены в сборнике под редакцией Минского [Minsky, 1968], включая классические статьи МакКарти (McCarthy), Квиллиана (Quillian), Боброу (Bobrow), Рафаэля (Raphael) и других.

Интересным примером применения парадигмы семантических сетей является система анализа естественного языка ConceptNet. Демонстрационную версию этой программы можно скачать с сайта http://conceptnet5.media.mit.edu. Другим примером использования семантических сетей является программа WordNet, предназначенная для определения отношений между понятиями. Программа доступна на сайте http://wordnet.princeton.edu.

4.3.1. Когнитивные модели ассоциативной памяти и представление знаний

Существенное влияние на развитие исследований в области когнитивной психологии и применения семантических сетей оказали работы Квиллиана, предположившего, что в основе восприятия текста человеком лежит «создание некоторого рода мысленного символического представления» [Quillian, 1968]. Он впервые предложил использовать для моделирования человеческой памяти сетевые структуры, в которых узлы соответствуют *словесным понятиям*, а связи между узлами — отношениям между понятиями. Квиллиан также ввел разделение узлов на *узлы-типы* и *узлы-лексемы*. Каждый узел-тип соответствует какому-либо понятию и связан с определенной комбинацией узлов-лексем, являющихся определением данного понятия. В свою очередь, смысл узла-лексемы определяется через ссылку на соответствующие узлы-типы. Такая организация памяти напоминает толковый словарь, где каждый элемент определяется другими понятиями, определенными в этом же словаре.

Пример 4.4. Семантическая сеть для понятия «коса» (рис. 4.22)

Существует три различных определения понятия «коса»: коса 1 — прическа, коса 2 — орудие и коса 3 — часть побережья. На рисунке 4.22 изображены три плоскости, в которых представлены сети для определения различных значений слова «коса»:

- коса 1 — женская прическа, состоящая из заплетенных волос;
- коса 2 — сельскохозяйственное орудие, используемое людьми для срезания травы;
- коса 3 — небольшой участок побережья, уходящий в море или реку.

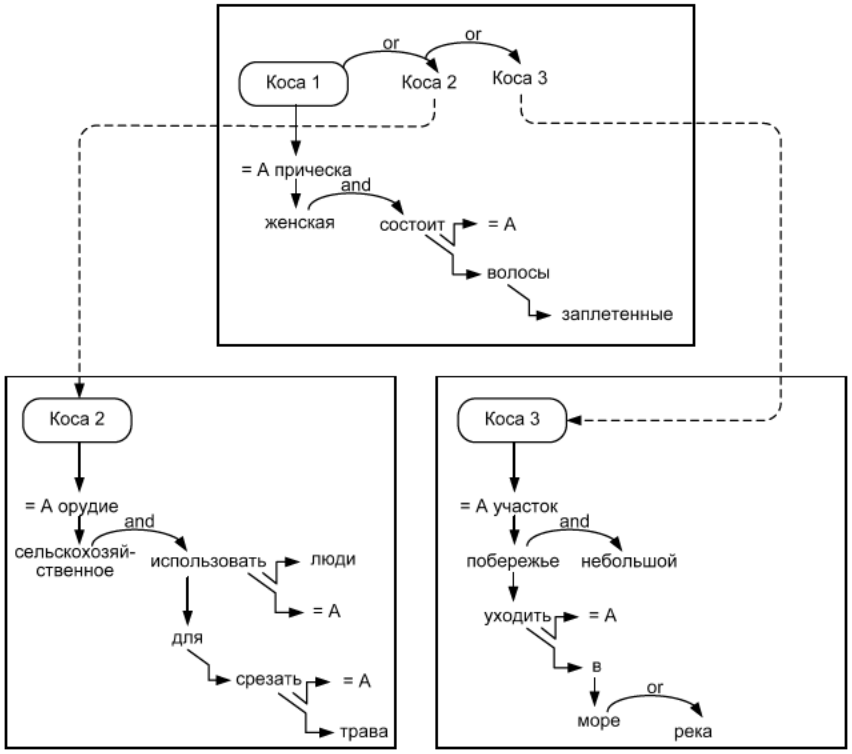


Рис. 4.22

Пример семантической сети для понятия «коса»

Для определения понятий в сети используются переменные, позволяющие формировать выражения, например: А = прическа, состоит (А, волосы).

Еще одним важным аспектом модели памяти Квиллиана является передача свойств от определяющих понятий или типов к определяемому понятию. Так, например, определим термин «машина» как конструкцию, состоящую из связанных компонентов, выполняющих некоторую работу. Это определение требует связать тип «машина» с лексемами «конструкция» и «компонент». Если теперь определить тип «компьютер» как разновидность «машины», то можно сказать, что компьютер является конструкцией из компонентов, выполняющих определенную работу. Таким образом, можно сэкономить память, поскольку нет смысла хранить информацию, связанную с термином «машина» в определении понятия «компьютер». Данный аспект получил название *когнитивная экономия*. В настоящее время этот механизм передачи свойств принято называть *наследованием свойств* (inheritance). Наследование получило широкое распространение в различных моделях представлении памяти.

В конце 1960-х гг. идеи Квиллиана были реализованы в компьютерной программе, которая характеризовала английские слова в терминах других английских слов. Программа осуществляла поиск в базе знаний отношений между парами слов, пытаясь определить общее определяющее понятие или узел пересечения. Пути к пересечениям представляли отношения между словесными понятиями. Так, например, эта программа смогла отыскать пересечение понятий «плач»

и «комфорт» и заключила следующее: «Плач (Cry 2) вызывает производство печальных звуков. Покой (Comfort 3) позволяет уменьшить печаль» [Quillan, 1967]. Числа обозначают номера значений слов, выбранные программой. На рисунке 4.23 показана точка пересечения данных путей.

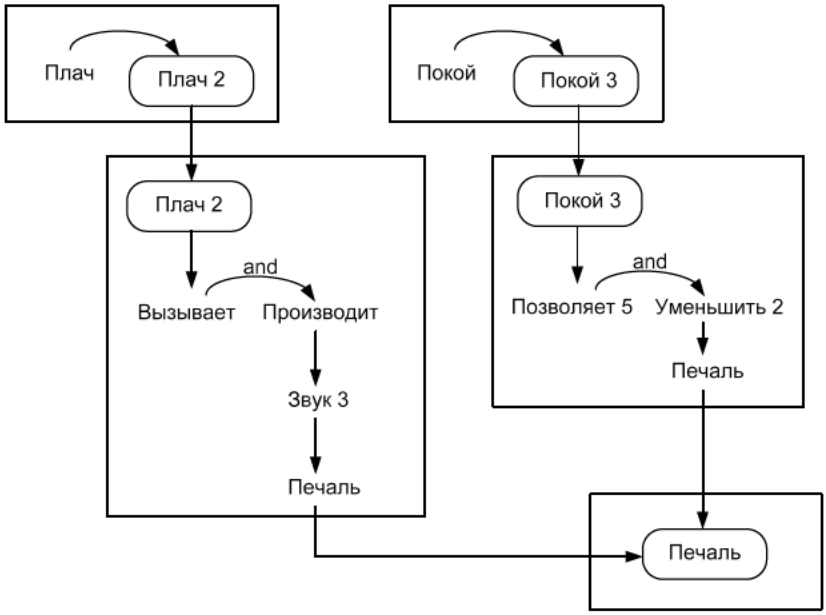


Рис. 4.23

Пересечение путей, ведущих от понятия «Плач» (Cry) к понятию «Покой» (Comfort) [Quillan, 1967]

Модели памяти, в которых значения объекта определяются через связи с другими объектами, развиваются в *ассоционистских теориях* (associationist theory). В частности, с точки зрения *ассоцианизма*, при формировании понятий связь между событиями устанавливается, если многократно происходит совместное наступление этих событий. На языке психологии этот процесс называется *стимул-реакция*. Таким образом, согласно принципу ассоциации, усвоение понятия есть результат:

1) подкрепления правильного сочетания стимула с реакцией;

2) неподкрепления «неправильного» сочетания стимула с реакцией опознания его как некоторого понятия.

Современная психология также предполагает наличие внутренней структуры, своего рода «метауровня», отвечающей за отбор, организацию и преобразование информации [Солсо, 2006]. В одном из экспериментов, проведенной группой Гордона Бауэра (Gordon H. Bower) [Anderson, Bower, 1979], было показано, как организация семантических единиц (т. е. структура сети понятий) влияет на память и воспроизведение при составлении различных понятийных иерархий. В ходе этих и других экспериментов была подтверждена гипотеза о том, что интеллектуальные функции человека подразумевают существование ассоциативной сети, в которой одни понятия соединяются с другими. Следовательно, наши знания происходят из этой сети ассоциаций.

Коллинзом и Квиллианом были исследованы вопросы хранения и скорости извлечения информации из памяти человека [Quillian, 1968]. В своем эксперименте они использовали семантическую сеть, в которой были представлены знания о канарейке (рис. 4.24). Например, «канарейка — это желтая птица, которая умеет петь», «птицы имеют перья и крылья, умеют летать» и т. п. Знания в этой сети представлены на различных уровнях: на нижнем уровне располагаются более частные знания, а на верхних — более общие. При таком подходе для понимания высказывания «Канарейка может летать» необходимо воспроизвести информацию о том, что канарейка — это член множества птиц, и у птиц есть общее свойство «летать». Согласно принципу когнитивной экономии, это свойство распространяется (наследуется) и на канареек.

Лабораторные эксперименты показали, что реакции людей на простые вопросы типа «Канарейка — это птица?», «Канарейка может летать?» или «Канарейка может петь?» различаются по времени. Ответ на вопрос «Может ли канарейка летать?» требует большего времени, чем на вопрос «Может ли канарейка петь». По мнению Коллинза и Квиллиана, это связано с тем, что информация запоминается человеком на наиболее абстрактном уровне. Вместо того чтобы запоминать все свойства каждой птицы, люди запоминают только отличительные особенности, например, такие как желтый цвет и умение петь у канареек, а все остальные свойства переносятся на более абстрактные уровни: канарейка как птица умеет летать и покрыта перьями; птицы, будучи животными, дышат и питаются и т. д. Действительно, ответ на вопрос «Может ли канарейка дышать?» требует большего времени, так как человеку необходимо проследовать по иерархии понятий в своей памяти. С другой стороны, конкретные свойства могут перекрывать более общие, что также требует меньшего времени на обработку информации. Например, вопрос «Может ли страус летать» требует меньшего времени для ответа, чем вопросы «Имеет ли страус крылья?» или «Может ли страус дышать?»

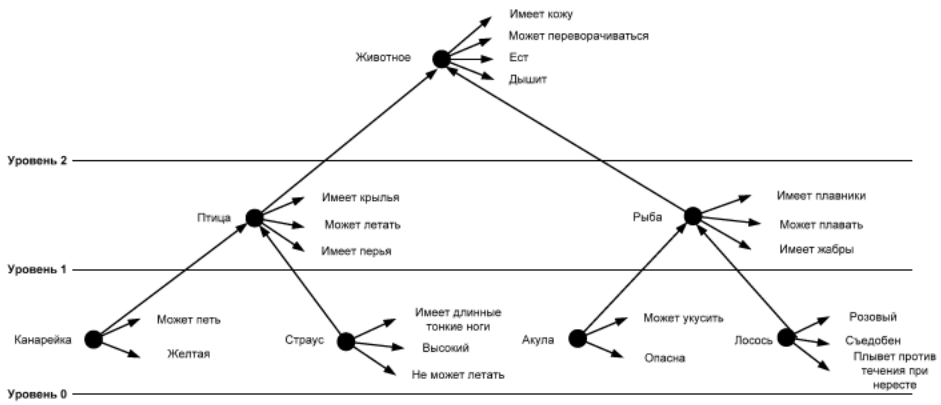


Рис. 4.24

Семантическая сеть для исследований хранения и воспроизведения информации в памяти человека

Идеи ассоциативного устройства памяти получили развитие и в *теории распространения активации* [Anderson, 1983], разработанной Коллинзом и Элизабет Лофтус (Loftus). Эта теория предполагает, что в сети ассоциаций связи имеют

различную «длину»: более короткие соответствуют более прочной связи между понятиями, а более длинные — менее сильной связи. Согласно данной модели, между понятиями распространяется *активация*, что может объяснить результаты экспериментов с использованием предварительной подготовки, в результате которой понятие становится более доступным после предъявления связанного с ним *подготавливающего стимула* или какого-либо другого слова. Например, при предъявлении зеленого цвета человек быстрее опознает слово «зеленый». Более того, при предъявлении зеленого цвета, опознавание таких слов, как «трава» или более отдаленной ассоциации — «лужайка», происходит быстрее, чем при отсутствии подготавливающего стимула.

Представление знаний в виде сетевых структур имеет глубокое психологическое обоснование и, по всей видимости, соответствует репрезентации знаний в памяти человека. Под термином «семантическая сеть» скрывается целое семейство подходов к представлению знаний, основанных на использовании сетей. Различия между подходами касаются, главным образом, принципов именования узлов, связей и механизмами вывода. Но общее множество понятий и отношений присуще всем сетевым языкам представления. Более детально структура и основные механизмы семантических сетей рассмотрены ниже.

4.3.2. Связь теории графов и семантических сетей

В ранних работах Квиллиана были сформулированы идеи использования формализма *теории графов* (graph theory) для построения семантических сетей. Терминология теории графов позволяет описывать практически любые структуры абстрактных данных. Однако термины теории графов получили в теории искусственного интеллекта несколько отличные определения, некоторые из которых приведены ниже [Джексон, 2001]. Для построения графа используются два основных вида примитивов: узлы и связи, которые могут быть дополнительно промаркированы. Связи могут быть направленными или ненаправленными. Узлы являются начальными и конечными точками связей. Узлы принято называть *вершинами* (*vertex*), а связи — *ребрами*, или *дугами* (*arc*).

С формальной точки зрения графом называется множество пар вершин:

> Пусть **N** — множество вершин, тогда любое подмножество **N × N** является обобщенным графом.

Каждая вершина может иметь любое количество дуг, но все дуги начинаются и завершаются на какой-либо вершине. Граф не обязательно должен быть связным, т. е. он может состоять из фрагментов. Пример обыкновенного, *несвязного (unconnected)* и *ненаправленного* (*undirected)* графа с маркированными вершинами приведен на рисунке 4.25.

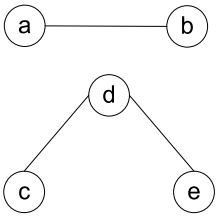


Рис. 4.25
Обыкновенный граф

В *обыкновенном* графе каждой дуге соответствует пара различных вершин, т. е. *петли (loop)* не допускаются. Однако это не является ограничением. Также возможно, что несколько дуг образуют «замкнутую цепь» — *цикл* (*cycle*). Если граф является единым фрагментом, то такой граф называют *связным* (*connected graph*). Пример связного графа с петлей и циклом приведен на рисунке 4.26.

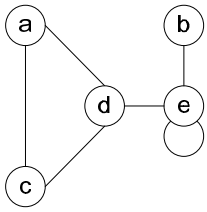


Рис. 4.26
Связный граф с петлей и циклом

Если на графе не допускаются ни петли, ни циклы, то такой граф называют *лесом* (*forest*). Ели для дуг графа явно определено, какая из вершин является начальной, а какая конечной, то такой граф будет *ориентированным* (*directed graph*).

Пусть граф содержит **N** вершин, если в парах подмножества **N×N** имеет значение порядок, то такой граф называется ориентированным.

Деревом (*tree*) будем называть связный лес. Одна из вершин дерева является его *корнем* (*root*), а остальные — образуют *ветви* (*branch*) дерева. Ветви начинаются в корне или являются продолжением других ветвей, образуя, таким образом, структуру «наследников» корневой вершины. Вершины, не имеющие наследников, называют *терминальными*, или *листьями* (*leaf*) дерева. Вершины, не являющиеся корнем или листьями, называются *промежуточными* или *нетерминальными* (*nonterminal vertex*).

Обыкновенный граф, состоящий из **N** вершин и **N – 1** дуг, в котором отсутствуют циклы, является деревом.

В теории графов *сетью* (*net*) принято называть *взвешенный* (*weighted*), *ориентированный* граф, т. е. граф, в котором дуги промаркированы некоторыми числами. Такой граф можно сопоставить с дорожной картой, на которой для каждой дороги обозначено расстояние между двумя населенными пунктами. Часто на графах весом дуги оценивается соответствующая «стоимость» пути. На рисунке 4.27 приведен пример простого, ориентированного дерева с маркированными и направленными связями, вершина *a* — корень дерева, *b*, *c*, *e* — листья, *d*, *f* — нетерминальные вершины.

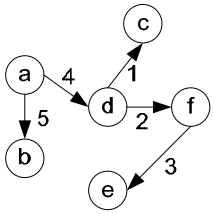


Рис. 4.27
Дерево

Двудольным графом *(bipartite graph) называется* граф, у которого существует такое разбиение множества вершин на две части (доли), что концы каждой дуги принадлежат разным долям. Если при этом любые две вершины, входящие в разные доли, являются *смежными* (*adjacent, joined*), т. е. они соединены ребром, то граф называется *полным двудольным*.

Однако в задачах ИИ веса дуг в сетях могут иметь более широкую трактовку и не обязательно являться числами. Дуги в этих сетях, как правило, является ориентированными, поскольку представляемые ими отношения не должны быть симметричными. Поэтому определение сети можно сформулировать следующим образом.

Пусть **L** — множество взвешенных дуг, **N** — множество вершин,
тогда сетью будем называть любое подмножество **N×L×N**,
в котором имеет значение порядок в триадах **NLN**.

Семантическая сеть представляет собой ориентированный граф, где вершины представляют понятия предметной области, а дуги — отношения между понятиями. Часто вершины в семантических сетях называют *узлами*, а дуги — *связями*. При построении семантической сети могут использоваться различные виды графов.

4.3.3. Предшественники современных семантических сетей

Среди первых работ по моделированию семантических сетей можно выделить:

- причинно-следственные сети Попла;
- деревья иерархических зависимостей и классификаций;
- падежи Филмора, рассмотренные в этом параграфе, а также ряд моделей, рассмотренных в параграфе 6.2;
- рематические графы;
- ассоциативные сети;
- дефинитивные сети;
- пропозициональные семантические сети;
- казуальные сети;
- концептуальные графы.

Для представления пространственно-временных отношений используются обыкновенные графы. Также их можно использовать для представления причинно-следственных связей. Так, на рисунке 4.28 приводится пример сети из работы [Pople, 1982], описывающей причинно-следственные связи, по которым заболевания могут быть объяснены в патофизиологических терминах. Примечательно, что в данном примере отображены только те причинно-следственные связи, которые непосредственно объясняют суть заболевания, а остальные несущественные связи отнесены к эллипсису понятия (т. е. опущены без искажения его смысла).

Для представления иерархических зависимостей и классификаций применяются деревья. Например, на рисунке 4.29 представлено дерево классификации заболеваний с этиологической точки зрения, т. е. на основе причины возникновения болезни. От корневого узла дерева «этиология» (etiology) наследуются наиболее общие категории «инфекция» (infection), «рак» (cancer), «нарушение иммунитета» (abnormal immunity) и т. п. Эти категории, в свою очередь, могут порождать более специфичную классификацию. Так, например, «инфекция» имеет такие подкатегории, как «болезнетворные микроорганизмы» (viral infections), «бактериальная инфекция» (bacterial infections) и «грибковая инфекция» (fungal infections).

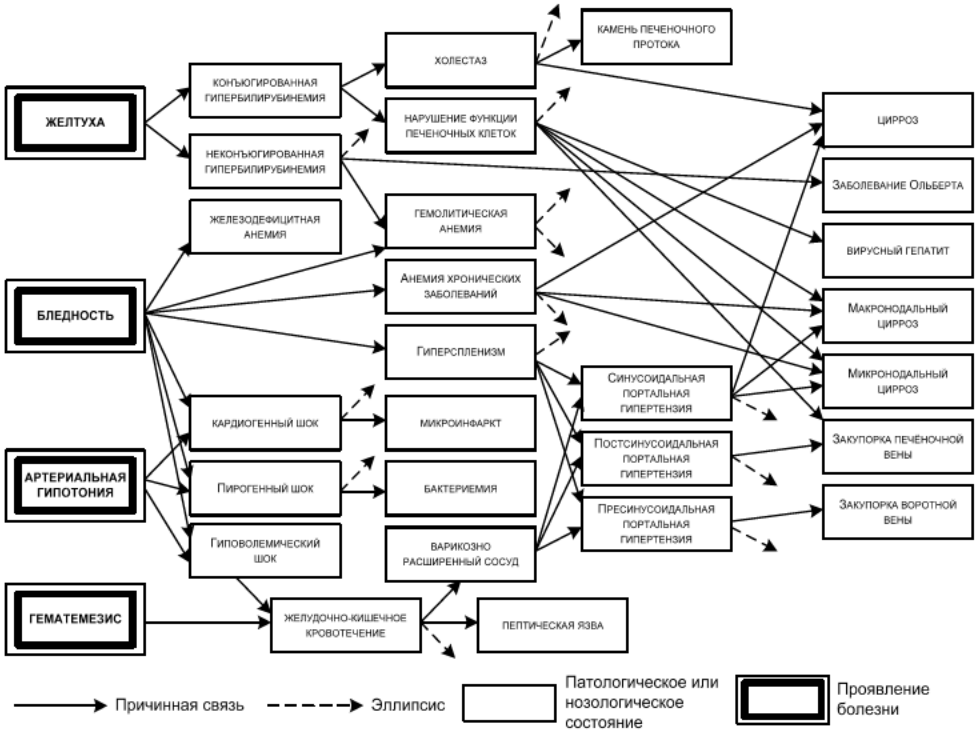


Рис. 4.28

Фрагмент сети, описывающей причинно-следственные связи

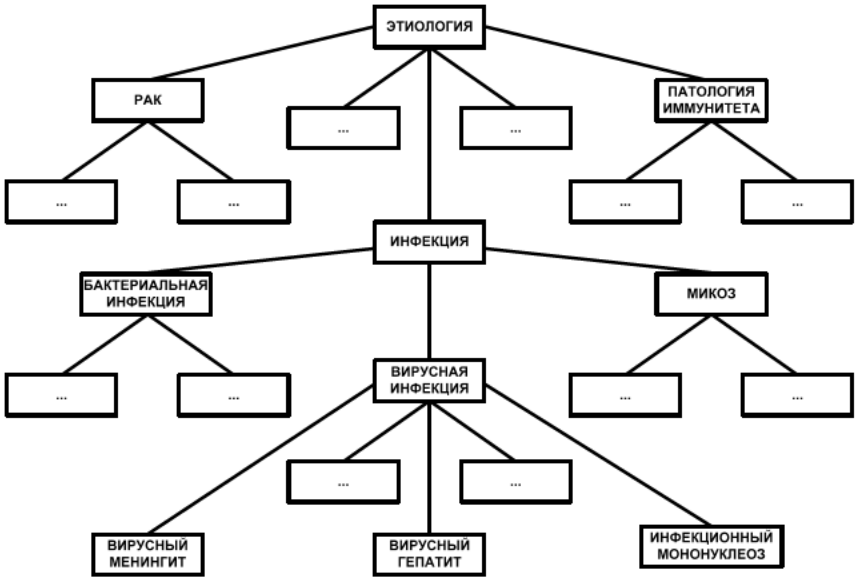


Рис. 4.29

Фрагмент дерева классификации заболеваний

Одной из главных проблем представления семантических отношений между понятиями является определение такого разнообразия связей, которое будет достаточно для описания различных семантических аспектов знаний данной предметной области. В то же время это разнообразие должно быть в значительной степени универсальным и не должно зависеть от специфики предметной области. Большинство работ по семантическим сетям были направлены именно на развитие сетевых отношений (связей) для более полного моделирования семантики естественного языка. Как отмечено в [Люгер, 2003], базы знаний обеспечивают общность и непротиворечивость за счет реализации базовых семантических отношений естественного языка как части формализма, а не как части знаний о предметной области. В работе [Brachman, 1979] отмечается, что «ключевым вопросом является выделение примитивов для языка семантических сетей. Примитивами сетевого языка являются те понятия, для понимания которых интерпретатор программируется заранее и которые не представлены в самом языке».

В работе [Fillmore, 1968] отношения определяются на основе грамматики английского языка. Связи соответствуют роли существительного или группы существительных, входящих в заданное предложение. К числу возможных ролей относятся *агент*, *объект*, *инструмент*, *время* и *место*. Предложение представляется набором узлов, один из которых соответствует глаголу, а другие узлы представляют остальные понятия в предложении. Такая структура получила название *падежи Филмора* (case frame). При разборе нового предложения ищется глагол и в базе знаний выбирается соответствующий ему падеж. Затем узлы связываются с значениями представленными в предложении. Например, предложение «Павел ударил по шайбе клюшкой» можно представить сетью, изображенной на рисунке 4.30.

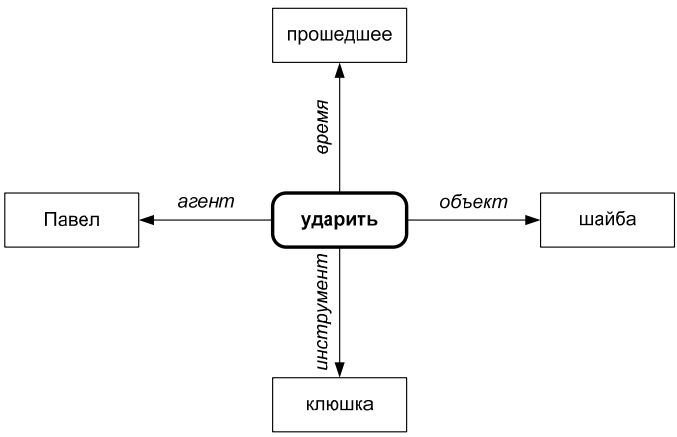


Рис. 4.30
Пример падежа Филмора

Связи, представленные сетью падежа, представляют отношения в естественном языке. Например, агентное отношение представляет связь между глаголом и его субъектом, а объектное отношение — между глаголом и объектом. Из отношений сети из примера следует, что Павел — это хоккеист, совершивший бросок шайбы, а клюшкой можно играть в шайбу. Данные знания не прописаны явно в предложении, но могут быть выведены из него при наличии соответствующих записей в базе знаний. Отличительной особенностью этого формализма является то, что отношения за-

поминаются отдельно от реального предложения и языка, на котором сформулировано предложение. Подобный подход был принят во многих сетевых языках.

4.3.4. Построение семантической сети

Итак, если узлы сети — понятия, то связи определяют отношения между ними.

Часто в качестве отношений семантической сети выступают связи следующих видов:

- связи, определяющие тип объектов («это есть» или «класс–подкласс», «иметь частью» или «часть–целое», «принадлежать» или «элемент–множество» и т. п.);
- функциональные связи (определяемые обычно глаголами «производит», «влияет» ...);
- количественные («больше», «меньше», «равно» ...);
- пространственные («далеко от», «близко от», «за», «под», «над» ...);
- временные («раньше», «позже», «в течение» ...);
- атрибутивные связи (иметь свойство, иметь значение...);
- логические связи («и», «или», «не») и др.

Характерной особенностью семантических сетей является обязательное наличие в одной сети трех типов отношений:

1) класс — элемент класса;

2) свойство — значение;

3) экземпляр — элемента класса.

Проблема поиска решения в базе знаний типа семантической сети сводится к задаче поиска фрагмента сети, соответствующего некоторой подсети, отвечающей поставленному вопросу. Подобного рода задачи решаются с помощью математического аппарата теории графов. Следует особо отметить роль фундаментальных признаков связей *рефлексивность*, *симметричность* и *транзитивность* в процессе вывода на сети. Так, например, связи вида «это есть» или «иметь частью» транзитивны, что позволяет говорить об установлении с помощью этой связи свойств иерархии наследования в сети. Это означает, что элементы более низкого уровня в сети могут наследовать свойства элементов более высокого уровня.

Примером простой семантической сети является описание объекта автомобиль и ряда связанных с ним понятий (см. рис. 4.31). На этой сети присутствует следующая цепочка понятий: «автомобиль имеет частью двигатель», «двигатель имеет частью стартер». В силу транзитивности отношения «иметь частью» можно вывести следующее утверждение «автомобиль имеет частью стартер». Аналогично можно сделать вполне очевидный вывод, что «Иванов обладает автомобилем» или что «Mercedes имеет частью двигатель и потребляет топливо».

Одним из очевидных ограничений семантических сетей является использование только бинарных отношений. Например, выражение «Иванов прилетит из Рима в Санкт-Петербург 28 сентября», которое может быть записано с помощью предиката *прилетит (Иванов, Рим, Санкт-Петербург, 28 сентября)* непосредственно не может быть реализовано в виде семантической сети. Однако представление n-арных отношений может быть легко реализовано с помощью введения соответствующей подсети для описываемой категории событий. Если определить подсеть для события «рейс 276: Рим — Санкт-Петербург, 28 сентября», то n-арное отношение «прилетит» можно заменить на бинарное «прилетит рейсом X». Сеть для данного примера приведена на рисунке 4.32.

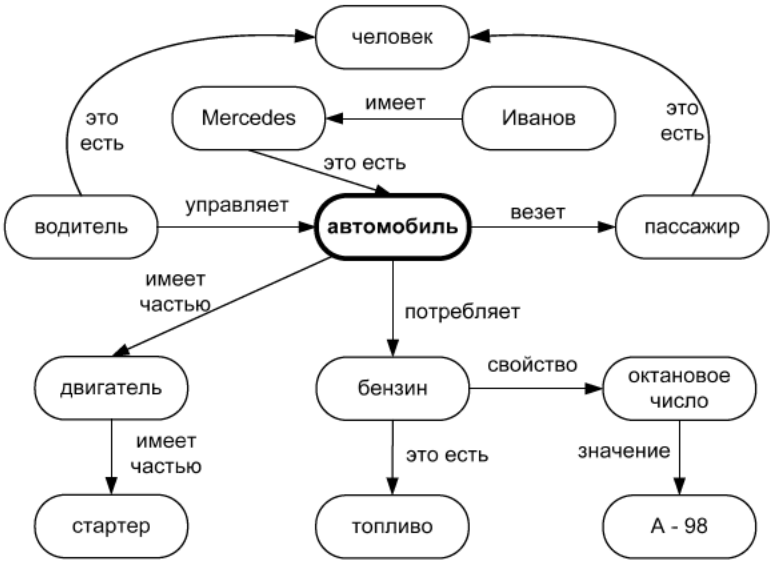


Рис. 4.31

Пример простейшей семантической сети

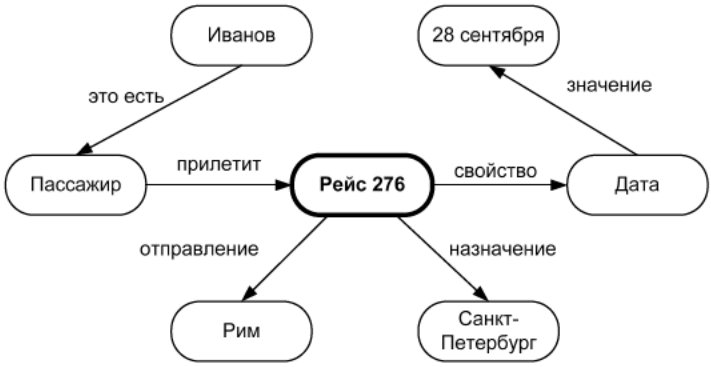


Рис. 4.32

Представление n-арных отношений на семантической сети

4.4. ПРИМЕРЫ МОДЕЛИРОВАНИЯ ВЫРАЖЕНИЙ ЕСТЕСТВЕННОГО ЯЗЫКА

Основная форма передачи знаний — это естественный язык. Автоматический анализ выражений на русском, английском или любом другом языке представляет собой сложную задачу. И хотя эта проблема выходит за рамки данной книги, рассмотрение некоторых примеров использования семантических сетей для моделирования выражений будет полезно для понимания принципов формирования и механизмов работы семантических сетей.

Выше были рассмотрены основные формализмы семантических сетей. Пример, приведенный на рисунке 4.31, иллюстрирует общепринятую форму представления знаний с использованием семантической сети. Однако эта форма не раскрывает ряд характерных для естественного языка аспектов. К ним относятся:

- актуальное членение предложения на исходную, изначально данную составляющую, называемую *темой*, и новую, утверждаемую говорящим составляющую, называемую *ремой*. Подобное разделение высказывания, определяет тема-рематические отношения внутри фразы и оформляется в соответствии с целью коммуникации [Распопов, 1961]. Грамматическая структура фразы зависит от задач целенаправленного сообщения. Таким образом, некорректное смысловое членение предложения (и формальное описание этого членения на ЯПЗ) делает невозможным его правильную интерпретацию;
- формализованное представление инструкций и действий. Решение любой проблемы связано с описанием последовательности действий, выполнение которых зависит от текущих значений свойств объектов предметной области. Изменение этой последовательности или неправильное описание свойств объектов может привести к некорректному решению;
- рассуждения на метауровне и представление широкого класса свойств семантики естественного языка, включая концептуальное описание предметной области. Множество высказываний, описывающих предметную область, составляют иерархическую структуру. Моделирование этой структуры позволяет применить логику для описания абстрактных рассуждений, с одной стороны, и для получения новых корректных выражений — с другой. Это делает возможным логический вывод знаний (рассуждения) на основе сетевых ЯПЗ.

Ниже рассмотрены некоторые примеры сетевых языков представления смысла. Особое внимание уделено формализму концептуальных графов Джона Сова, как одному из наиболее универсальных графических языков.

4.4.1. Применение сетевых моделей для понимания естественного языка

За первыми работами в области семантических сетей последовал ряд исследований по созданию сетевых языков для представления смысла выражений. Для всех языков общим является использование нотации из теории графов, а различие заключается в синтаксических правилах построения сети и особенностях механизмов вывода. Краткие описания некоторых из этих языков приведены в данном параграфе.

Одной из модификаций семантических сетей являются *ассоциативные сети*, предложенные в работах Г. С. Цейтина [Цейтин, 1985]. Его сети содержат узлы, соответствующие понятиям в тексте или объектам во внешнем мире, и направленные дуги. В качестве имени узла может выступать число, строка символов, исполняемая процедура или подсеть из других узлов. Имена всех дуг, исходящих из данного узла, должны различаться. Обычно имена дуг обозначают семантические и синтаксические роли узлов. Например, роли, обозначаемые словами «открыть» и «закрыть», могут соответствовать действиям с разными объектами: открыть/закрыть дверь, бутылку, компьютерный файл. В терминах ассоциативных сетей эти слова могут быть представлены соединяющими некоторые виды объектов дугами, с которыми связываются соответствующие действия.

Теория *рематических графов* [Parker-Rhodes, 1978; Lehman, 1992] применяет механизм сетевых представлений для отражения феноменов, характерных для

языка как средства коммуникации, а не в качестве механизма моделирования знаний о мире (онтологии). Эта теория основана на математической теории решеток (частично упорядоченных множеств) и обеспечивает представление одновременно семантической, синтаксической и фонетической информации. Построение рематических графов осуществляется в соответствии с *актуальным членением предложения*, описывающего функциональные компоненты предложения — ремы, или сообщаемой части, и темы, отправной точки сообщения. Например, в предложении «Студенты на первом курсе стараются ходить на все занятия» выделяется тема «студенты на первом курсе» и рема «стараются ходить на все занятия», т. е. о студентах первого курса сообщается, что они стараются ходить на все занятия. В некоторых теориях функционального членения предложения тема и рема называются, соответственно, *топиком* (topic) и *фокусом* (focus) или комментарием (comment).

Рематические графы, соответствующие предложениям «*Larry was reading some trash*» и «*Larry was reading a comic bought at the station*», изображены на рисунке 4.33*a* и *b*. Эти графы строятся из дерева синтаксического разбора предложений путем конвертирования его в ациклический граф. Процесс включает слияние конечных узлов, относящихся к одному и тому же объекту, и внесение упорядочения в соответствии с *фокусом внимания* (т. е. ремой): на рисунке прерывистые линии связывают *точку ссылки* (point of reference) и *точку интереса*.

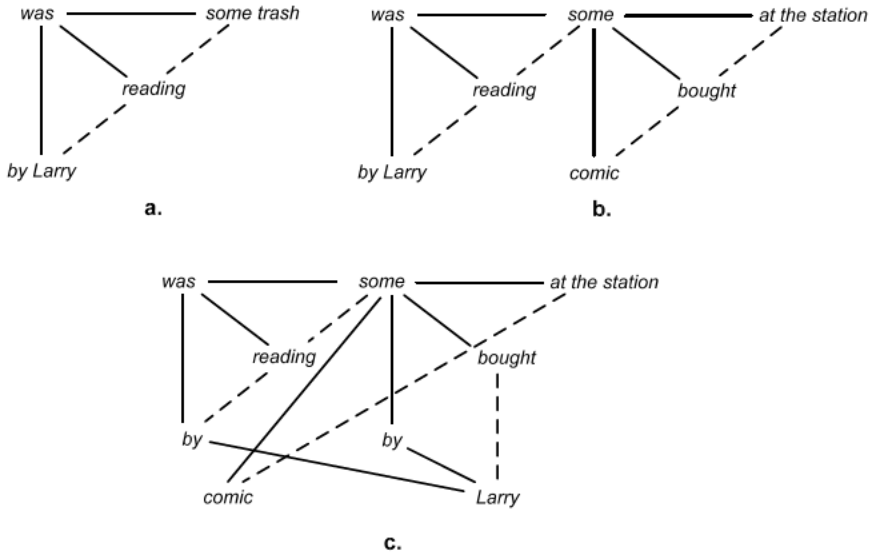


Рис. 4.33
Примеры рематических графов [Parker-Rhodes, 1978]

На основании рематических графов, объединяемых в единое представление при анализе целого текста (рис. 4.33*c*: анализ предложения «*Larry was reading a comic he had bought at the station*»), осуществляется логический вывод с помощью модуса (пассивно-активное отношение между действием и ситуацией), вида действия (первичное, вторичное, третичное) и решеток, содержащих классические отношения класс–подкласс и фазы (направление действия, включающее источник, цель и позицию).

В системе SNOOP (System with Networks and Object-Oriented Productions) [Шаров, 1995] реализован другой подход к использованию механизма семантических сетей для лингвистических задач, интегрируя сетевые представления с методами объектно-ориентированного программирования и продукционных правил. Узлы и отношения сети принадлежат классам, описываемым с помощью отдельной сети наследования, в которой для класса определяется набор возможных полей и свойств, ассоциированных с объектами, соответствующими этому классу. Свойства объектов и классов описываются с помощью групп продукционных правил, каждое из которых состоит из образца, позволяющего осуществлять навигацию по сети. Объектам также приписываются действия по изменению внутренней структуры узлов и всей сети, например, созданию и удалению узлов и отношений.

Реализацией так называемых *дефинитивных сетей* (Definitional Networks) явилась система KL-ONE (Knowledge Language One) [Brachman, 1979], идея которой была предложена в работе [Woods, 1975]. На рисунке 4.34 приведен пример сети в терминах KL-ONE, определяющей концепты «Грузовик» и «Автопоезд» как подтипы понятия «Транспорт». Узлы этой сети подразделяются на *общие концепты* (generic concepts) — белые овалы, представляющие типы, и *индивидуальные концепты* (individual concept) — серый овал (число 18), представляющий экземпляр. Овал, отмеченный символом «*», обозначает, что «Integer» (целое число) — это встроенный (built-in) или примитивный (primitive) тип.

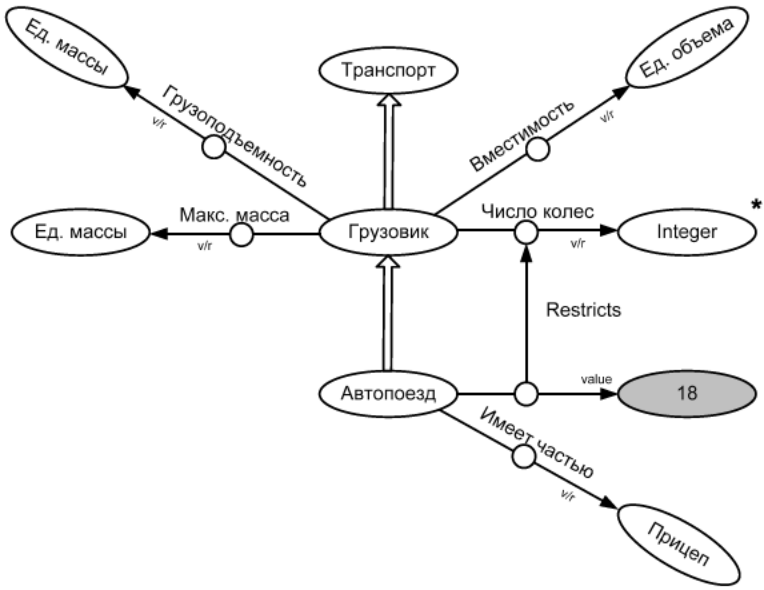


Рис. 4.34

Концепты «Грузовик» и «Автопоезд», определенные в KL-ONE

Связи также подразделяются на виды: дуга в виде двойной линии представляет отношение *подтип–супертип* (subtype–supertype link) между концептами «Автопоезд» и «Грузовик», а также между «Грузовик» и «Транспорт». В системе также должны присутствовать диаграммы для определения понятий «Ед. массы», «Ед. объема», «Транспорт» и «Прицеп». Связи с маленькой окружностью посередине представляют *роли* (roles). Например, узел «Грузовик» имеет четыре роли: «Макс.

масса», «Грузоподъемность», «Вместимость» и «Число колес». Узел «Автопоезд» имеет две роли: одна помечена как «Имеет частью», а вторая ограничивает число колес значением 18. Метка дуги «v/r» означает *присвоение значения параметра* (value restrictions) допустимых значений для данной роли.

Представление знаний на основе *пропозициональных семантических сетей* (propositional semantic network) было реализовано Стюартом Шапиро (Stuart C. Shapiro) в системе *Semantic Network Processing System* (SNePS) [Shapiro, 1971]. Данная система предназначена для представления широкого класса свойств семантики естественного языка [Shapiro, 1979], [Maida & Shapiro, 1982], [Shapiro & Rapaport, 1992]. На рисунке 4.35 показано представление предложения «*Сью думает, что Боб полагает, что собака ест кость*» в системе SNePS. Каждый из узлов, помеченных от M1 до M5, представляет отдельное предложение, относительное содержание которого соотнесено к *пропозициональному узлу* (propositional node).

Предложение M1 утверждает, что «Сью» — это *потребитель* (experiencer — Expr) *глагола* (verb) «думать», *темой* (theme — Thme) которого является другое предложение M2. Для M2 потребителем является «Боб», глаголом — «полагает», а темой — предложение M3. В предложении M3 присутствует *агент* (agent — Agnt) — некое существо B1, которое является *экземпляром* (member) *класса* (class) «собака», глагол «ест» и *пациент* (patient — Ptnt) — существо B2, которое является экземпляром класса «кость». Данный пример иллюстрирует, как предложения могут быть рассмотрены на метауровне с целью получения других утверждений: M1 утверждает, что M2 есть предмет размышления Сью, а M2, в свою очередь, утверждает, что M3 — это то, что предположил Боб. Система SNePS доступна на сайте исследовательской группы университета Буффало http://www.cse.buffalo.edu/sneps/. Краткое описание работы с системой можно найти в [Муромцев, 2009а].

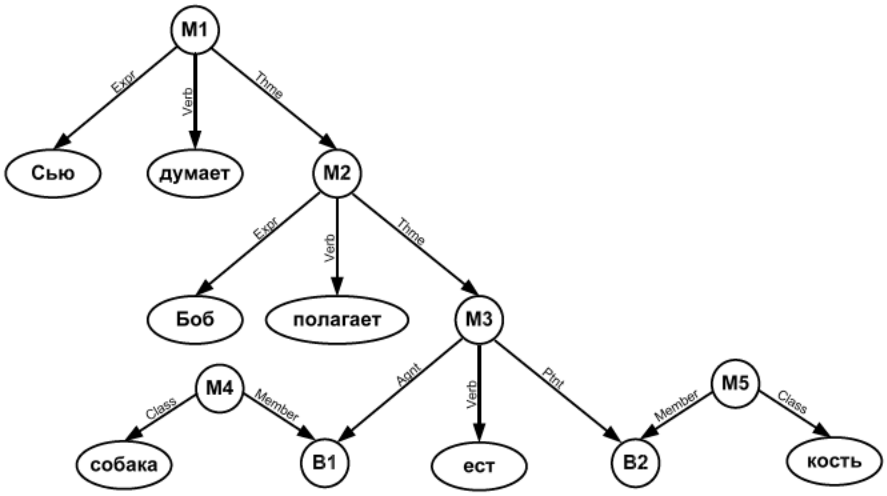


Рис. 4.35

Пример сети в системе SNePS

Среди исследований по *казуальным сетям* (causal networks) следует отметить работы Чака Ригера (Chuck Rieger), разработавшего сеть анализа описания про-

блем на английском языке, позволяющую осуществлять рассуждения на метауровне [Rieger, 1976]. Бенджамин Киперс (Benjamin Kuipers) на основе идей Ригера разработал метод *качественных рассуждений* (qualitative reasoning), который послужил мостом между символическими методами ИИ и дифференциальными исчислениями, используемыми в физике [Kuipers, 1984, 1994]. Идеи Джуди Перла (Judea Pearl) по применению методов статистики и теории вероятности в задачах ИИ реализованы в формализме *сетей доверия* (belief networks), представляющих собой казуальные сети, весам дуг которых назначаются вероятности [Pearl, 1988, 2000].

4.4.2. Концептуальные графы для семантического анализа текста

Одним из наиболее распространенных универсальных сетевых языков для представления смысла является механизм *концептуальных графов* (conceptual graph). Впервые идея концептуальных графов была предложена в работе Джона Сова (John Sowa) [Sowa, 84]. Более позднее описание исследований можно найти в [Sowa, 92]. Современное состояние дел изложено на сайте http://www.jfsowa.com/cg/ или http://conceptualgraphs.org/.

Язык концептуальных графов по своей сути является развитием формализма экзистенциальных графов и графической логики, предложенных Ч. Пирсом. Предпосылкой создания данного языка послужило стремление построить механизм, обладающий выразительностью естественного языка и вычислительными возможностями символической логики, предназначенный для реализации различных сетевых способов представления семантики. При этом автор стремится объединить сети Петри, концептуальные зависимости Шенка, теорию дискурса Кампа, семантическое представление Мельчука и роли Филмора.

Концептуальный граф (conceptual graph) — это конечный связный двудольный граф. Узлы графа бывают двух видов: первые представляют *понятия*, а вторые — *концептуальные отношения* (conceptual relation), при этом метки дуг на графах не используются. Вершины, представляющие концептуальные отношения, служат для отражения взаимосвязи между понятиями. Чтобы различать вершины понятий и отношений используется графическое обозначение двух видов: прямоугольники — для вершин-понятий и эллипсы — для вершин-отношений. На рисунке 4.36 приведен пример концептуального графа для предложения «*Кот гонялся за мышью*», в котором четыре прямоугольника соответствуют концептам: кот, преследовать, мышь и факт преследования, имеющий неявный тип «*ситуация*» («*situation*»), эллипсы соответствуют отношениям: «*агент*» («*agent*»), «*объект*» («*object*») и «*время действия*» («*past*»). Последнее отношение присоединяется к *графу-контексту*, т. е. графу, который содержит вложенный в него подграф.

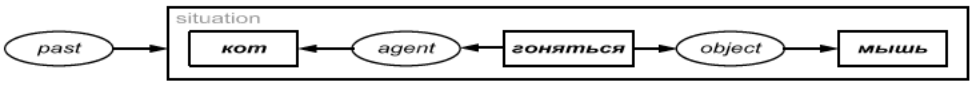


Рис. 4.36
Пример концептуального графа

Возможно также представление концептуального графа и в текстовой форме:

(PAST) → [[КОТ] ← (AGENT) ← [ГОНЯТЬСЯ] → (OBJECT) → [МЫШЬ]]

Одним из преимуществ языка концептуальных графов является простота представления отношений любой арности. В отличие от дуг, соответствующих бинарным отношениям, вершины, представляющие концептуальные отношения, могут быть связаны с любым конечным количеством вершин-понятий. Аналогично какое-либо понятие может быть связано множеством отношений.

Таким образом, концептуальный граф может представлять высказывание любой сложности. Например, на рисунке 4.37 представлен граф для предложения «*Миша сообщил Жене новость по электронной почте*». Глагол «*сообщить*» связан с другими понятиями с помощью четырех концептуальных отношений: «*агент*» («*agent*»), «*объект*» («*object*»), «*получатель*» («*recipient*») и «*средство*» («*mean*»).

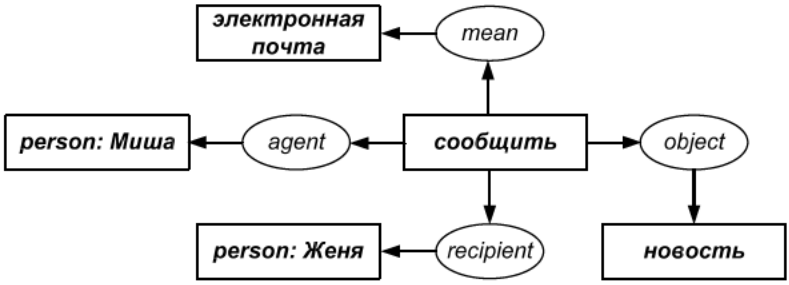


Рис. 4.37

Граф предложения «Миша сообщил Жене новость по электронной почте»

В нотации концептуальных графов вводятся явные обозначения отношений класс–элемент, класс–подкласс. Любое понятие рассматривается как экземпляр конкретного типа. Например, на рисунке 4.37 перед именами понятий «*Миша*» и «*Женя*» используется имя типа «*person*», определяющего тип или класс экземпляра, представленного данной вершиной. Таким образом, можно утверждать, что вершины «*Миша*» и «*Женя*» представляют собой объекты класса «*person*». Классы понятий имеют иерархическую структуру или, иначе говоря, множество классов является частично упорядоченным. Например, класс «*person*», можно рассматривать как подкласс «*homo sapiens*» («*человек разумный*»), который, в свою очередь, является подклассом «*being*» («*живое существо*»). В то же время «*homo sapiens*» является суперклассом для «*person*», а «*being*» — для «*homo sapiens*». Тот факт, что один тип является подклассом другого типа принято обозначать символом: $person \leq homo\ sapiens \leq being$.

Совокупность всех типов образует решетку множественного наследования, в которой каждый тип может иметь множество родителей и детей. Но не все типы могут иметь общих родителей или детей. Поэтому, чтобы решетка типов была единой иерархией, в концептуальные графы включают два специальных типа: *универсальный тип* (universal type), являющийся суперклассом для всех классов, и *абсурдный тип* (absurd type), являющийся подклассом для всех типов. Универсальный тип принято обозначать символом «T», а абсурдный тип — символом «⊥». Прямоугольники с одинаковыми метками типа представляют экземпляры одного класса, но также эти прямоугольники могут представлять или не представлять одно и то же индивидуальное понятие.

Прямоугольники понятий помечаются именем типа и именем экземпляра, разделенными символом двоеточия. Однако можно и не указывать имени экзем-

пляра. В этом случае применяются специальные *маркеры* (marker). Для обозначения любого или неспецифицированного экземпляра используется обобщенный маркер «*». Метки понятий «*person*: *» и «*person*» являются эквивалентными и указывают на произвольный объект типа *person*. В дополнение к обобщенному маркеру «*» допускается использование переменных, например, «*person*: **X*». Переменные используются, если несколько различных вершин на графе обозначают один и тот же неспецифицированный (т. е. не указанный явно) экземпляр класса. На рисунке 4.38 приведен пример концептуального графа для предложения «*Котенок ловит лапой свой хвост*». Хотя и не известно, о каком конкретно котенке идет речь, переменная **X* позволяет указать, что лапа и хвост принадлежат одному и тому же котенку. Для более сложных манипуляций с неспецифицированными экземплярами классов формализм концептуальных графов предполагает использование лямбда-выражений (λ).

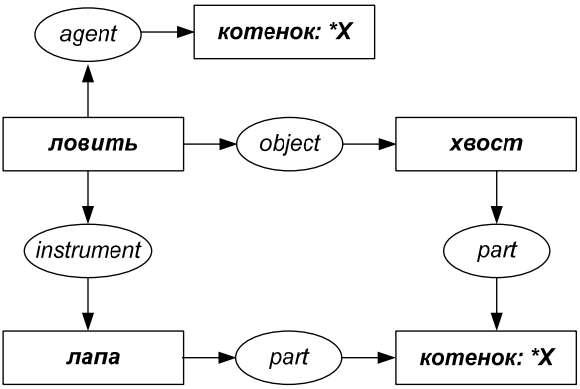


Рис. 4.38

Концептуальный граф для предложения «Котенок ловит лапой свой хвост»

С помощью маркеров можно указать и конкретный экземпляр класса, не используя при этом имени объекта. Для этого служат *индексные маркеры* «#», позволяющие отделить экземпляры от своих имен. Для указания имени объекта можно использовать отношение «*name*». Это позволяет, с одной стороны, указать для одного объекта несколько имен, а с другой стороны — обозначать различные объекты могут одним и тем же именем. Кроме того, само имя можно рассматривать как понятие, которое также может быть связано какими-либо отношениями с другими понятиями. Например, в предложении «*Рыжую лошадь звали женским именем Елена*» речь идет о свойстве имени «*быть женским*», в то же время известно, что эти именем звали какую-то конкретную лошадь. Граф для этой фразы приведен на рисунке 4.39. Отношение «*gender*» («*род*») указывает на свойство имени, а отношение «*color*» («*цвет*») — на свойство конкретной лошади. Примечательно, что из данного графа невозможно сделать заключение о том, какого пола лошадь, хотя нам и очевидно, что, скорее всего, женским именем назовут кобылу, а не коня.

Теория концептуальных графов включает специальные операции для специализации или обобщения графов. На основе этих операций можно формировать новые графы из уже существующих.

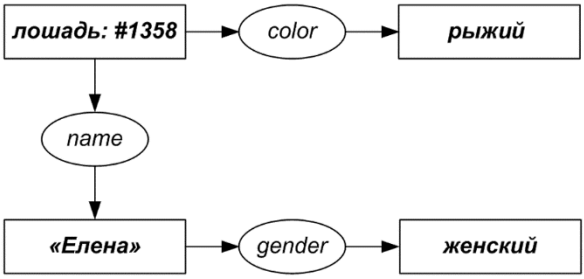


Рис. 4.39

Концептуальный граф с использованием индексного маркера

К этим операциям относятся *канонические правила формирования* (canonical formation rules):

1. *Копирование* (copy) — позволяет получить точную копию какого-либо графа.

2. *Упрощение* (detach) — позволяет исключить дублирующиеся отношения. Дублирование отношений часто возникает в результате операции объединения.

3. *Ограничение* (restrict) — позволяет заменить вершины понятий графа другими вершинами, представляющими их специализацию, или заменить метку типа на метку подтипа.

4. *Объединение* (join) — позволяет интегрировать два графа в один, если одна из вершин первого графа, представляющая какое-либо понятие, идентична одной из вершин второго графа.

Не являясь правилами вывода, канонические правила формирования накладывают определенные ограничения на операции с графами и не позволяют формировать бессмысленные графы на основе осмысленных. Рассмотрим пример из статьи [Sowa, 1976]. Графы для двух исходных предложений «*Менеджер нанял на работу человека*» и «*Работник нанят на работу 13-12-08*» приведены на рисунке 4.40.

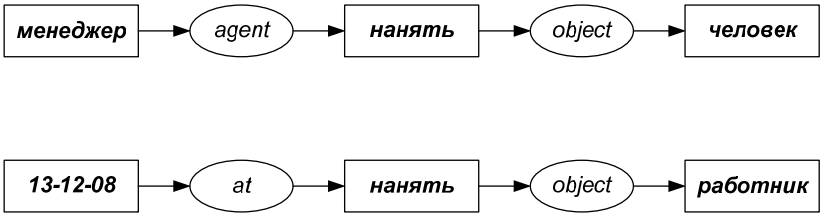


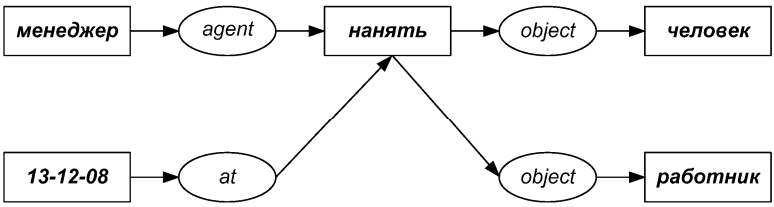
Рис. 4.40

Исходные концептуальные графы

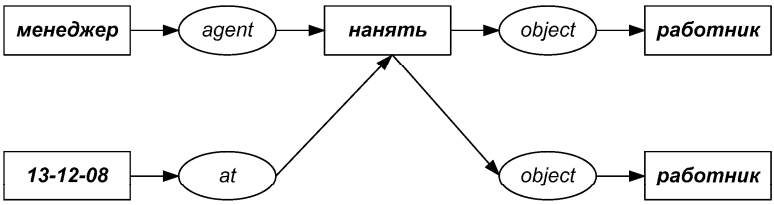
Типы концептов в данном примере образуют следующую иерархию классов «*менеджер* ≤ *работник* ≤ *человек*»: тип «*человек*» является более общим (т. е. суперклассом) по отношению к «*работник*» и «*менеджер*», тип «*менеджер*» является подклассом типа «*работник*».

Оба исходных графа, изображенных на рисунке 4.40, содержат вершину «*нанять*». Применяя правило объединения к этим двум графам, по вершине «*нанять*» получим новый граф, показанный на рисунке 4.41*а*. Далее на основании отношения класс–подкласс между концептами *человек* и *работник* можно применить правило ограничения и заменить метку вершины «*человек*» на метку

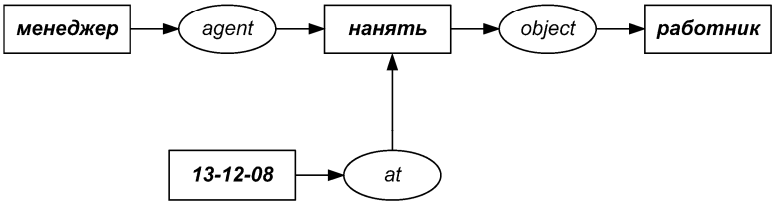
«*работник*». Результат этой операции показан на рисунке 4.41*б*. Полученный граф содержит две вершины «*работник*», которые могут быть объединены в одну, так как указывают на один и тот же концепт. Наконец, операция упрощения позволяет устранить дублирование отношения «*object*» между концептами «*нанять*» и «*работник*». Результирующий граф показан на рисунке 4.41*в*.



а. Операция объединения по вершине «*нанять*»



б. Операция ограничения по иерархии типов *человек* >= *работник*



в. Операция упрощения: устранено дублирвание отношения «*object*»

Рис. 4.41

Примеры использования канонических правил формирования

Часто возникает необходимость для определения отношений между высказываниями или высказыванием и концептом. Например, предложение «*Том верит, что он нравится Мэри*» устанавливает отношение между концептом «*верит*» и высказыванием «*Мэри нравится Том*». В концептуальных графах вводится специальный тип *высказывание* (proposition), объектом ссылки которого является множество концептуальных графов (являющихся подграфами данного графа). Понятия-высказывания, или *пропозициональные понятия*, обозначаются прямоугольниками, содержащими подграф, аналогично графам-контекстам. Пример графа для приведенного выше предложения показан на рисунке 4.42.

Теория концептуальных графов вводит оператор, отображающий построенный граф на формулу исчисления предикатов. Конъюнктивные отношения могут

быть непосредственно представлены вершинами отношений. Реализация логического отрицания возможна с помощью унарного отношения «*neg*» и пропозиционального понятия (графа, представляющего операнд логического отрицания). На основе логического отрицания и конъюнкции можно представить на графах и другие логические операции, используя правила логики. Однако остаются нерешенными ряд вопросов, в том числе и вопрос квантификации переменных.

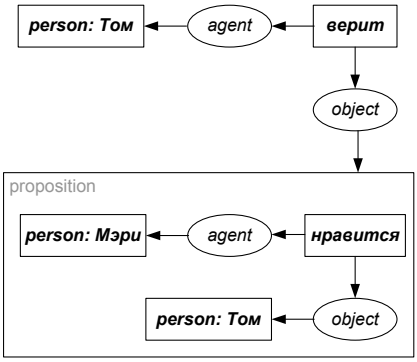


Рис. 4.42

Концептуальный граф с пропозициональным понятием

Формализм концептуальных графов разрабатывался как средство представления высказываний на естественном языке [Sowa, 1991]. По этой причине с момента широкого использования данной парадигмы обсуждались такие вопросы, как представление синтаксических и семантических структур, структуры дискурса, видовременных признаков, лексического выбора, тема-рематических отношений, контекста, метафоры и аналогий. В то же время концептуальные графы эквивалентны по своей выразительной силе исчислению предикатов. Многие инструментальные системы для разработки приложений этой теории включили в себя компонент, соответствующий переводу выражений на естественном языке в представление в виде концептуальных графов, и наоборот, в частности, речь идет о таких проектах, как PEIRCE [Ellis & Levinson, 1992], Loughborough toolset [Kocura et al., 1991], Caramel [Sabah & Vilnat, 1993], объектно-ориентированная оболочка UNE–CG–KEE [Munday et al., 1994], BEELINE [Mann, 1995].

4.5. ФРЕЙМЫ И ПРЕДСТАВЛЕНИЕ СТЕРЕОТИПОВ

Термин *фрейм* (frame), предложенный Марвином Минским (Marvin Minsky), обозначает определенным образом структурированные данные, представляющие некий абстрактный образ или стереотип. Фреймы характеризуются иерархической организацией, включающей, в частности, слоты (конкретные знания об атрибутах фрейма) и имя, выражающее смысл самого фрейма как целого. Это позволяет описывать знания на различных уровнях детализации, переходя от общих прототипов к более конкретным.

В основе теории Минского лежат исследования по восприятию и запоминанию стереотипных ситуаций и прототипному сравнению образов. Отталкиваясь от проблемы восприятия и распознавания пространственных сцен, Минский предложил модель механизмов памяти, объединяющих различные аспекты знания в единой

сущности. Так, например, при восприятии фразы «мальчик пнул мяч», большинство людей скорее представят конкретный футбольный мяч, чем какой-то абстрактный шарообразный образ без атрибутов и признаков.

В данном параграфе рассмотрены основные понятия и элементы теории фреймового представления. Подобно с классическими и современными трудами М. Минского можно познакомиться на его домашней веб-странице http://web.media.mit.edu/~minsky/.

4.5.1. Представление стереотипных ситуаций

Одним из ключевых свойств языка представления знаний является его выразительность при описании типовых ситуаций. Это свойство, прежде всего, связано с представлением семантики высказываний. Понимание даже простейшей фразы на естественном языке предполагает использование большого количества базовых (контекстных) знаний. Экспериментальные исследования показали, что в процессе запоминания люди организуют знания в структуры, соответствующие типовым ситуациям [Barlett, 1932]. Несмотря на то, что одни и те же понятия в разных ситуациях принимают различный смысл, человек без труда находит верную интерпретацию высказывания в рамках существующего контекста. Так, например, читая новости об экономике, спорте или политике, мы, учитывая тематику статьи, не испытываем трудности с однозначным пониманием текста, даже если в тексте имеется много опечаток, сокращений или неполных фраз. В процессе чтения, переходя от одной статьи к другой, человек модифицирует структуру знания, подстраиваясь, таким образом, под контекст получаемой информации. Но если читаемый текст плохо структурирован, одни и те же предложения содержат не связанные по смыслу фрагменты, восприятие такой информации значительно усложняется, так как становиться невозможно быстро установить, к какой структуре знания следует отнести ту или иную фразу. В такой ситуации возникает большое количество ошибок в понимании, потому что непонятно, какой контекст следует использовать для разрешения двусмысленных слов и фрагментов текста, таких как местоимения, сокращения, аббревиатуры и пр.

В то же время большинство людей без труда воспринимает хорошо структурированный текст, даже если этот текст содержит большое количество опечаток. Это объясняется тем, что фразы, построенные в соответствии с типовыми знаниями, легко интерпретируются человеком, так как составляющие их фрагменты однозначно проецируются на более абстрактные стереотипные структуры знаний. Ниже приведен известный пример подобного «текста».

Пример 4.5

По рзелульаттам илссеовадний одонго анлигйсокго унвиертисета, не иеемт занчнеия, в кокам пряокде рсапожолены бкувы в солве. Галвоне, чотбы преавя и пслоендяя бквуы блыи на мсете. Осатьлыне бкувы мгоут селдовтаь в плоонм бсепордяке, всервано ткест чтаитсея без побрелм. Пичрионй эгото ялвятеся то, что мы не чиатем кдаужю бкуву по отдльенотси, а все солво цликеом.

Проблемы эффективного представления и понимания «общих знаний», структурирования семантики выражений, установление взаимосвязи между общими и частными понятиями были подняты Марвином Минским (Marvin Minsky) в статье [Minsky, 1974]. Сущность своей теории он сформулировал следующим образом: при возникновении новой или изменении содержания существующей ситуации в процессе выполнения каких-либо действий из памяти выбирается структура, называемая *фреймом* (frame). Эта структура представляет собой некий шаб-

лон, который может быть адаптирован под сложившуюся реальность. Таким образом, фреймы позволяют описать взаимосвязи понятий предметной области в виде организованных статических структур без занесения в них реальных данных.

Альтернативный подход для описания стереотипной последовательности действий в заданном контексте предложили Роджер Шенк (Roger C. Schank) и Роберт Абельсон (Robert P. Abelson) в работе [Shank, Abelson, 1977]. Этот подход, получивший название *сценарии* (script), первоначально рассматривался как способ структурирования данных, удобный для представления концептуальных зависимостей (умозаключений) при описании типовых ситуаций. Впоследствии возникла идея, что таким же образом можно моделировать память и обработку текста [Schank, 1982]. С современным состоянием проблем представления и обработки *общих знаний* (common sense knowledge) можно ознакомиться на сайте лаборатории MIT Media Lab (http://xnet.media.mit.edu/), в частности решению этих задач посвящен проект X-Net Knowledge Bases.

4.5.2. Прототипное сравнение образов

В психологии предполагается, что в долговременной памяти человека хранятся не конкретные эталоны образов и тем более не детальные признаки этих образов, а некоторое их абстрактное представление в виде *прототипов*. Реальный образ сопоставляется с прототипом и при наличии сходства происходит его опознавание [Солсо, 2006]. Подтверждений в пользу теории *прототипного сравнения* множество: мы легко опознаем известную нам марку машины, даже если у нее другой цвет и форма, или, например, она обвешена различными украшениями, наклейками, антеннами, которые никак не могут соответствовать идеальному представлению о данной машине в нашей памяти. Прототипы содержат не только наиболее общее представление какого-либо образа, но и детальное описание его отличительных особенностей. Это подтверждается тем, что мы редко путаем буквы O и Q или B, R и P, несмотря на то, что сходство между ними весьма велико.

В теории прототипов рассматриваются две модели формирования: *центральная тенденция* и *частота признаков* [Солсо, 2006]. Согласно первой модели, прототип представляет собой среднее из всех экземпляров образцов и является абстракцией, хранящейся в памяти и отражающей центральную тенденцию некоторой категории. Вторая модель предполагает, что прототип отражает сочетание наиболее часто встречающихся признаков, т. е. является уникальным синтезом признаков, относящихся к различным образам. Благодаря этому, если новый образ состоит из большого количества известных признаков, у человека возникает уверенность, что этот он уже видел ранее. Многочисленные эксперименты о формировании у человека зрительного прототипа, проведенные различными психологами показали следующее:

- человек формирует прототипы на основе усредненных характеристик отдельных образов данного класса;
- приобретение знаний о прототипе возможно даже в том случае, когда доступны для изучения только его видоизмененные формы;
- прототипы обладают обобщенными признаками, благодаря которым хорошо известные прототипы содержат меньше признаков, чем менее знакомые или недавно приобретенные;
- о модифицированных экземплярах человек судит по степени их близости к прототипу;
- формирование прототипа происходит путем абстрагирования отдельных образцов.

Один из подходов к изучению организации и опознавания зрительных образов в памяти человека предложен в рамках широко известного направления в психологии, под названием *гештальт-психология* [Вертгеймер, 1987]. По мнению гештальт-психологов, при восприятии образа поступающие стимулы действуют совместно и производят бóльшее впечатление, чем сумма ощущений от каждого отдельного стимула. Таким образом, происходит естественная или спонтанная организация признаков. Этим объясняются многочисленные «оптические обманы» при восприятии образов. Мысленный взгляд при этом постоянно пытается отыскать альтернативные абстрактные образы (прототипы) при поступлении одних и тех же стимулов, и «видимый» образ может видоизменяться.

Влияние памяти на восприятие формы можно проиллюстрировать следующим примером. На рисунке 4.43 изображены две одинаковых фигуры, но фигура **а** повернута по отношению к фигуре **б** на 45°. При этом возникает ощущение, что фигура **а** — это устойчивый двухмерный объект, а фигура **б** — неустойчивый трехмерный. Этот феномен можно объяснить тем, что исходя из прошлого опыта фигура **б** напоминает куб. Мы видим переднюю и заднюю плоскости, а также одно из боковых ребер. На рисунке **а** в первую очередь видны два повернутых и соединенных между собой квадрата. Таким образом, прототип фигуры **а** является плоским, а фигуры **б** — объемным.

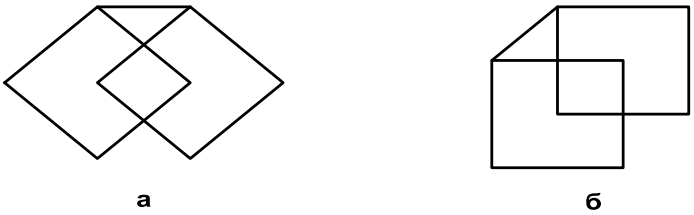


Рис. 4.43
Влияние ориентации образа на его восприятие — одна из форм кажется трехмерной

Проблемы восприятия и распознавания пространственных сцен рассматриваются в работах Минского, в частности в [Minsky, 1974]. Минский в то время работал над проблемой восприятия сцены при движении робота по комнате. Рассматривалась задача определения соответствия различных проекций одному и тому же объекту. Минский начал с рассмотрения упрощенной системы фреймов для пространственного описания куба. Это описание было представлено в виде графа, в котором одни узлы соответствовали более высокой степени абстракции и описывали понятия, постоянно присутствующие на данной сцене (куб). На более низких уровнях были представлены узлы, соответствующие текущему состоянию сцены (перечень видимых сторон). На рисунке 4.44 приведено описание куба, данное Минским в работе [Minsky, 1974]. Для вершин указаны прототипы соответствующих фреймов, а для дуг — отношения между фреймами.

При перемещении наблюдателя сцена начинает меняться: одни стороны исчезают, а другие появляются в поле зрения. Например, при вращении влево сторона ***A*** исчезнет, а сторона ***B*** появится. Следовательно, для каждой проекции можно сопоставить сеть фреймов, отражающую соотношение видимых сторон. На рисунке 4.45 показаны три проекции куба при повороте сцены на 45° и соответствующие этим проекциям сети фреймов.

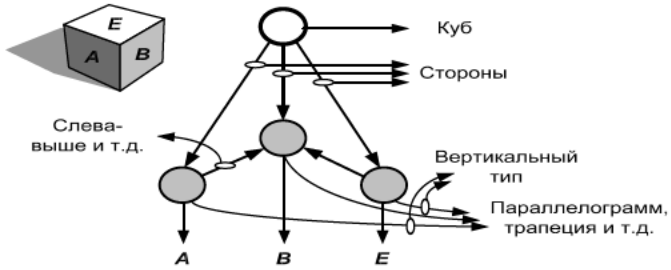


Рис. 4.44

Пространственное представление куба в виде сети фреймов

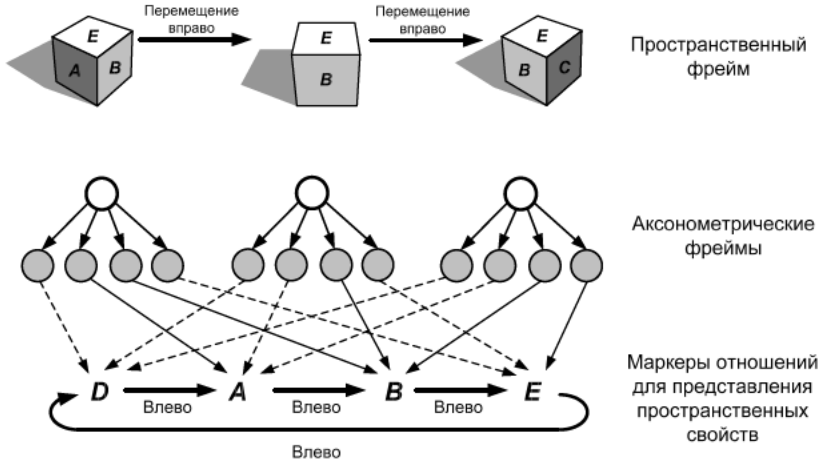


Рис. 4.45

Пространственные и аксонометрические фреймы при изменении проекции

Пунктирные линии показывают невидимые плоскости в данной проекции. При изменении наблюдаемой сцены осуществляется повторный анализ всех объектов, в процессе которого человек подбирает наиболее подходящие к новому описанию прототипы. На этом этапе необходимо конкретизировать описание прототипа, но в условиях ограниченной информации (например, часть сторон куба не видны) необходимо выполнить реконструкцию описания, используя связи между фреймами. Как только фреймы установлены, выполняется идентификация отношений (например, взаимное расположение сторон). Наконец, используя связь между уровнями абстракции, можно установить, что на данной сцене представлена одна из проекций того же самого объекта, что и на предыдущей сцене.

Описанные механизмы верны не только для представления в памяти визуальных сцен. Как показали эксперименты, при игре в шахматы опытные игроки используют механизмы памяти, близкие к прототипному сравнению. Концепция фреймового представления знаний наиболее близко моделирует эти механизмы памяти, объединяя различные аспекты знания в единой сущности.

4.5.3. Структура фреймового представления

Впервые термин «фрейм» был предложен в 1970-е гг. Минским [Minsky, 1974], который определил его следующим образом:

> «Фрейм — это структура данных, представляющая стереотипную ситуацию, вроде нахождения внутри некоторого рода жилой комнаты или сбора на вечеринку по поводу дня рождения ребенка. К каждому фрейму присоединяется несколько видов информации. Часть этой информации — о том, как использовать фрейм. Часть о том, чего можно ожидать далее. Часть о том, что следует делать, если эти ожидания не подтвердятся».

Из определения Минского следует, что фрейм подобен *записи* (record), в которой сохранена информация, описывающая стереотипную ситуацию, и которая может быть адаптирована под реальное состояние среды. Подобные структуры можно представить и с помощью семантических сетей. Но для представления каждой стереотипной ситуации понадобится отдельная сеть. В сравнении с сетью фрейм представляет собой более естественную форму для отображения иерархически организованных знаний. Такого рода знания базируются на понятии абстрактного образа или ситуации. В психологии и философии известно понятие абстрактного образа. Например, слово «комната» вызывает следующий стереотипный образ: «жилое помещение с четырьмя стенами, полом, потолком, окнами и дверью, площадью 6–20 м^2». Это описание является необходимым, но необязательно достаточным. Например, комната без окон — это чулан или кладовка, а не комната. Абстрактный образ имеет своего рода «дырки» в определении — это незаполненные значения некоторых атрибутов — количество окон, цвет стен, высота потолка, покрытие пола и др.

В основе теории фреймов лежит предположение, что представление понятий в мозге человека не требует четкого определения набора свойств, а базируется на понятии типа или класса, объединяющего в себе наиболее общие свойства. Сущности, соответствующие классам, получили название *прототипа* (prototype). Прототип — это уже не абстрактный образ, а наиболее типичный представитель своего класса, с обобщенными, но вполне конкретными значениями своих свойств. Например, прототип понятия четырехугольник можно определить, как фигуру, имеющую четыре угла. Возникает желание задать значения величин углов по умолчанию одинаковыми. И этому есть вполне убедительное объяснение. Рассматривая различные примеры фигур, представленных на рисунке 4.46 слева направо, кажется, что «типичность» объектов увеличивается. Четырехугольник, не обладающий выпуклостью, кажется «менее» типичным, чем прямоугольник, возможно потому, что понятие «площадь фигуры» в нашем сознании кореллирует с длиной периметра. А это лучше просматривается у фигуры с равными внутренним углами.

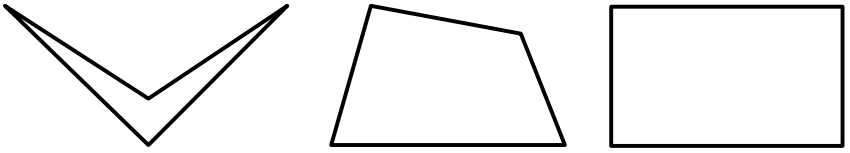


Рис. 4.46

«Менее» и «более» типичные четырехугольники

Прототип может быть основой для описания любой сущности данного класса в случае неполноты информации, так как представляет обобщенные свойства большинства разновидностей объектов класса. Так, если требуется оценить площадь более или менее прямоугольного участка земли, для которого известны длины сторон, то можно оценить площадь, предполагая, что внутренние углы его контура равны. В худшем случае, если предположение о равенстве углов будет нарушено, то значение площади будет завышено, что в большинстве случаев вполне допустимо.

Основная идея фреймового подхода заключается в сосредоточении всей информации, относящейся к одному объекту в одной структуре данных вместо распределения ее по базе знаний. Действительно, оказываясь в новой ситуации, человек организует и запоминает получаемые знания во фреймоподобных структурах. В дальнейшем эти структуры могут быть вызваны из памяти и адаптированы под сложившуюся действительность. Так, например, впервые оказавшись на вокзале, человек составляет представление обо всех вокзалах. Это просторное здание с большими холлами внутри и красивым фасадом. К вокзалу примыкают платформы, куда прибывают и откуда отправляются поезда. На вокзале всегда имеются один или несколько залов ожидания, билетные кассы и почти всегда — ресторан. Детали внутреннего интерьера залов ожидания могут варьироваться. Может изменяться цвет стен, количество и размер окон, удобство кресел. С фреймом вокзала также связывается информация, принимаемая по умолчанию. Если необходимо узнать время отправления ближайшего поезда, следует искать расписание или обратиться в информационное окно. Таким образом, человеку не нужно подстраивать сознание под каждый новый вокзал, напротив, оказавшись на вокзале в чужой стране, мы чувствуем себя увереннее, так как общая обстановка нам кажется знакомой.

Структура фрейма достаточно проста (табл. 4.8): с представляемым понятием связывается список именованных атрибутов, называемых *слотами* (slot). Каждый слот может содержать следующую информацию:

- *значение параметра, представленного слотом*. Например, количество путей на вокзале;
- *ссылка на другой фрейм*. «Автомат по продаже билетов» может быть специализированным понятием «Терминал»;
- *информация по умолчанию*. Используется, если нет другой информации. Например, кассы работают 24 часа в сутки;
- *дескрипторы для значений слотов*. Дескрипторы представляют собой требования, используемые для определения соответствия новых объектов стереотипу фрейма;
- *неинициализированные значения*. Слоты могут оставаться незаполненными, пока они не понадобятся для решения какой-либо задачи. Например, направления следования поездов является обязательным атрибутом каждого вокзала, но эти направления не могут быть определены, пока не известно о каком конкретно вокзале идет речь;
- *метаинформация* — информация о допустимых значениях для заполнения слотов.

У фрейма обязательно должно быть имя, по которому может осуществляться его идентификация. Также важной особенностью фреймов является возможность связывать со слотами произвольные *процедуры*, выполняемые при смене его значения. Эти процедуры в литературе часто именуются *демонами* (demon). Со слотом можно связать любое количество процедур, но наиболее часто используются следующие:

- Процедура на событие «если добавлено» (IF-REMOVED). Выполняется, когда новая информация записывается в слот.
- Процедура на событие «если удалено» (IF-ADDED). Выполняется, когда информация удаляется из слота.
- Процедура на событие «по требованию» (IF-NEEDED). Выполняется, когда запрашивается информация из пустого слота.

Таблица 4.8

Структура фрейма

Название понятия (имя фрейма)		
Слоты	Значения	Список связанных процедур
Имя слота 1	Значение 1	Процедура 1
Имя слота 2	Значение 2	Процедура 2.1, Процедура 2.2
...	...	...
Имя слота N	Значение N	Процедура N

Фреймовое представление имеет ряд преимуществ перед сетевым представлением. Фреймовое описание стереотипной ситуации намного ясней сетевой версии и не зависит от интерпретации соотношения вершин и дуг. В сети каждое понятие описывается на одном и том же уровне детализации. Иерархическая структура фреймов часто бывает гораздо удобней сетевой, так как не требует рассмотрения понятия как единой сущности (сети) с детализацией всех аспектов. Например, пока автомобиль исправно работает, нам важны только ходовые характеристики, экономичность, эргономика и т. п. Механическое устройство агрегатов нас не интересует, пока не произошла какая-либо поломка. И только при обнаружении неисправности мы обращаемся к справочнику и пытаемся устранить поломку.

Фреймы-прототипы, хранящиеся в базе знаний, позволяют реализовать *наследование* (inheritance) свойств одних фреймов от других. Аналогично тому, как в семантической сети наследование происходит по транзитивным отношениям («имеет частью», «это есть»), фреймы-экземпляры могут наследовать свойства фреймов-прототипов через специальные слоты АКО (A Kind Of = «это есть»). Слот АКО указывает на фрейм более высокого уровня иерархии, откуда неявно наследуются, т. е. переносятся, список и значения слотов. Фреймы-экземпляры создаются в БЗ для отображения реальных объектов и ситуаций на основе поступающих данных. Возможно наследование свойств от нескольких прототипов. Такой вид наследования получи название *множественное наследование* (multiple inheritance).

С помощью фреймовой модели можно представить все многообразие знаний о мире через:

- фреймы-структуры, для обозначения объектов и понятий (устройство, отчет, заказ);
- фреймы-роли (менеджер, пользователь, клиент);
- фреймы-сценарии (банкротство, собрание акционеров, тестирование);
- фреймы-ситуации (тревога, авария, рабочий режим устройства) и др.

В качестве примера рассмотрим формирование понятия «заказ товара». Понятие «Заказ» сочетает в себе свойства понятий «Заявка» и «Счет». Следовательно, можно наследовать прототип фрейма «Заказ» от фреймов «Заявка» и «Счет» (рис. 4.47).

Для заказа определенного товара «X» следует ввести цену и форму оплаты. При этом, предположим, что в большинстве случаев этот товар покупает в количестве одна штука за наличный расчет. Таким образом, можно записать значения по умол-

чанию в слоты «Количество» и «Форма оплаты». Для конкретного покупателя продавец может изменить количество или предоставить скидку. В этом случае должны отработать процедуры пересчета значений.

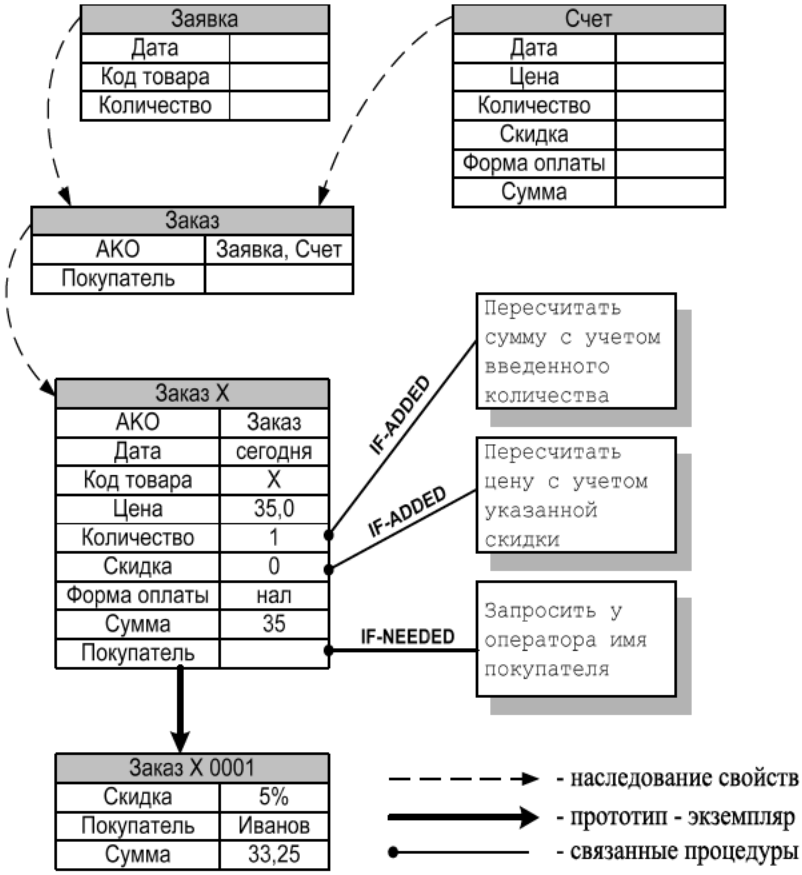


Рис. 4.47

Пример описания знаний с помощью фреймов

Использование фреймов совместно с семантическими сетями позволяет сделать модель представления знаний значительно более наглядной и гибкой. Заменяя вершины-понятия в сети на вершины-фреймы, можно перейти от громоздкой сети к более компактным диаграммам, где иерархические отношения и структура понятий описываются фреймами, а семантика отношений между понятиями представляется сетью. Следует отметить, что отношения также могут быть реализованы в виде фреймовых структур, что позволит адаптировать заданные типы отношений под конкретную ситуацию.

Естественным развитием теории фреймов в ходе исследований, проведенных в 1970-х гг. в Массачусетском технологическом институте и исследовательском центре Xerox Palo Alto Research Center, стала разработка парадигмы объектно-ориентированного представления, о которой речь пойдет ниже.

4.5.4. Множественное наследование

Одно из главных фундаментальных понятий теории фреймов — «наследование свойств» — впервые появилось в работах Квиллиана, названное термином «когнитивная экономия». В теории фреймов является общепризнанным, что некоторый прототип может наследовать свойства от многих родителей в системе фреймов. В основном граф иерархии наследования представляет собой дерево. Но если некоторые свойства отдельных ветвей пересекаются, то представляется целесообразным выделение этой группы свойств в отдельный фрейм. Например, иерархия определений многоугольников содержит понятия «Равносторонний треугольник» и «Квадрат». Очевидно, принцип классификации фигур по признаку количества сторон никак не отражает равенство этих сторон. Следовательно, можно выделить в данной иерархии «правильные» фигуры, обладающие правильным контуром, и наследовать понятия «Равносторонний треугольник» и «Квадрат» от понятия «Правильный многоугольник» (рис. 4.48).

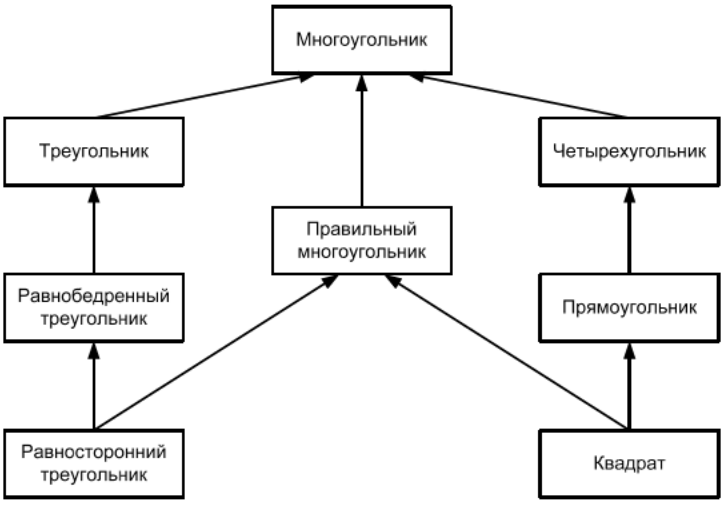


Рис. 4.48

Иерархия наследования для геометрических фигур

Таким образом, с помощью механизма множественного наследования, появляется возможность передать таким фреймам, как «Квадрат» и «Равносторонний треугольник», свойства, характерные именно для равносторонних фигур. Например, все равносторонние многоугольники имеют равные значения внутренних углов. Следуя принципу когнитивной экономии лучше всего хранить информацию об этих свойствах именно во фрейме «Правильный многоугольник».

Множественное наследование может быть эффективно использовано и в случае, если наследуемое понятие комбинирует в себе свойства различных категорий или даже иерархий. Если на рисунке 4.48 была показана единая классификация однотипных понятий «Многоугольники», то на рисунке 4.49 представлены сразу несколько параллельных иерархий для понятий «Транспортное средство», «Морские суда» и «Летательный аппарат». Корневые узлы этих иерархий представляют собой непересекающиеся понятия. Так, летательный аппарат вовсе не обязательно должен быть транспортным средством — таким аппаратом, например, является «крылатая ракета». Морские судна типа «плавучий кран» тоже не могут

использоваться как транспорт. Следовательно, узлы для всех этих понятий не обязательно должны иметь общего предшественника. В то же время «Катер на воздушной подушке» является одновременно и транспортным средством, и летательным аппаратом, и способен перемещаться по воде. Конфликты между параллельными определениями порождают двусмысленности, которые должны разрешаться во время проектирования баз знаний. По этой причине на практике следует избегать таких двусмысленностей или неоднозначностей.

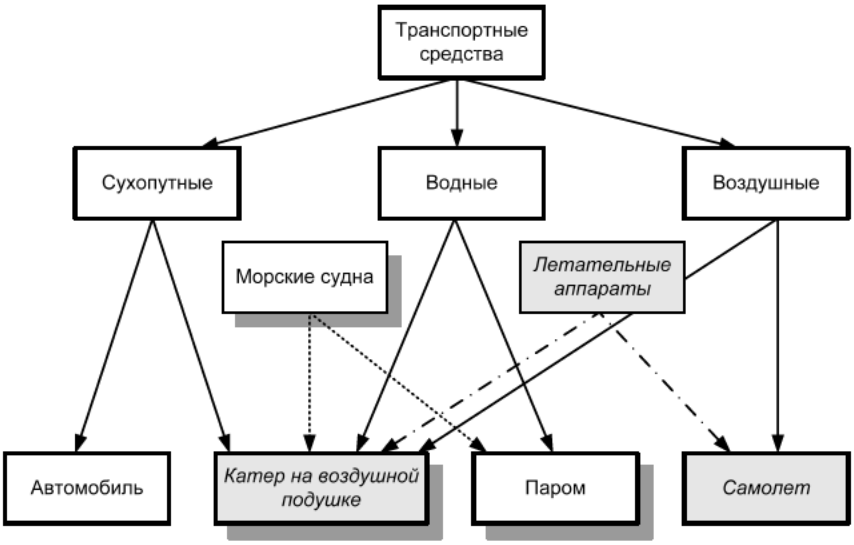


Рис. 4.49

Пример сложной иерархии множественного наследования

Множественное наследование в системе фреймов не влечет за собой никаких проблем только до тех пор, пока информация, поступающая от различных родителей, не становится противоречивой. Классический пример такой ситуации, представленный на рисунке 4.50, в литературе получил имя собственное — «Алмаз Никсона» (Nickson diamond) [Джексон, 2001], а представленная в этом примере проблема — *проблема алмаза* (diamond problem), так как иерархия по ромбовидной форме напоминает алмаз. На рисунке представлена классификация политических убеждений человека, живущего в США. Положим, что квакеры — это пацифисты, т. е. в слоте «пацифист» фрейма «квакер» должно быть значение истина. Напротив, республиканцы пацифистами не являются, значит, в слоте «пацифизм» фрейма «республиканец» будет значение «ложь». Но что в таком случае можно сказать о квакере, который является сторонником Республиканской партии? Именно в такой роли выступал бывший президент США Ричард Никсон. Является он пацифистом или нет? Иными словами, откуда должен унаследовать квакер-республиканец значение слота «пацифизм», если считать, что мы не располагаем никакой дополнительной уточняющей информацией?

Поскольку значения, предлагаемые по умолчанию, конфликтуют друг с другом, мы, используя только ранее введенную информацию, не можем ничего сказать о пацифизме Ричарда Никсона. В такой ситуации некоторые системы, использующие механизм наследования, отказываются давать однозначное заклю-

чение. Системы с таким поведением получили наименование *скептических*. Другие, обнаружив подобный конфликт, осуществляют выбор случайно. За ними закрепилось определение *доверчивые*. Разумеется, эти стратегии не могут гарантированно разрешить конфликт. Самый правильный выход в этой ситуации — это явно указать значения для конфликтующих слотов. Есть и альтернативный вариант — подключить к слоту специальную процедуру — демон по требованию, использующий для устранения неоднозначности какие-либо «посторонние» знания, которыми мы не располагаем на стадии конструирования системы фреймов.



Рис. 4.50

Пример противоречий при множественном наследовании «Алмаз Никсона»

Другая известная слабая сторона в архитектуре множественного наследования получила название *проблема хрупкости базового класса* (fragile base class problem). Базовое (корневое) понятие в иерархии является «хрупким» в том смысле, что изменения во фрейме этого узла могут быть небезопасными для наследуемых фреймов. На основании анализа одного базового уровня иерархии невозможно определить, приведут ли эти изменения к нарушениям функционирования фреймов более низкого уровня наследования.

Наконец, еще одной проблемой часто оказывается непомерно большой граф иерархии наследования. Для работы с этим графом необходимо постоянно перескакивать с одного узла иерархии на другой, чтобы понять логику представляемого процесса или концепта. Эта ситуация получила название *проблема йо-йо* (yo-yo problem), так как работа с подобной иерархией напоминает слежение за движением игрушки «йо-йо» (вращающаяся катушка с намотанной на нее ниткой) вверх-вниз. Единственное решение в данной ситуации максимально сократить иерархию наследований и прибегнуть к композиции свойств (добавление слотов со ссылкой на фреймы) вместо наследования.

4.5.5. Сценарии

Сценарий (script) — один из видов фрейма, предложенных в [Schank, Abelson, 1977] как средство представления стереотипной последовательности действий в частном контексте. Изначально сценарии применялись для организации структур данных, удобных для представления *концептуальной зависимости* (conceptual dependency) в описании типовых ситуаций и причинно-следственных отношений. Позднее сценарии начали использовать и в обработке естественного языка для описания стереотипной смены событий в привычных ситуациях [Lehnert, 1980]. В психологии предполагается, что большинство сценариев пове-

дения усваивается в детстве в результате либо прямого опыта, либо при наблюдении за окружающим миром. Многие ситуации становятся известны из книг, телевидения или кино: мало кто лично участвовал в ограблении банка, угоне самолета или в пытках и т. п., но почти все примерно представляют себе, как это делается, а значит, обладают соответствующими сценариями.

На основе сценариев можно моделировать память и обработку текста [Schank, 1982]. Типовая ситуация посещения ресторана знакома большинству людей, так как необходимая последовательность действий не вызывает никаких вопросов: проследовать к столику, ознакомиться с меню, сделать заказ, дождаться блюд, оплатить счет и оставить чаевые. В статье Шенка (Schank) поход в ресторан описан через сценарий «РЕСТОРАН». Компонентами сценария являются следующие элементы:

- *начальные условия* (Entry conditions), которые должны выполняться при вызове сценария. В сценарии «РЕСТОРАН» эти условия включают открытый ресторан, посетителя, желающего поесть и имеющего необходимую сумму денег для оплаты;
- *результаты*, или *факты* (Results), которые становятся истинными по завершении выполнения сценария. В данном примере это прибыль ресторана, чаевые для официанта, клиент сыт и доволен;
- *реквизиты* (Props), составляющие контекст сценария. Это могут быть меню, официант, свободный столик и т. д. Множество предположений описывает реализацию сценария, принятую по умолчанию;
- *роли* (Roles), представляющие действия, совершаемые отдельными участниками сценария. Клиент делает заказ, ест и платит, повар готовит пищу, а официант обслуживает клиента;
- *сцены* (Scene). Последовательность сцен составляет сюжет сценария. Каждая сцена соответствует определенному временному участку выполнения сценария. В данном примере сценами могут быть вход, заказ, принятие пищи и т. п.

Из указанных элементов сценария «собираются» отношения концептуальной зависимости. Описанный выше сценарий «РЕСТОРАН» представлен в таблице 4.9.

В данном примере используются примитивы из теории *концептуальных зависимостей* Шенка (Schank) и Ригера (Rieger) [Schank, Rieger, 1974]. В рамках этой теории смысл выражений определяется с помощью четырех примитивов:

- ACT — действия (actions);
- PP — объекты (picture procedures);
- AA — модификаторы действий (action aiders);
- PA — модификаторы объектов (picture aiders).

Предполагается, что каждое действие должно приводить к созданию одного или нескольких примитивов ACT. Основными компонентами действий являются следующие примитивы:

- ATRANS — передавать отношение (давать);
- PTRANS — передавать физическое расположение объекта (идти);
- PROPEL — прикладывать физическую силу к объекту (толкать);
- MOVE — перемещать часть или все тело (владельцем);
- GRASP — захватывать объект (исполнителем);
- INGEST — поглощать объект (есть);
- EXPEL — издавать звуки (кричать);
- MTRANS — передавать ментальную информацию (сказать);
- MBUILD — создавать новую ментальную информацию (решать);
- CONC — осмысливать идею (думать);

Таблица 4.8

Сценарий «РЕСТОРАН» из статьи [Schank, Abelson, 1977]

Сценарий: РЕСТОРАН Аналог: Кафе Реквизиты: Столы Меню F=Пища (Food) Чек Деньги	**Сцена 1:** Вход S PTRANS S в ресторан S ATTEND выбирает стол S MBUILD куда сесть S PTRANS S за стол S MOVE S в сидячее положение
Roles: S=Клиент W=Официант C= Повар M=Кассир O=Владелец	**Сцена 2:** Заказ (меню на столе) (W приносит меню) (S спрашивает меню) S PTRANS меню S — S MTRANS сигнал W W PTRANS W к столу S MTRANS ‘меню’ W W PTRANS W к меню W PTRANS W к столу W ATRANS меню S S MTRANS список блюд S *S MBUILD выбор F S MTRANS сигнал W W PTRANS W к столу S MTRANS ‘Я буду есть F’ W W PTRANS W к C W MTRANS (ATRANS F) к C
Нач. условия: S голоден у S есть деньги	C MTRANS ‘нет F’ W W PTRANS W к S W MTRANS ‘нет F’ S вернуться к (*) или перейти к Сцене 4 на (уход без оплаты) C выполняет (сценарий ‘приготовить F’) к Сцене 3
Результаты: S потратил деньги O заработал деньги S сыт S доволен (optional)	**Сцена 3:** Прием пищи C ATRANS F to W W ATRANS F to S S INGEST F (вариант: вернуться к Сцене 2, чтобы сделать еще заказ; иначе, перейти к Сцене 4)
	Сцена 4: Выход S MTRANS к W (W ATRANS чек S) W MOVE выписывать чек W PTRANS W к S W ATRANS чек S S ATRANS чаевые W S PTRANS S к M S ATRANS деньги M S PTRANS S к выходу ресторана (уход без оплаты)

- SPEAK — произносить звуки (говорить);
- ATTEND — воспринимать звуки (слушать).

Отношения концептуальной зависимости — это концептуальные синтаксические правила, выражающие семантические связи в соответствии с грамматикой языка. Выражение «Клиент направляется в ресторан» будет выглядеть следующим образом: «S PTRANS S в ресторан». Другое выражение «Официант дает меню клиенту» можно представить, например, так: «W ATRANS меню S».

Программа, описанная в статье [Schank, Abelson, 1977], способна читать небольшой текст о ресторане и преобразовывать этот текст во внутреннее представление концептуальной зависимости. Начальные условия сценария сопоставляются с ключевыми понятиями текста, а роли и реквизиты сценария связываются с описываемыми объектами. Отсутствующая в тексте информация дополняется информацией по умолчанию из сценария, таким образом, представляемый контекст может быть расширен типовыми понятиями. После обработки текста программа способна отвечать на вопросы, ссылаясь на сценарий.

Пример 4.6. Понимание естественного языка на основе сценария РЕСТОРАН

«Вчера вечером Джон зашел в ресторан. Он заказал бифштекс. Когда он оплачивал счет, обнаружилось, что он остался без денег. Он поспешил домой, так как начинался дождь». Основываясь на этом описании программа Шенка на основании сценария могла ответить на вопросы типа: «Обедал ли Джон прошлым вечером?», «Что использовал Джон: наличные или карточку?», «Как Джон получил меню, что он купил?» Примечательно, что в тексте нет явного указания на многие действия Джона, например, «обедал ли Джон», известно лишь, что он зашел в ресторан и заказал бифштекс. Согласно сценарию, в этой ситуации он должен был сначала занять какой-либо столик, прочитать меню, затем съесть заказанное блюдо. А при оплате наличными (в тексте говорится, что деньги закончились; в случае оплаты карточкой он бы об этом не узнал, так как для этого необходимо обратиться в банк за выпиской), возможно, он не смог оставить чаевые.

На основе сценариев можно и синтезировать подобные описания стереотипных ситуаций. Например, сценарий РЕСТОРАН («омар», «Джон», «Метрополь») достаточен для того, чтобы воспроизвести повествование о том, как Джон ел омара в ресторане «Метрополь». Сценарии можно считать разновидностью *схемы* (schema; Schema; schéma), термина используемого для описания способа представления операциональной информации в когнитивной психологии [Bartlett, 1932]. В рамках этой теории понятиями оперируют как структурами, упорядочивающими поток входящей и исходящей информации. Последовательная когнитивная переработка внешних сигналов (восприятия, ощущения) и внутренних возбуждений (представления о собственных импульсах к действию, чувства и т. п.) составляет поток сознания. Конструирование понятий привносит в него порядок, определяя их сходство и различие. В результате этой переработки формируется множество схем, составляющих некий *поведенческий контекст*. При этом происходит упорядочивание понятий, а схемы (или сценарии) являются не столько результатом, сколько факторами упорядочения при обработке информации [Schönpflug, 1987]. Сценарий позволяет одновременно представлять и когнитивную репрезентацию такого контекста, и план, по которому можно учитывать этот контекст в будущем поведении [Kruse, 1986].

В более поздней концепции пришли к определению сценария как набора ожиданий о том, что в воспринимаемой ситуации должно произойти дальше [Schank, 1990]. Многие ситуации в жизни можно проинтерпретировать так, словно участ-

ники этих ситуаций «играют» свою роль, заранее заготовленную в тексте «некоторой пьесы». Официант следует роли официанта, клиент — роли клиента. Жизненный опыт часто означает знание того, как поступать и как поступят другие в конкретных стереотипных ситуациях. Это знание и называется сценарием. Была также выдвинута гипотеза, что размышление и вообще мышление человека представляет собой применение одного из сценариев. Мы живем, следуя своим сценариям или предписаниям. Чем больше мы знаем, тем в большем числе ситуаций мы чувствуем себя комфортно, т. е. способны эффективно играть разные роли. В большинстве случаев мышление — поиск имеющихся сценариев и сличение их со старыми (готовыми), а не порождение новых идей и вопросов. Но сценарии не могут дать исчерпывающих ответов [Демьянков, 1996].

Один сценарий связан со стереотипизированной последовательностью других сценариев с общими для них ролями [Abelson, 1981; Schank, Abelson, 1977]. Сценарии позволяют понимать не только реальную или описываемую ситуацию, но и детальный план поведения, предписываемый в этой ситуации [Bellezza, Bower, 1982]. Существуют центральные, зависимые и связанные между собой сценарии. Разные люди (как показано в большом числе исследований) довольно единообразно отвечают на вопрос о том, какие события составляют данный сценарий [Graeser et al., 1979]. Информация, подходящая для активированного сценария, понимается и вспоминается быстрее, чем та, которая для него является нерелевантной.

Проблемы хранения и поиска знаний очень трудны и характерны для моделирования семантики понятий. Евгений Черняк показал, какое количество знаний требуется для понимания даже простых детских рассказов [Chairniak, 1972]. Но, несмотря на эти проблемы, программы, использующие в качестве языка представления знаний сценарии и другие семантические представления, способны понимать естественный язык в ограниченной предметной области и эффективно находить ответы на поставленные вопросы. Примером может служить программа, интерпретирующая сообщения, поступающие по каналам информационных лент. Благодаря использованию сценариев природных катастроф, успешных сделок и других стереотипных ситуаций, программа весьма успешно работала в этой ограниченной, но вполне реальной предметной области [Schank, Riesbeck, 1981].

4.6. ОБЪЕКТНО-ОРИЕНТИРОВАННАЯ МОДЕЛЬ

Объектно-ориентированная технология (object-oriented approach) программирования (ООП) сформировалась в начале 1980-х гг. [Буч, 1992; Буч, 1998]. Сегодня она по праву получила широкое распространение для решения самых различных задач: традиционные и интеллектуальные задачи программирования, хранение данных и интерактивная графика, и многое другое. В каком-то смысле объектный подход можно считать развитием фреймового представления. Однако если фреймы изначально предназначены для представления знаний и подразумевают описание семантической структуры информации, то объектно-ориентированная технология является универсальным средством программирования и может применяться для решения любых задач.

Объектно-ориентированному программированию посвящено огромное количество публикаций, которые в основном предлагают описание каких-либо коммерческих инструментальных средств. И хотя большинство из этих инструментариев имеют свои ограничения на объектную модель, подробное ознакомление с сутью подхода можно получить на примере любого объектно-ориентированного

языка программирования. В силу указанных соображений, содержание данного параграфа ограничено только описанием основных понятий «чистого» объектно-ориентированного подхода.

При всей мощности и выразительности объектно-ориентированного моделирования решение слабоформализованных задач с трудом вписываются в данную методологию, так как часто механизм решения описываются людьми в терминах правил «ЕСЛИ ..., ТО ...». В этой ситуации оказывается эффективным применение *объектно-структурированных компонентов* (object-structured components). Различие между чистым объектно-ориентированным и объектно-структурированным программированием можно проиллюстрировать на примере Microsoft's Visual Basic (VB). VB не является объектно-ориентированным языком, но предоставляет множество структур, обладающих свойствами объектов. Это упрощение делает VB более легким в применении, что и объясняет его популярность. Объектно-структурированный подход для разработки экспертных систем реализован, в частности, в системе Exsys CORVID [Муромцев, 2006]. Настраивая объектно-ориентированные структуры, разработчик экспертной системы может сохранить привычный способ описания задачи, шаги и логику ее решения. Различные типы этих структур позволяют использовать широкий спектр предопределенных функций, упрощая и ускоряя процесс наполнения базы знаний. Объектно-ориентированные структуры являются гибким и мощным механизмом разработки экспертных систем, позволяющим объединить различные модели представления знаний, а также автоматизирующим способы получения исходных фактов и отображения результатов.

В последнее десятилетие в объектно-ориентированном анализе получили широкое распространение так называемые *шаблоны проектирования* (design patterns), содержащие описания наиболее удачных проектных и архитектурных решений. Представление прошедших временем методик в виде шаблонов проектирования возводит на новый качественный уровень идею повторного использования объектов, предлагая разработчикам новых систем доступ к формализованному опыту проектирования предыдущих разработок. Примечательно, что впервые идею шаблонов проектирования озвучил и внедрил в практику проектирования зданий и сооружений Кристофер Александр (Christopher Alexander).

В области разработки интеллектуальных систем формализованное описание повторяющихся схем решений задач на абстрактном уровне, без привязки к какой-либо предметной области, было реализовано в системе KADS — Knowledge Analysis and Design Support. Этот проект вырос из европейской программы «The European ESPRIT IT Programme» и постепенно трансформировался в проект CommonKADS, предлагающий новую методологию инженерии знаний, де-факто являющейся в настоящее время стандартом проектирования и разработки систем, основанных на знаниях.

Ниже приведены основные понятия ООП, наиболее критичные для задач представления знаний и разработки интеллектуальных систем. Вопросы, связанные с проектированием (методология CommonKADS и шаблоны проектирования), рассматриваются в следующей главе.

4.6.1. Абстрагирование и классификация в объектно-ориентированном проектировании

В основе методологии ООП лежит использование естественных способностей человеческого мышления к классификации и абстрагированию. Объекты и их свойства, наследование и иерархия — с этими понятиями человек ежесекундно сталкивается в своей повседневной жизни, они привычны и интуитивно понятны

любому [Хабаров, 2001]. По своей сути, ООП предполагает моделирование понятий, концепций и отношений в предметной области, а не описание последовательности действий, свойственных классическому программированию, что делает ООП эффективным методом представления знаний.

Центральным элементом в ООП является некое понятие, которое подлежит объектно-ориентированному анализу. При этом понятие, используемое в процессе решения какой-либо задачи, неразрывно связано с мышлением эксперта, решающего эту задачу. С точки зрения логики, мышление заключает в себе образование суждений и умозаключений путем анализа и синтеза понятий. т. е., другими словами, анализ понятий ведет к формированию знаний.

Среди логических приемов формирования понятий наиболее важными, по всей видимости, являются абстрагирование и классификация. Под *абстрагированием* следует понимать мысленное выделение одних признаков предмета и отвлечение от других. *Классификация* — это распределение предметов по группам, где каждая группа, каждый класс имеют свое постоянное место. Очень важную роль играет адекватный выбор базиса для классификации. Классификация может проводиться как в контексте решаемой задачи, так и на основании наиболее общих признаков.

Абстрагирование является одним из основных методов, используемых для решения сложных задач. Различные авторы по-разному трактуют процесс получения абстракций. Одни предлагают акцентировать внимание на нахождении сходств между определенными объектами, ситуациями или процессами реального мира, отвлекаясь от имеющихся различий. Другие считают важным выделение деталей, существенных для рассмотрения и использования, и игнорирование остальных. Третьи предлагают считать абстракциями только то, что может быть изложено, осмыслено и проанализировано независимо от механизма, который принят для ее реализации. Суммируя эти разные точки зрения, Грэди Буч (Grady Booch) приводит следующее определение абстракции [Буч, 1998]:

> Абстракция выделяет существенные характеристики некоторого объекта, отличающие его от всех других видов объектов, и таким образом четко определяет его концептуальные границы с точки зрения наблюдателя.

В процессе абстрагирования существенные внешние особенности поведения объекта должны быть отделены от несущественных. Это разделение смысла и реализации Хал Абельсон (Hal Abelson) и Геральд Зюссман (Gerald Jay Sussman) назвали барьером абстракции [Abelson, Sussman, 1985], который основывается на принципе минимизации связей, отражающих только существенные аспекты поведения объекта.

Выбор правильного набора абстракций для заданной предметной области представляет собой главную задачу объектно-ориентированного проектирования. В работе [Seidewitz, Stark, 1986] приведен набор абстракций, начиная с объектов, которые почти точно соответствуют реалиям предметной области, и кончая бессмысленными объектами:

- абстракция сущности — объект представляет собой полезную модель некой сущности в предметной области;
- абстракция поведения — объект состоит из обобщенного множества операций;
- абстракция виртуальной машины — объект группирует операции, которые либо вместе используются более высоким уровнем управления, либо сами используют некоторый набор операций более низкого уровня;
- произвольная абстракция — объект включает в себя набор операций, не имеющих друг с другом ничего общего.

Абстрактные сущности должны прямо соответствовать сущностям предметной области. При этом внешнее проявление, или поведение, объекта рассматривается с точки зрения его взаимодействия с другими объектами, в соответствии с которым должно быть выполнено и его внутреннее устройство. Эту идею Бертран Мейер (Bertrand Meyer) назвал *контрактной моделью* программирования [Meyer, 1988]. Контракт фиксирует все обязательства, которые один объект имеет перед другим. Обязательствам соответствуют операции объекта, предусмотренные его контрактом, которые однозначно определяются множеством формальных параметров и типом возвращаемого значения.

Центральной идеей абстракции является понятие инварианта. *Инвариант* — это некоторое логическое условие, значение которого (истина или ложь) должно сохраняться. Для каждой операции объекта можно задать *предусловия* (инварианты предполагаемые операцией) и *постусловия* (инварианты, которым удовлетворяет операция). Изменение инварианта нарушает контракт, связанный с абстракцией [Буч, 1998].

Примеры 4.7. Абстракция датчика (Адаптировано из [Буч, 1998])

В тепличном хозяйстве, использующем гидропонику, растения выращиваются на питательном растворе без песка, гравия или другой почвы. Управление режимом работы парниковой установки зависит как от вида выращиваемых культур, так и от стадии выращивания. Нужно контролировать целый ряд факторов: температуру, влажность, освещение, кислотность (показатель pH) и концентрацию питательных веществ. Одна из ключевых абстракций в такой задаче — датчики, измеряющие температуру воды и воздуха, влажность, pH, освещение и концентрацию питательных веществ. С внешней точки зрения датчик температуры — это объект, который способен измерять температуру там, где он расположен. Для датчика температуры существенно не столько само местоположение, сколько тот факт, что данный датчик расположен именно в данном месте и это отличает его от других датчиков. Теперь можно задать вопрос о том, каковы обязанности датчика температуры? Мы решаем, что датчик должен знать температуру в своем местонахождении и сообщать ее по запросу (вызову операции). Одно из определений абстрактного датчика температуры на языке Python может выглядеть следующим образом:

```
class TemperatureSensor():
  def __init__(self, location):
    self.location = location
    self.temperature = 0
  def currentTemperature(self):
    return self.temperature
  ...
```

Класс TemperatureSensor — это только спецификация датчика; настоящая его начинка скрыта в его зарытых, невидимых другим объектам операциях. Класс TemperatureSensor это еще не объект. Реальные датчики — это его *экземпляры,* и их нужно создать, прежде чем с ними можно будет оперировать. Теперь рассмотрим инварианты, связанные с операцией currentTemperature. Предусловие включает предположение, что датчик установлен в правильным месте в теплице, а постусловие — что датчик возвращает значение температуры в градусах Фаренгейта.

Еще одним важнейшим методом ООП является классификация. Очень краткое, но в тоже время конкретное определение дано в [Буч, 1998]:

Классификация — средство упорядочения знаний.

В объектно-ориентированном анализе определение общих свойств объектов помогает найти общие ключевые абстракции и механизмы. Наиболее сложным этапом объектно-ориентированного проектирования является определение классов и объектов. Корректная классификация позволяет выделить общие свойства объектов, имеющих одинаковое строение или одинаковое поведение, что помогает правильно определить модульную структуру предметной области.

Создание разумной классификации, являясь весьма трудоемкой задачей, составляет существенную часть любой научной или инженерной деятельности. Кристофер Александр (Christopher Alexander), автор теории шаблонного проектирования в области строительной архитектуры и городского планирования, очень точно охарактеризовал работу архитектора: «его проектная деятельность, и скромная, и гигантская по размеру, управляется целиком образами, которые он держит в своем сознании в данный момент, и его способностью комбинировать эти образы при создании нового проекта» [Alexander, 1979]. Аналогичные трудности возникают при решении задач практически в любой предметной области. Рассмотрим примеры классификации для двух других научных дисциплин: биологии и химии [Буч, 1998].

Примеры 4.8. Проблемы классификации живых существ

Вплоть до XVIII в. идея о возможности классификации живых организмов по степени сложности была господствующей. Мера сложности была субъективной, поэтому неудивительно, что человек оказался в списке на первом месте. В середине XVIII в. шведский ботаник Карл Линней предложил более подробную таксономию для классификации организмов: он ввел понятия рода и вида. Век спустя Дарвин выдвинул теорию, по которой механизмом эволюции является естественный отбор и ныне существующие виды животных — продукт эволюции древних организмов. Теория Дарвина основывалась на разумной классификации видов, в которой наиболее похожие живые организмы относились к одному классу, а различные — к разным. В современной биологии термин классификация обозначает установление иерархической системы категорий на основе предположительно существующих естественных связей между организмами. Наиболее общее понятие в биологической таксономии — царство, затем, в порядке убывания общности: тип (отдел), класс, отряд (порядок), семейство, род и, наконец, вид. Исторически сложилось так, что место каждого организма в иерархической системе определяется на основании внешнего и внутреннего строения тела и эволюционных связей. В современной классификации живых существ выделяются группы организмов, имеющих общую генетическую историю, т. е. организмы, имеющие сходные ДНК, включаются в одну группу. По современным воззрениям дельфины ближе к коровам, чем к форели. Но даже по критерию ДНК, классификация не может быть универсальной. Результат классификации зависит от задачи, которую требуется решить. Например, если необходимо, чтобы классификация говорила о кровном родстве видов, будет получен один ответ, если же следует отразить уровень приспособления, ответ будет другой.

Примеры 4.9. Классификация в химии

В древние времена считалось, что все вещества суть комбинации земли, воздуха, огня и воды. В настоящее время такая классификация не может считаться сколько-нибудь удовлетворительной. В середине XVII в. Роберт Бойль предложил элементы как примитивные химические абстракции, из которых составляются более сложные вещества. Век спустя, в 1789 г., Лавуазье опубликовал первый список, содержащий 23 элемента, хотя впоследствии было открыто, что некоторые из них таковыми не являются. Но открытие новых элементов продолжалось, список увеличивался. Наконец, в 1869 г. Менделеев предложил периодический закон, который давал точные критерии для классификации известных элементов и даже мог предсказывать

свойства еще не открытых элементов. Но даже периодический закон не был концом истории о классификации элементов. В начале XX в. были открыты элементы с одинаковыми химическими свойствами, но с разными атомными весами — изотопы.

Эти примеры показывают, что даже в строгих научных дисциплинах методы и критерии классификации зависят от поставленных целей. Однако усилиями многочисленных философов, лингвистов, когнитологов и математиков были сформированы три подхода к классификации [Буч, 1998]:

- классическая категоризация — на основе свойств объектов;
- концептуальная кластеризация — на основе определений;
- теория прототипов — на основе примеров.

Классическая категоризация

В классическом подходе, берущем свое начало от Платона и Аристотеля, классификация понятий осуществляется, исходя из определенного свойства или совокупности свойств, на основании которых формируется некоторая категория. Причем наличие этих свойств является необходимым и достаточным условием, определяющим категорию. Например, пол человека — это категория: каждый человек или мужчина, или женщина, и этот признак достаточен для решения вопроса, к какой категории принадлежит тот или иной индивидуум. С другой стороны, высокие люди не определяют категории, так как, например, для команды баскетболистов это понятие не то же самое, что для верховых скачек. Таким образом, классический подход в качестве критерия похожести объектов использует родственность их свойств. В частности, объекты можно разбивать на непересекающиеся множества в зависимости от наличия или отсутствия некоторого признака. Минский предположил, что «лучшими являются такие наборы свойств, элементы которых мало взаимодействуют между собой. Этим объясняется всеобщая любовь к таким критериям, как размер, цвет, форма и материал. Так как эти критерии не пересекаются, про какой-нибудь предмет можно утверждать, что он большой, серый, круглый и деревянный» [Minsky, 1986].

Концептуальная кластеризация

Это современный вариант классического подхода. Он возник из попыток формального представления знаний и подразумевает формирование концептуальных описаний классов (кластеров объектов), на основании которых затем выполняется классификация сущностей. Например, возьмем понятие «приключенческий фильм». Это именно понятие, а не признак или свойство, поскольку количество приключений в фильме едва ли можно измерить. Но если можно утверждать, что фильм скорее про приключения, чем про что-то другое, то мы помещаем его в эту категорию. Концептуальную кластеризацию можно связать с теорией нечетких (многозначных) множеств, в которой объект может принадлежать к нескольким категориям одновременно с разной степенью точности. Концептуальная кластеризация делает в классификации абсолютные суждения, основываясь на наилучшем согласии.

Теория прототипов

Существуют ситуации, в которых классическая категоризация и концептуальная кластеризация не работают. Многие абстракции не имеют ни четких свойств, ни четкого определения. Джорж Лакофф (George Lakoff) объясняет эту проблему так: «По утверждению Виттгенштейна (Wittgenstein), существуют категории (например, игры), которые не соответствуют классически образцам, так как нет признаков, свойственных всем играм… По этой причине их можно объединить так называемой се-

мейной схожестью... Виттгенштейн утверждает, что у категории игр нет четкой границы. Категорию можно расширить и включить новые виды игр при условии, что они напоминают уже известные игры» [Lakoff, Johnson, 2003]. Вот почему этот подход называется теорией прототипов: класс определяется одним объектом-прототипом, и новый объект можно отнести к классу при условии, что он наделен существенным сходством с прототипом. Лакофф и Джонсон применяют классификацию на основе прототипов к проблеме стульев. Они замечают, что «мы считаем мягкий пуф, парикмахерское кресло и складной стул стульями не потому, что они удовлетворяют некоторому фиксированному набору признаков прототипа, но потому, что они имеют достаточное фамильное сходство с прототипом... Не требуется никакого общего набора свойств прототипа, которое годилось бы и для пуфика, и для парикмахерского кресла, но они оба — стулья, так как каждый из них в отдельности похож на прототипный стул, пусть даже каждый по-своему. Свойства, определяемые при взаимодействии с объектом (свойства взаимодействия), являются главными при определении семейного сходства» [Lakoff, Johnson, 2003]. Понятие свойств взаимодействия — центральное для теории прототипов. В концептуальной кластеризации группировка осуществляется в соответствии с различными концепциями. В теории прототипов классификация объектов производится по степени их сходства с конкретным прототипом.

На практике выявление классов и объектов сначала производится по свойствам, важным в данной ситуации, т. е. выделяются и отбираются структуры и типы поведения с помощью словаря предметной области. Если таким путем не удалось построить удобоваримой структуры классов, можно применить концептуальный подход. В этом случае в центре внимания оказывается поведение объектов, когда они взаимодействуют друг с другом. Наконец, выделяются прототипы и ассоциирующие с ними объекты. Эти три способа классификации составляют теоретическую основу объектно-ориентированного подхода к анализу, предлагающего много практических советов и правил, которые можно применять для идентификации классов и объектов при анализе предметной области [Буч, 1998].

4.6.2. «Чистая» объектно-ориентированная модель

Представление концептуальных понятий в объектно-ориентированном языке осуществляется с помощью *классов* (class), задача которых — объединить связанные данные (атрибуты) и поведение (методы). Классы могут *наследовать* (inherit) некоторые или все свойства своих родителей — *суперклассов* (superclass), они также могут определять свои собственные *атрибуты* (attribute) и *методы* (method). Таким образом, классы образуют *иерархию наследования* (inheritance hierarchy). Классы, как правило, выступают в качестве шаблонов (аналогично прототипам для фреймов) для создания экземпляров классов — *объектов* (object). Различные экземпляры одного и того же класса обычно содержат различные данные, но имеют единообразное представление, т. е. значения атрибутов объектов различны, а их перечень и методы для всех объектов одного класса одни и те же.

При использовании объектно-ориентированного подхода не принято использовать прямой доступ к атрибутам какого-либо класса из методов других классов. Взаимодействие между классами осуществляется с помощью отправления *сообщений* (message), или вызовов методов объектов. Когда объект получает сообщение, он может реагировать на него по-разному, в зависимости от своего текущего состояния. Методы и атрибуты класса, или *члены* класса, подразделяются на *открытые* (public) и *закрытые* (private). Открытые члены класса образуют его *ин-*

терфейс (class interface), через который объекты данного класса взаимодействуют с другими объектами. Закрытые члены класса недоступны для других объектов. Сокрытие реализации класса и отделение его внутреннего представления от внешнего получило название *инкапсуляция* (encapsulation).

Еще одним фундаментальным понятием в объектно-ориентированном программировании, наряду с наследованием и инкапсуляцией, является *полиморфизм* (polymorphism). Слово полиморфизм греческого происхождения и означает «имеющий много форм». Формально полиморфизм — это положение теории ООП, согласно которому одинаковые имена методов могут обозначать различное поведение объектов для нескольких классов, имеющих общий суперкласс. Следовательно, любой объект, обозначаемый полиморфным именем, может по-своему реагировать на некий общий набор операций. Суть изложенного выше заключается в переопределении поведения дочерних классов в зависимости от их семантики. Примером полиморфизма может быть практически любая иерархия. В зоологии весь животный мир объединен в иерархию видов. Каждое животное способно перемещаться в пространстве, но одни ходят, другие — плавают, третьи — летают, а кто-то может и летать, и плавать, и ходить. Этот пример как нельзя лучше иллюстрирует полиморфизм применительно к понятию «перемещаться в пространстве».

Некоторые объектно-ориентированные языки программирования предусматривают *интроспективные* (introspective) или *рефлексивные* (reflex) объекты. Это объекты, в поведении которых предусмотрено предоставление собственного описания: принадлежность к классу данного экземпляра, имена суперклассов, список доступных методов и атрибутов. Интроспекция позволяет функции или методу, управляющему объектами, принимать решения, основываясь на том, какой вид или тип объекта ему передается.

Во многих ООП-языках сами классы также являются объектами, которые можно создавать, передавать и подвергать интроспекции. Но поскольку объекты, как отмечалось, создаются с использованием классов в качестве шаблонов, то для создания объектов-классов также нужны шаблоны. Подобные классы для классов получили название *метаклассы* (metaclass), которые позволяют динамически генерировать новые классы с различными атрибутами и методами, или, иначе говоря, управлять процессом создания не только объектов, но и классов. Этот аспект особенно важен тогда, когда за ранее невозможно предусмотреть в тексте программы все возможные классы. Примером такой ситуации является анализ текста — концепции формируются по ходу анализа и зависят от содержания нового текста. Наконец, поскольку метакласс сам является классом, то нет никакого смысла в заведении «мета-метаклассов».

Объектно-ориентированное программирование позволяет создавать расширяемые системы. Это одно из самых значительных достоинств ООП, и именно оно отличает данный подход от традиционных методов программирования. Расширяемость означает, что существующую систему можно заставить работать с новыми компонентами, причем без внесения в нее каких-либо изменений. Компоненты могут быть добавлены на этапе исполнения программы. Независимые от предметной области части системы могут быть реализованы в виде набора универсальных классов и в дальнейшем расширены за счет добавления частей, специфичных для конкретного приложения. Эти преимущества делают ООП мощным средством не только представления данных, но и знаний. Однако, чтобы эффективно использовать эти преимущества в задачах искусственного интеллекта, требуются значительные вычислительные ресурсы и развитые (как правило, дорогие) инструментальные комплексы.

ЧАСТЬ II

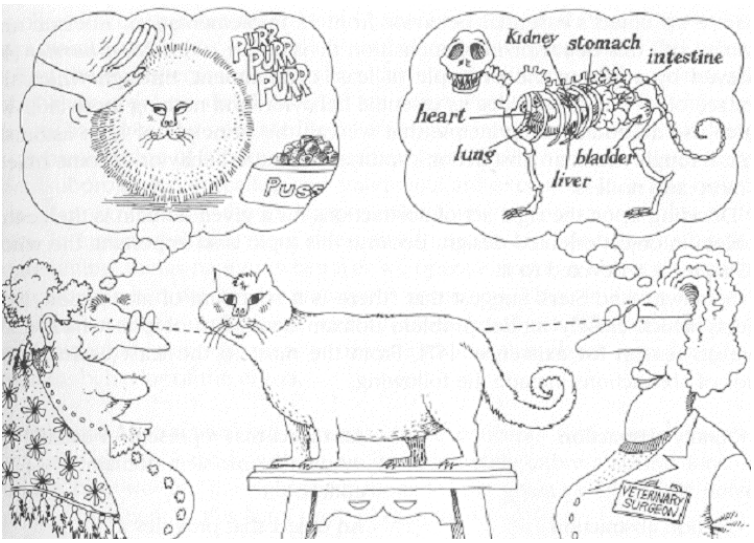
Глава 5

ОНТОЛОГИИ

5.1. ПОТРЕБНОСТЬ В ОНТОЛОГИЯХ

«Люди не могут обмениваться знаниями,
если у них нет общего языка».
Том Дейвенпорт

Для одного и того же объекта реального мира различные люди и программные агенты могут иметь различные абстрактные модели (рис. 5.1), что препятствует обмену знаниями между ними.



Источник: [Буч, 1998]

Рис. 5.1

Различие в абстрактных моделях разных субъектов при рассмотрении одного и того же объекта

Для понимания роли «общего языка» в работе со знаниями рассмотрим роль знаковой системы в коммуникации.

Обычно коммуникацией называют: а) передачу информации, т. е. сообщение; б) взаимодействие, т. е. общение. Она может осуществляться в процессе любой деятельности с помощью различных знаковых способов, например, речи, жестов, азбуки Морзе и др. Изучением таких знаковых систем занимается семиотика.

Семиотика, или семиология (от греч. Semeiotikon, от semeion — знак, признак) — наука, исследующая свойства знаковых систем и, соответственно, знаков (естественных и искусственных). С одной стороны, семиотика близка к кибернетике, исследующей процессы связи и управления в живом организме, природе и обществе, а с другой — к лингвистике, изучающей человеческий язык [Василик, 2007].

Знаковая система — это совокупность знаков, устроенная определенным образом. Знаковая система есть материальный посредник, который служит для обмена информацией между двумя другими материальными системами (например, человек — знаковая система — человек, человек — знаковая система — компьютер).

Три основных аспекта изучения знака и знаковой системы, выделяемые семиотикой:

1) синтактика, изучающая отношения между знаками, т. е. ***внутренние свойства систем знаков*** (иначе, правила построения знаков в рамках знаковой системы);

2) семантика, изучающая отношения между знаками и обозначаемым предметом, т. е. ***содержанием знаков***;

3) прагматика, изучающая отношения между знаком и человеком-пользователем знака: говорящим, слушающим, пишущим, читающим, т. е. ***назначение знаков***.

Примером знаков могут служить широко используемые в операционных системах ярлыки-иконки, например, ярлык с изображением принтера. Его изображение на экране является именем знака. Содержание знака может быть раскрыто с помощью перехода от имени к таблице, содержащей все нужные сведения о принтерах, а включение процедуры печати при щелчке по ярлыку принтера определяет назначение этого знака.

Синтактика изучает объективные законы устройства знаковых систем. Ее задачей является описание запаса правильно построенных текстов (составных знаков) для различных классов знаковых систем. В случае естественного языка формальное описание синтактики фрагментарно (например, в русском языке множество исключений из правил) вследствие избыточности и динамичности естественного языка. Для искусственных языков синтактика, как правило, описывает все допустимые тексты. Для этого в теории формальных языков используется грамматика — любой конечный механизм задания языка.

Семантика изучает законы смысла. Центральным понятием семантики является понятие значения. Обычно для разъяснения структуры языкового значения используется «семантический треугольник» или «треугольник Фреге».

Строение знака — треугольник Фреге (рис. 5.2):

I. Вещь, предмет, явление действительности и т. д. Иное название — денотат.

II. Знак: в лингвистике, например, фонетическое слово или написанное слово; в математике — математический символ; иное название, принятое особенно в философии и математической логике, — имя.

III. Понятие о предмете, вещи. Иные названия: в лингвистике — сигнификат, десигнат, в математике — смысл имени, или концепт денотата.

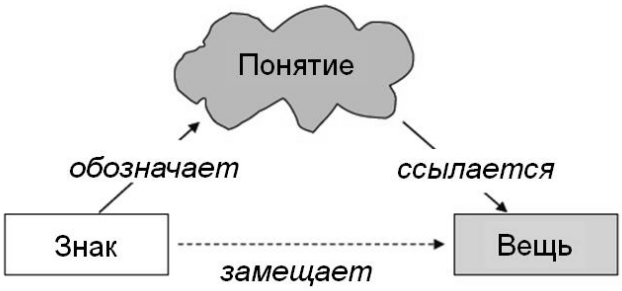


Рис. 5.2

«Семантический треугольник» или «треугольник Фреге»

Например, слово «стул» обозначает класс реально существующих предметов. Все существующие виды стульев отображаются в нашем сознании в виде понятия о стуле. Значение слова стул содержит минимальный набор характерных признаков (признаков понятия), позволяющих соотнести слово и обозначаемый объект.

Возможные расширения «треугольника Фреге»:

- Иногда вершиной треугольника обозначают не саму вещь (денотат), а ее восприятие или представление о ней, словом, ее отражение в сознании человека, которое уже, в свою очередь, связано с денотатом [Поспелов, Осипов, 1999];
- Иногда в схему вводится образ в качестве вершины, для разделения понятия о предмете на две части: логическую («левополушарную») и образную («правополушарную») [Валькман, Исмагилова, 2004].

В «треугольнике Фреге» часто возникает неоднозначность соответствия знак — понятие — денотат, выражающаяся в омонимии, полисемии и синонимии.

Омонимия (рис. 5.3) состоит в том, что существует несколько совершенно одинаковых по форме знаков, каждый из которых имеет свое значение, причем эти значения между собой абсолютно не связаны, так же как и соответствующие денотаты.

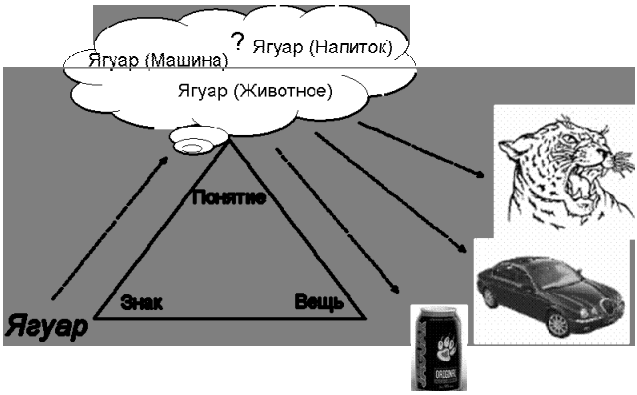


Рис. 5.3
Пример омонимии

Полисемия представляет собой наличие различных значений у одного и того же знака. Обычно данный термин применяется в ситуациях, когда эти различные значения в какой-либо мере связаны между собой (в отличие от омонимии).

Пример: определения понятия для словосочетания «Капитальный ремонт», встречающиеся на одном из отечественных предприятий:

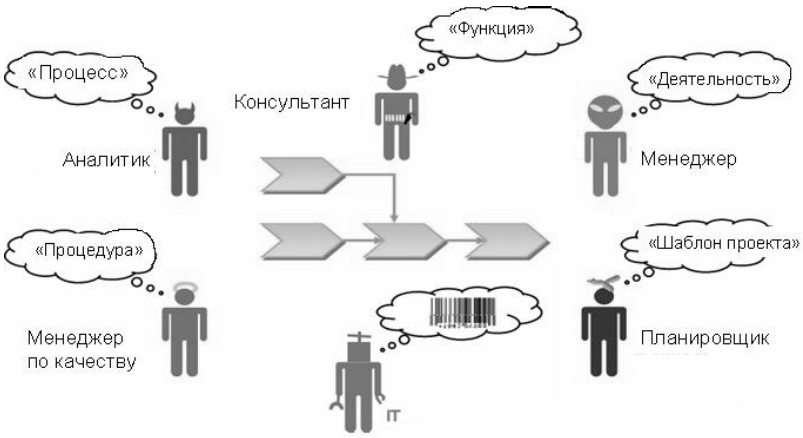
– работы по техническому обслуживанию и ремонту оборудования, выполняемые в капитальный останов, т. е. останов производственной линии длительностью свыше 24-х часов.

– ремонт, выполняемый для восстановления исправности и полного или близкого к полному восстановлению ресурса изделия с заменой или восстановлением любых его частей, включая базовые.

Синонимия (рис. 5.4) указывает на равнозначность, но не тождественность знаков. Под равнозначностью понимается: 1. либо соотнесенность с одним и тем же денотатом (предметом); 2. либо соотнесенность с одним и тем же понятием, точнее с той его частью, которая содержит характеризующую информацию.

Для снижения представленной выше неоднозначности соответствия знак — понятие — денотат необходим «общий язык», включающий в себя:

- Строго определенный словарь лексических единиц (знаков),
- Непротиворечивое понимание того, какие понятия обозначаются заданными лексическими единицами (знаками).



По материалам компании FutureModels

Рис. 5.4
Пример распространенной синонимии в организациях

Кроме потребности в общем языке организации испытывают потребность в интеграции информации и создании единого информационного пространства. Необходимые знания могут находиться в различных местах:

- В документах (например, отчет по маркетинговому исследованию),
- В головах сотрудников (например, неформализованные знания специалиста отдела продаж X о клиенте Y),
- В базах данных (например, формализованные знания о клиенте Y в базе данных системы управления отношениями с клиентами, CRM),
- В базах знаний (например, в аналитическом модуле системы управления отношениями с клиентами, CRM).

При появлении потребности в знаниях необходимо проводить раздельный поиск в различных источниках (рис. 5.5):

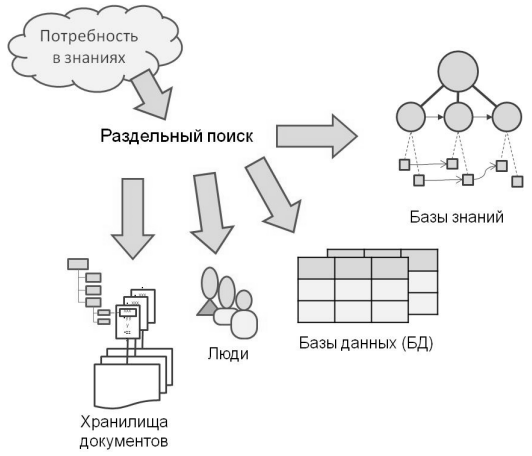


Рис. 5.5
В традиционных информационных системах потребность в знаниях нужно удовлетворять путем раздельного поиска по различным источникам

5.2. ОПРЕДЕЛЕНИЕ ОНТОЛОГИИ

Онтология (от древнегреч. онтос — сущее, логос — учение, понятие) — термин, определяющий учение о сущем, бытии, в отличие от гносеологии — учение о познании. В философском смысле, а этот термин заимствован из философии, онтология есть определенная система категорий, являющихся следствием определенных взглядов на мир.

В настоящее время термин «онтология» переместился в область информационных технологий, где был использован рядом исследовательских сообществ по искусственному интеллекту вначале в области инженерии знаний, в обработке естественных языков, а затем в представлении знаний [Gruber, 1993; Uschold, Jasper, 1999]. В конце 1990-х — начале 2000-х годов понятие онтологии также стало широко использоваться в таких областях, как интеллектуальная интеграция информации, поиск информации в Интернете и управление знаниями [Fensel, 2001; Gomez-Perez et al, 2006]. Позже онтологии стали рассматриваться в качестве ключевого элемента в проекте семантического веба — нового этапа развития сети WWW (Word Wide Web). Если существующая Web-сеть — это огромное множество документов, которые связаны перекрестными ссылками, то создаваемый семантический веб должен добавить к существующей сети множество онтологий и метаописаний знаний, содержащихся в документах Web-сети (включая стандарты и программные инструменты) [Berners-Lee et al, 2001; Domingue et al, 2011].

Понимание термина «онтология» зависит от контекста и целей его использования. В целом онтология состоит из иерархии понятий предметной области, связей между ними и законов, которые действуют в рамках этой модели. Онтология строится как сеть, состоящая из концептов и связей между ними. Связи могут быть различного типа (например, «является», «состоит из», «является исполнителем» и т. п.). Для выполнения роли общего языка будет включать в себя словарь (множество слов/лексических единиц) и набор определений понятий (формальных, на языке логической теории, или неформальных), задающих смысл элементов словаря. Связи между словарем (знаками) и семантикой ограничивают множество возможных интерпретаций знаков. Например, после установки связи между словом «Ягуар» и интенсиональным определением «хищное животное семейства кошек» мы исключаем из множества возможных интерпретаций этого слова машины и напитки. В результате онтологии позволяют правильно соотносить знаки, используемые людьми и компьютерами, с понятиями (семантическими моделями) и обозначаемыми объектами реального мира (рис.5.6).

Штудер и его коллеги дают более развернутое определение онтологии в информатике (computer science) [Studer et al, 1998] — это *формальная спецификация разделяемой концептуальной модели*, где

- под «концептуальной» моделью подразумевается абстрактная модель предметной области, описывающая систему понятий данной области,
- под «разделяемой» подразумевается согласованное понимание концептуальной модели определенным сообществом (группой людей),
- «спецификация» подразумевает описание системы понятий в явном виде,
- «формальная» подразумевает, что концептуальная модель задана на формализованном языке.

Под формальной моделью онтологии O будем понимать:

$$O = < C, P, R, A >,$$

где: C — конечное множество понятий (классов сущностей) предметной области; P — конечное множество свойств этих понятий (классов); R — конечное множество связей между понятиями (классами); A — множество аксиом, утверждений, построенных из этих понятий, их свойств и связей между ними.

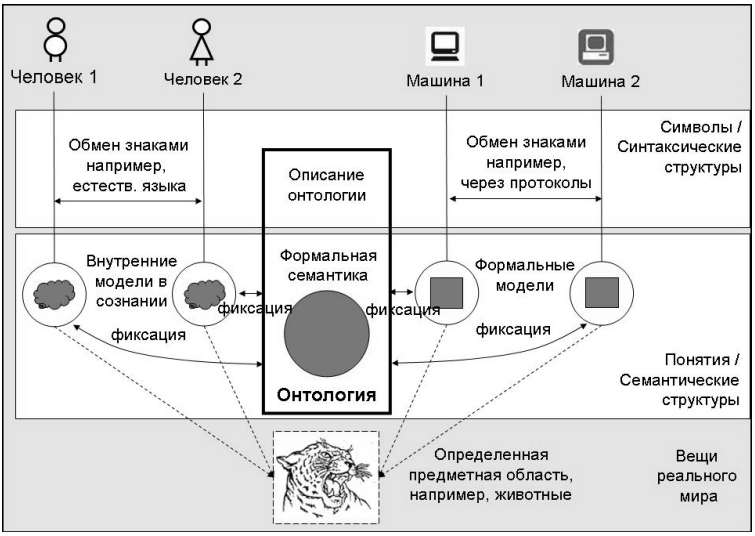


Рис. 5.6

Использование онтологий для коммуникаций между людьми и/или машинами [Studer, 2008]

Особо важным характеристическим свойством онтологии является наличие родовидовых отношений между классами объектов — таксономии классов. Например, в онтологии организации, предложенной компанией «Бизнес Инжиниринг Групп» (http://bigc.ru/), таксономия понятия «Деятельность» выглядит следующим образом (рис. 5.7):

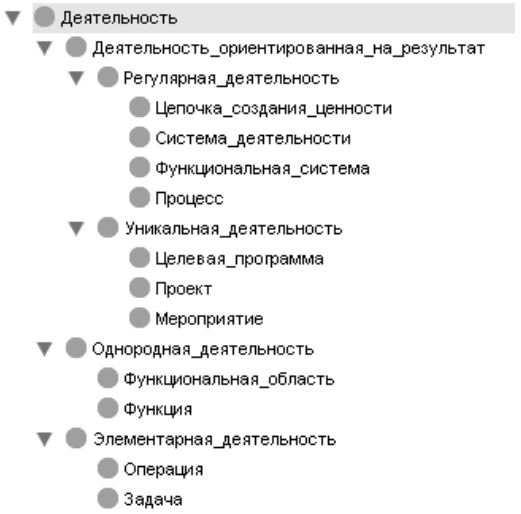


Рис. 5.7

Пример таксономии классов для понятия «Деятельность»

Именно благодаря родовидовым отношениям удается описать содержание понятия, дать его интенсиональное определение. Некоторые примеры таких определений для понятий из таксономии деятельности:

Мероприятие *— уникальная деятельность (родовое понятие), выполняемая для решения какой-либо задачи (характеристические признаки);*

Операция — *повторяющееся (характеристический признак) элементарное действие (родовое понятие), способность реализации которого поддерживается в организации;*

Проект — *уникальная деятельность (родовое понятие), направленная на получение определенного результата и/или достижения цели, состоящая из совокупности скоординированных и управляемых мероприятий с начальной и конечной датами, соответствующая конкретным требованиям, включающим ограничения по срокам, стоимости и ресурсам (характеристические признаки);*

Процесс (деловой процесс) — *регулярная деятельность (родовое понятие), представляющая собой повторяющуюся последовательность взаимосвязанных операций, направленная на получение определенного результата (характеристические признаки).*

Для пояснения роли онтологии в коммуникации можно расширить формальное определение онтологии, введя в него словарь:

$$O = \{C, R, A, L, F, G, H\},$$

где $L = L_C \cup L_R$ — словарь онтологии, содержащий набор лексических единиц (знаков) для понятий L_C и набор знаков для отношений L_R; C — набор понятий онтологии; R — отношения между понятиями онтологии; A — набор аксиом онтологии; F и G — функции ссылок такие, которые связывают наборы лексических единиц $\{Lj\} \subset L$ с наборами понятий и отношений, на которые они соответственно ссылаются в данной онтологии. При этом одна лексическая единица может ссылаться на несколько понятий или отношений, и одно понятие или отношение может ссылаться на несколько лексических единиц; H — фиксирует таксономический характер отношений (связей), при котором понятия онтологии связаны нерефлексивными, ациклическими, транзитивными отношениями $H \subset C \times C$. Выражение H (C1, C2) означает, что понятие C1 является подпонятием C2.

Кроме повышения эффективности коммуникаций, одним из ключевых преимуществ онтологии является возможность интеграции разнородной информации. Онтология задает структуру базы знаний организации, обеспечивая доступ к ее содержимому, предоставляет словарь для описания документов (контента) и профилей компетентности сотрудников, выступает общей схемой для интеграции различных баз данных (рис. 5.8). В результате, при возникновении потребности в знаниях сотруднику не требуется осуществлять поиск по разным источникам. Онтология позволит ему через один интерфейс получить доступ ко всем необходимым знаниям по требуемой теме, вне зависимости от формы представления знаний.

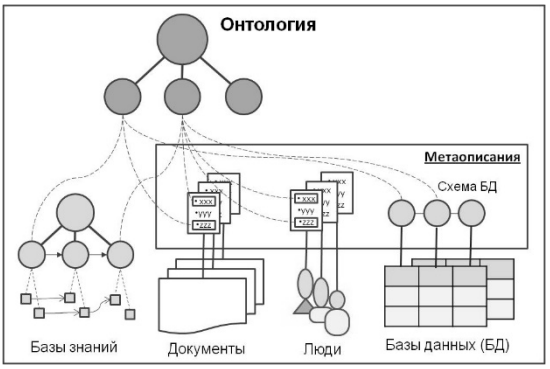


Рис. 5.8

Онтология позволяет интегрировать разнородные источники информации

5.3. КЛАССИФИКАЦИЯ И ПРИМЕРЫ ОНТОЛОГИЙ

Классификация по объекту концептуализации

В рамках классификации по объекту концептуализации выделяют четыре уровня (рис. 5.9): онтология представления, онтология верхнего уровня, онтология предметной области и прикладная онтология.



Рис. 5.9

Классификация онтологий по объекту концептуализации [Добров и др., 2006; Studer et al., 1998]

Начнем рассмотрение с предметных онтологий, с которыми, как правило, в первую очередь сталкиваются люди, знакомящиеся с онтологиями. Потом поднимемся к более абстрактным онтологиям (верхнего уровня), рассмотрим более конкретные (прикладные), а в завершение остановимся на онтологиях представления, которые, в основном, актуальны для специалистов в инженерии знаний и помогают в выборе языков представления знаний.

Онтология предметной области

Другое название — онтология домена (от англ. Domain) или доменная онтология. Область охвата таких онтологий ограничена одной предметной областью (доменом), например, авиация, медицина, культура, дистанционное обучение, организация. Онтология предметной области обобщает понятия, использующиеся в некоторых задачах домена, абстрагируясь от самих задач (так, онтология автомоби-

лей независима от любых особенностей конкретных марок машин). Во многих дисциплинах сейчас разрабатываются стандартные онтологии, которые могут использоваться экспертами по предметным областям для совместного использования и аннотирования информации в своей области.

Например, в области медицины созданы большие стандартные, структурированные словари, такие как SNOMED CT (Systematized Nomenclature of Medicine — Clinical Terms — систематизированная номенклатура медицины — клиническая терминология, http://www.ihtsdo.ru/snomed_ct/) [Cornet, de Keizer, 2008; Пашкина, Зарубина, 2013] и UMLS (Unified Medical Language System — система унифицированного медицинского языка) [Bodenreider, 2004], онтологии по многим разделам, например, генная онтология (gene ontology, http://geneontology.org/, рис. 5.10) [Ashburner, 2000].

Другим примером предметной онтологии является онтология исследовательской деятельности в области семантического веба — SWRC (Semantic Web Research Community) ontology [Sure et al., 2005]. Данная онтология включает в себя такие понятия, как человек, организация, публикация, научная группа, конференция, семинар и т. п. (рис. 5.11).

Активно создаются онтологии организации (enterprise ontologies), которые включают в себя понятия, характерные для любой организации, такие как, цель, процесс, проект, должностное лицо, подразделение, полномочия. Одной из самых первых и известных онтологий предприятия является разработка Учхолда, Кинга и других специалистов [Uschold et al, 1997]. Все термины их онтологии попадают в пять верхнеуровневых разделов, отражающих разные аспекты предприятия: метаонтология и время (Meta Ontology and Time), деятельность, план, способность и ресурс (Activity, Plan, Capability, Resource), организация (Organisation), стратегия (Strategy) и маркетинг (Marketing) (см. подробнее в главе 6).

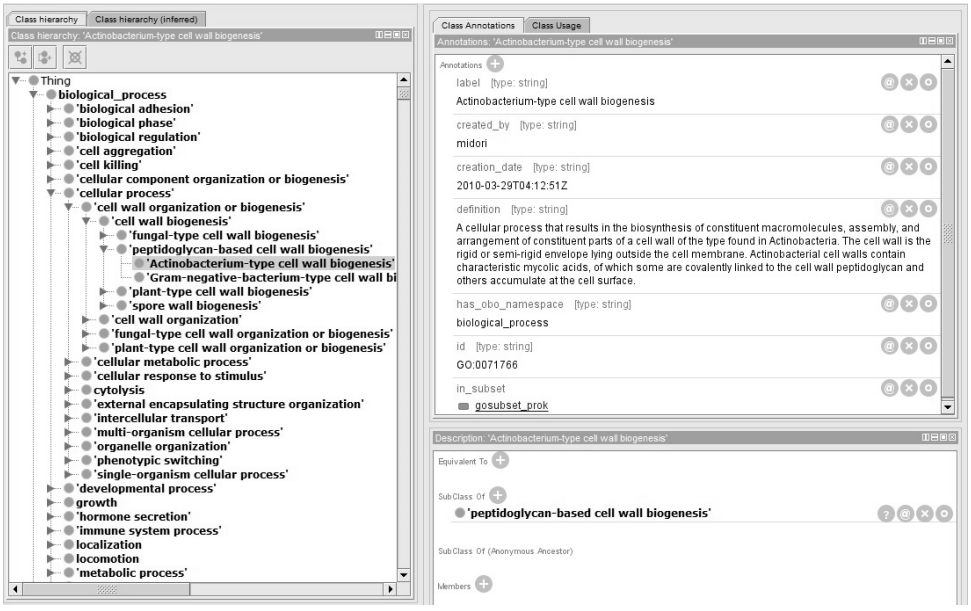


Рис. 5.10

Описание биологических процессов в генной онтологии

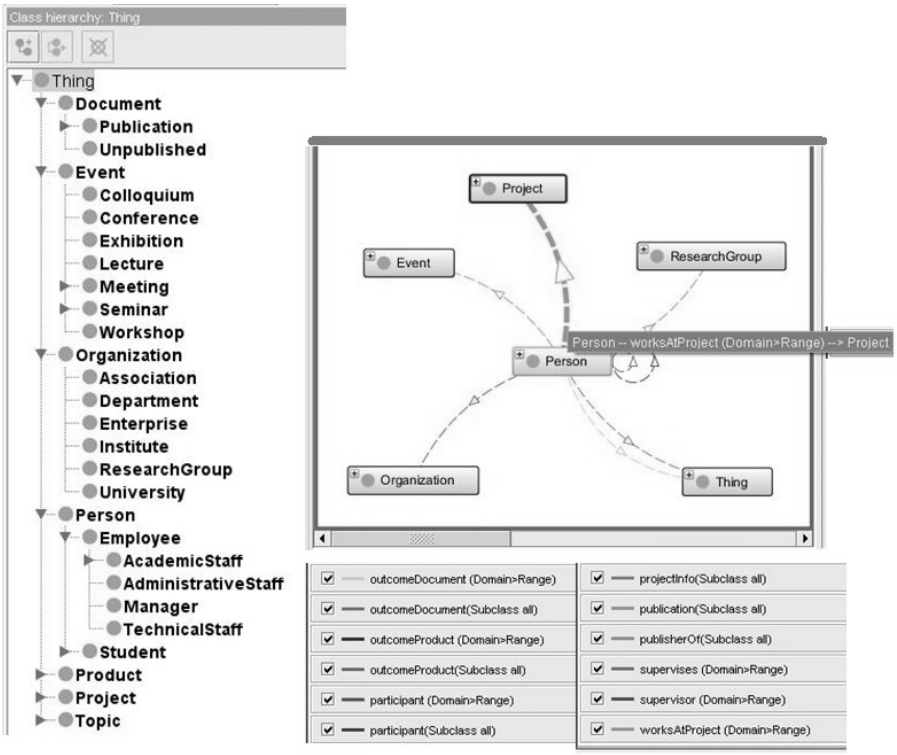


Рис. 5.11

Онтология научной деятельности SWRC (Semantic Web Research Community)

Онтология верхнего уровня

Ее назначение — в создании единой «правильной» онтологии, фиксирующей знания, общие для нескольких предметных областей, и в многократном использовании данной онтологии. Существует несколько крупных онтологий верхнего уровня: Cyc, DOLCE, SUMO, онтология Джона Совы (J. Sowa) и другие. Но в целом попытки создать онтологию верхнего уровня на все случаи жизни пока не привели к ожидаемым результатам. Многие онтологии верхнего уровня похожи друг на друга. Они содержат одни и те же концепты: сущность, явление, процесс, объект, роль, пространство, время, материя, событие, действие и т. п. Рассмотрим в качестве примера DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering) — первую из онтологий в библиотеке базовых онтологий проекта WonderWeb (http://wonderweb.man.ac.uk/), см. рис. 5.12.

Онтологию DOLCE предполагается применять в Semantic Web для согласования между интеллектуальными агентами, использующими разную терминологию. При этом онтология не претендует на звание универсальной, стандартной или общей. Основная цель разработчиков — создать модель, помогающую при сравнении и объяснении связей с другими онтологиями библиотеки онтологий WonderWeb, а также для выявления скрытых допущений, лежащих в основе существующих онтологий и лингвистических ресурсов, таких как WordNet. DOLCE имеет когнитивный уклон, поскольку фиксирует онтологические категории естественного языка и знания «здравого смысла».

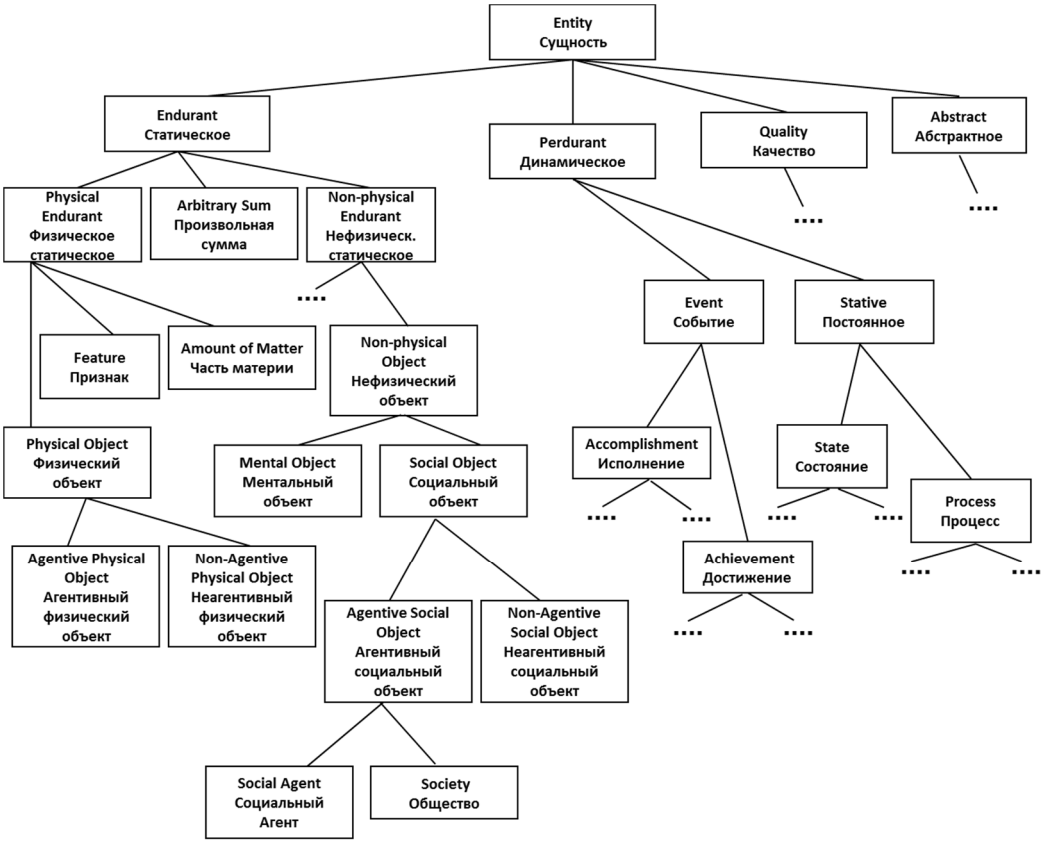


Рис. 5.12

Верхние уровни иерархии DOLCE, фрагмент [Пивоварова, 2010; Masolo et al., 2003]

В основу процесса проектирования легло фундаментальное философское разделение всех сущностей на универсалии (сущности, потенциально или реально имеющие экземпляры) и индивиды (или частности), которые не имеют и не могут иметь экземпляров. DOLCE — онтология индивидов, в том смысле, что область описания ограничена только ими. В качестве примера универсалии можно привести понятие «собака» (оно имеет множество экземпляров, конкретных примеров в окружающем мире). В отличие от этого понятия, понятие «время» скорее рассматривается как индивид (едва ли кому-то понадобится трактовать «время» как множество различных сущностей, конечно, если речь не идет о параллельных мирах).

Еще одна черта DOLCE (также заимствованная разработчиками из философии) — явное разделение на «статические» и «динамические» сущности. Различие между ними состоит в том, что «статические» сущности имеются в наличии целиком и неизменно в некотором фиксированном промежутке времени (например, стол или дом в течение периода своего существования). «Динамические» разворачиваются во времени и в каждый момент в некотором временном интервале они могут быть различными, по-разному себя проявлять, иметь разный состав (например: ураган или период раннего Ренессанса), однако при этом их идентичность сохраняется.

Такое разделение на «объект» и «процесс» весьма условно, и здесь прослеживается когнитивный уклон DOLCE. Оно привело к тому, что в онтологии определены два типа отношения ЧАСТЬ-ЦЕЛОЕ. Первое никак не зависит от времени, второе имеет временной индекс, определяющий, в каких временных рамках отношение действует. Подобное «раздвоение» наблюдается и для отношения «КАЧЕСТВО — ОБЛАДАТЕЛЬ КАЧЕСТВА». Другие базовые отношения онтологии: УЧАСТНИК-ПРОЦЕСС, КОМПОНЕНТ-ЦЕЛОЕ (компонент входит в состав целого) и отношение зависимости имеют временной индекс. Для сравнения, в онтологии OpenCyc нет явного деления на «Статическое» и «Динамическое». Поэтому среди множества отношений в разделе «Части объектов» нет отношения, учитывающего временной аспект: возможное непостоянство данного отношения.

Прикладная онтология

Назначение такой онтологии в том, чтобы описать концептуальную модель конкретной задачи или приложения. Прикладные онтологии содержат необходимые в конкретном приложении понятия, заимствованные из одной или нескольких онтологий предметных областей, а также понятия, тесно связанные с решаемыми задачами.

Например, онтология организации для задач организационного проектирования может иметь свои особенности. Продолжим рассматривать онтологию, предложенную компанией «Бизнес Инжиниринг Групп». Для коммерческих организаций состав задач оргпроектирования, для решения которых используется онтология, следующий:

1. Стратегическая фокусировка бизнес-системы: идентификация стратегий, «Контроль качества» стратегий, идентификация системы стратегических показателей.

2. Формирование оптимальной организационно-функциональной структуры: идентификация существующей организационно-функциональной структуры, анализ существующей организационно-функциональной структуры, разработка функционала и системы процессов бизнес-системы, разработка/оптимизация оргструктуры и распределения ответственности, выпуск и актуализация организационной документации по ОФМ.

3. Разработка, реинжиниринг и непрерывное совершенствование процессов: идентификация и описание процессов, анализ процессов, оптимизация процессов, выпуск и поддержание в актуализированном состоянии регламентов исполнения процессов.

Задачи, методы и, особенно, правила принятия решений требуют более детальных классификаций понятий (рис. 5.13), а также приводят к возникновению в онтологии классов-ролей, принадлежность к которым зависит от рассматриваемой задачи/метода (табл. 5.1).

Данная таблица иллюстрирует, что в контексте стратегического SWOT-анализа[1] какие-либо финансовые ресурсы организации могут исполнять роль «Силы» организации, а например, позиция на рынке — роль «Слабости».

[1] Один из методов стратегического анализа деятельности организации. Подробнее рассмотрен в разделе «Процесс управления знаниями».

Таблица 5.1

Пример классов-ролей в прикладной онтологии

Контекст задачи и метода	Контекстно-зависимые классы-роли	Классы-типы — исполнители ролей
Прикладной контекст: 1. Анализ деятельности 1.1. Стратегический SWOT-анализ Позиционный контекст: Вид деятельности	Силы Слабости	Ресурсы организации (финансовые, материальные, компетенции) Позиция на рынке Ценностное предложение
	Возможности Угрозы	Текущее состояние и тенденции внешней среды (политика, экономика, технологии, общество)

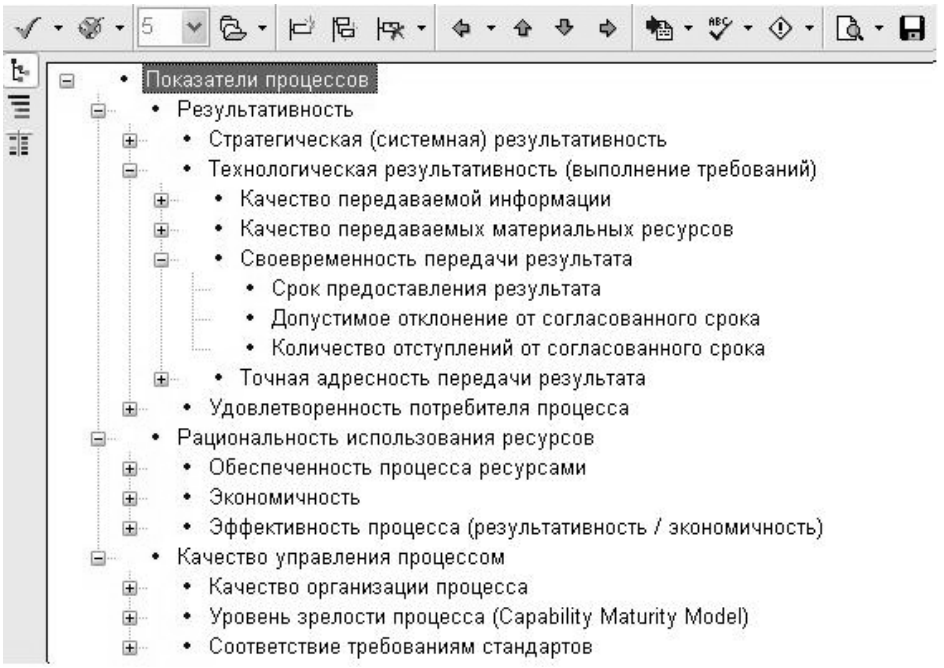


Рис. 5.13

Детализация таксономии «Показателей процессов» в прикладной онтологии организации

Онтология представления

Цель ее создания — описать область представления знаний, создать язык для спецификации других онтологий более низких уровней. Пример: описание понятий языка OWL средствами RDF/RDFS (рис. 5.14). В данном описании определяются такие понятия, как «класс», «отношение», «ограничение на значение свойства», «домен», «диапазон» и т. п.

Классификация онтологий по степени формальности

Данная классификация онтологий возникла в 2003 г. [McGuinness, 2003], когда в литературе накопилось внушительное количество упоминаний онтологий с различной интерпретацией данного понятия. На русском языке данная классификация была описана в работе [Добров и др., 2006].

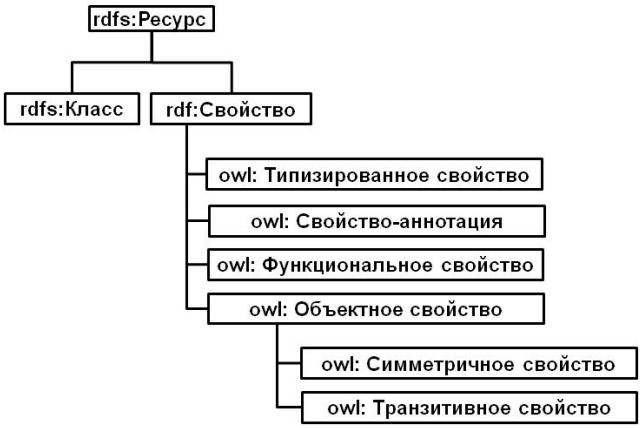


Рис. 5.14
Онтология представления для языка OWL

Обычно люди и компьютерные программы имеют некоторое представление о значениях терминов. Онтологии могут быть применены, чтобы предоставить конкретную спецификацию имен терминов и значений терминов. В рамках этого понимания (где онтология является спецификацией концептуальной модели — *концептуализации*) существует простор для вариаций. Отдельные виды онтологий могут быть представлены как точки на спектре в зависимости от деталей их реализации.

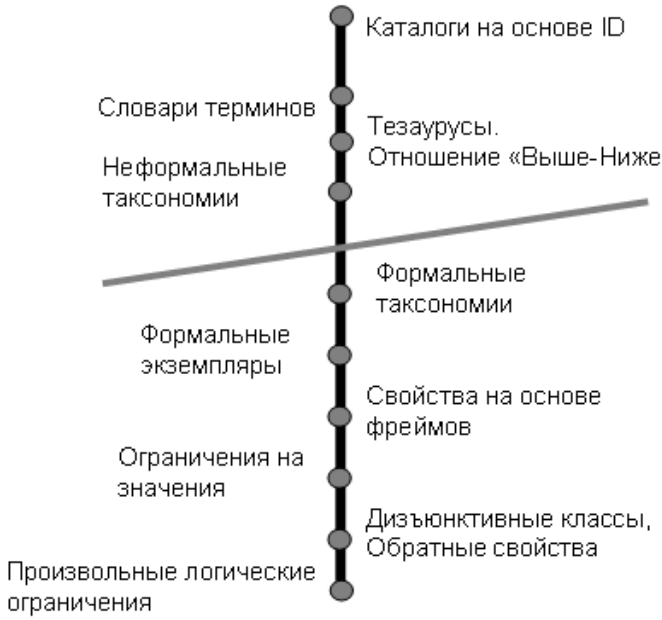


Рис. 5.15
Спектр онтологий. Косая черта разделяет системы, предоставляющие «человеко-понятные» (выше черты) и «машино-понятные» (ниже черты) описания [Добров и др., 2006; McGuinness, 2003]

На рисунке 5.15 изображен так называемый спектр онтологий по степени формальности представления, использованию тех или иных синтаксических конструкций. Каждая точка соответствует наличию некоторых ключевых структур в онтологии, отличающих ее от других точек на спектре. Косая черта условно отделяет онтологии от других ресурсов, имеющих онтологический характер.

Первой точке на спектре соответствует ***контролируемый словарь***, т. е. конечный список терминов (простейшим примером является каталог на основе идентификаторов). Каталоги представляют точную (не многозначную) интерпретацию терминов. Например, каждый раз, ссылаясь на термин «машина», мы будем использовать одно и то же значение (соответствующее некоторому идентификатору в словаре), вне зависимости от того, о чем идет речь в контексте: о «стиральной машине», «автомобиле» или «государственной машине».

Другой спецификацией онтологии может быть ***глоссарий***, представляющий собой список терминов с их значениями. Значения описываются в виде комментариев (определений) на естественном языке. Это дает больше информации, поскольку люди могут прочесть такой комментарий (определение) и понять смысл термина. Интерпретации терминов могут быть многозначными. Глоссарии непригодны для автоматической обработки программными агентами, но можно, как и ранее, присвоить терминам идентификаторы.

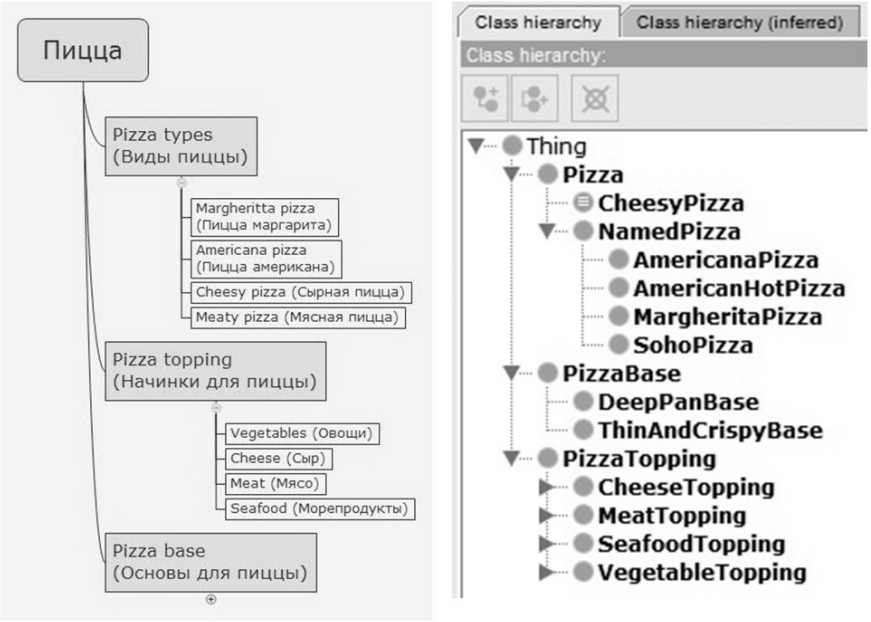
Тезаурусы несут дополнительную семантику, определяя связи между терминами. Отношения, свойственные для тезаурусов: синонимия, *иерархическое отношение* и ассоциация.

Ранние ***иерархии терминов*** (***неформальные таксономии***), появившиеся в Сети, определяли термины через операции обобщения и уточнения. Yahoo, например, ввела небольшое число категорий верхнего уровня, таких, как «предметы одежды». Затем «платье» определялось как вид (женской) одежды. Явная иерархия Yahoo не соответствовала в точности формальным свойствам иерархического отношения ПОДКЛАСС-КЛАСС. В таких иерархиях может встретиться ситуация, в которой экземпляр класса-потомка не является экземпляром класса-предка. Например, общая категория «Пицца» имеет подкатегорию «Начинки для пиццы», а эта категория, в свою очередь, включает подкатегории «Овощи» и «Мясо». Ясно, что овощи, например помидоры, не являются пиццой. Здесь не выполняется важное свойство отношения ПОДКЛАСС-КЛАСС — транзитивность.

Далее следует точка ***формальные таксономии***. Эта разновидность онтологий включает точное определение отношения ПОДКЛАСС-КЛАСС (обозначаемого как isA или AKO — A Kind Of). В таких системах строго соблюдается *транзитивность* отношения ПОДКЛАСС-КЛАСС: если А является подклассом класса В, то каждый подкласс класса А также является подклассом класса В. Строгая иерархия классов необходима при использовании наследования для процедуры логического вывода.

Примеры неформальной и формальной таксономий представлены на рисунке 5.16.

Следующая точка спектра — наличие в онтологической системе ***формального отношения ЭКЗЕМПЛЯР-КЛАСС*** (обозначаемого как isInstanceOf). Некоторые классификации включают только имена классов, другие содержат на нижнем уровне экземпляры (индивиды). Для отношения ЭКЗЕМПЛЯР-КЛАСС выполняется так называемая «наследуемость»: если А является подклассом класса В, то каждый экземпляр класса А также является экземпляром класса В. Поэтому в приведенном выше примере «Овощи» не могут быть помещены в иерархии ниже категории «Пицца» или стать экземпляром этой категории.



А) Неформальная таксономия

Б) Формальная таксономия

Рис. 5.16

Примеры неформальной и формальной таксономий пиццы

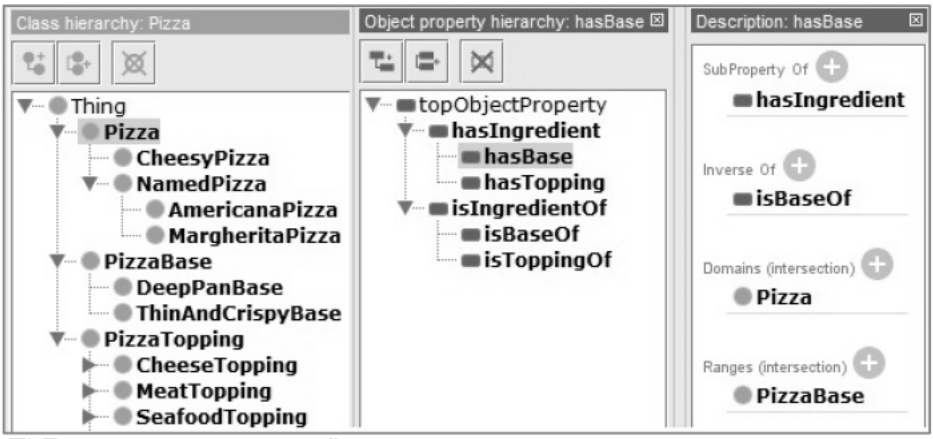
Далее у структурных элементов ***появляются свойства*** (иногда их называют *слотами)*. Здесь *классы* (иногда их называют *фреймами*) могут иметь информацию о *свойствах (слотах*). Например, класс «Пицца» может иметь свойства «калорийность», «имеет начинку». Свойства бывают особенно полезными, когда они определены на верхних уровнях иерархии и наследуются подклассами. Так, в потребительской иерархии класс «продукт» может иметь свойство «цена», которое получат все его подклассы.

Большей выразительностью обладают онтологии, включающие ***ограничения на область значений свойств*** (см. рис. 5.17). Значения свойств берутся из некоторого предопределенного множества (целые числа, символьные константы) или из подмножества концептов онтологии (множество экземпляров данного класса, множество классов). Можно ввести дополнительные ограничения на то, что может заполнять свойство. Например, для свойства «имеет начинку» класса «Пицца» значения могут быть ограничены экземплярами класса «Начинка пиццы». Легко увидеть проблемы, которые могут возникнуть в этом случае при использовании нестрогой таксономии. Если «Овощи» — потомок класса «Пицца», то он унаследует свойство «имеет начинку» вместе с ограничением на его значения («Начинка пиццы»).

В целом с необходимостью описывать более сложные факты выразительные средства онтологии (и ее структура) усложняются. Классы и свойства в онтологии могут иметь дополнительные свойства, полезные для целей дальнейшей компьютерной обработки. Классы, например, можно объявить дизъюнктивными (непересекающимися) — не имеющими общих экземпляров, это в дальнейшем может

быть использовано, например, для проверки непротиворечивости онтологии. Для свойств могут быть указаны обратные свойства (например, «имеет начинку» — «быть начинкой в»), транзитивность, симметричность и т. п. Также может потребоваться заполнить значение какого-либо свойства экземпляра, используя математическое выражение, основанное на значениях других свойств данного экземпляра или значениях свойств других экземпляров.

А) В редакторе онтологий



Б) В виде концептуальной карты

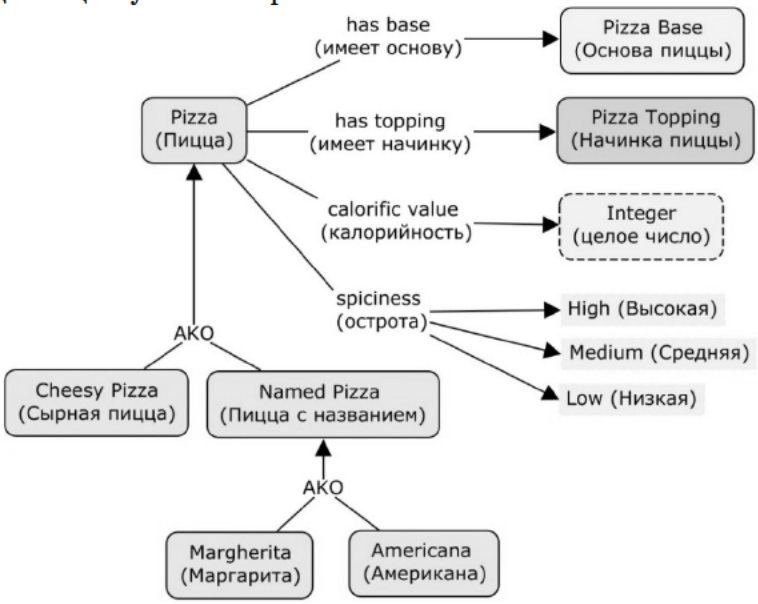


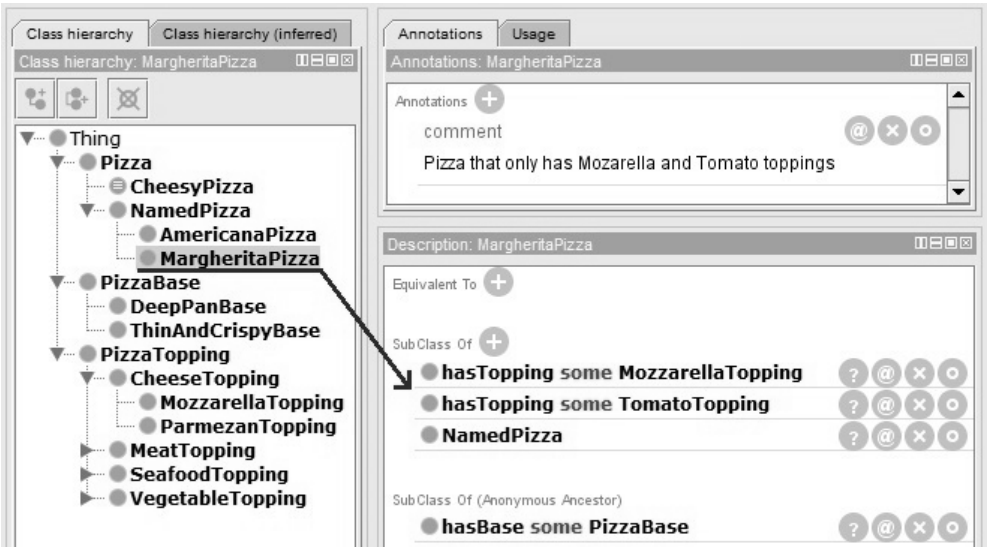
Рис. 5.17

Пример онтологии пиццы со свойствами и ограничениями на их значения

Специализированные языки описания онтологий позволяют делать ***произвольные логические утверждения о понятиях*** (***аксиомы***). С их помощью

можно задавать формальные определения для понятий (классов), которые в дальнейшем могут использоваться для логического вывода, например, для автоматической классификации и отнесения объекта к определеному классу. На рисунке 5.18 представлены необходимые условия принадлежности к классу «Пицца Маргарита» (рис. 5.18, А) и достаточные условия принадлежности к классу «Сырная пицца» (рис. 5.18, Б). Достаточные условия задают определение класса.

А) Если индивид является MargheritaPizza, то он является NamedPizza и имеет начинку из Mozzarella и Tomato (Необходимые условия принадлежности к классу)



Б) Если индивид является членом класса Pizza и связан по крайней мере с одним членом класса CheeseTopping, то этих условий достаточно, чтобы определить, что индивид должен быть членом класса CheesyPizza (Достаточные условия принадлежности к классу)

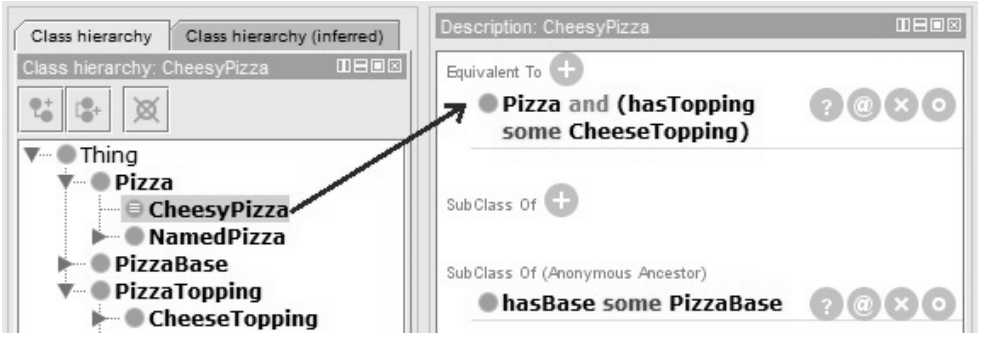


Рис. 5.18

Пример онтологии пиццы с аксиомами (логическими утверждениями о классах)

5.4. ТИПЫ ОТНОШЕНИЙ В ОНТОЛОГИЯХ

Некоторые типы отношений между понятиями (классами) встречаются почти во всех онтологиях и составляют основу онтологического инжиниринга. В данном разделе эти типы отношений и связанные с ними проблемные вопросы будут рассмотрены подробно. Существенная часть материала данного раздела заимствована из [Лукашевич, 2011].

Отношения класс-подкласс и таксономия

Отношение между классами и подклассами понятий может носить разное название в зависимости от терминологических традиций в области использования ресурса: таксономическое отношение, родовидовое отношение, IS-a отношение, AKO (акроним от A Kind Of — является видом), отношение гипонимии и гиперонимии (в лексических ресурсах). Далее в данной главе мы будем ссылаться на это отношение как отношение *класс-подкласс.*

Концептуальная структура (фрагмент онтологии), построенная на основе отношения класс-подкласс называется таксономией. Этот термин (от греч. расположение по порядку + закон) в науке и математике обозначает классификацию и систематизацию сложноорганизованных областей действительности, имеющих обычно иерархическое строение (органический мир, объекты географии...)[1]. В качестве синонима таксономии может выступать родовидовая классификация понятий, в которой вышестоящий элемент называется родом, а его потомок — видом. Пример таксономии «Автомобиль» представлен на рисунке 5.19.

Отношения *класс-подкласс* обладают такими важными свойствами, как транзитивность и наследование, на которых основывается логический вывод во многих компьютерных системах [Осипов, 1997].

Пусть T (X, Y) — отношение *класс-подкласс* между понятиями X и Y, Y является подклассом (видом) X, R (X, Z) — это произвольное отношение между понятиями X и Z. Тогда свойства отношения *класс-подкласс* могут быть записаны следующим образом:

T (X, Y) ∧ T (Y, Z) → T (X, Z) — транзитивность отношения *класс-подкласс,*

T (X, Y) ∧ R (X, Z) → R (Y, Z) — свойство наследования по отношению класс-подкласс: «Если Y является подклассом X и X связан неким отношением с Z, то Y тоже связан этим отношением с Z».

Исторически наиболее ранними принципами установления отношений *класс-подкласс* (родовидовых отношений), используемых и в работах по искусственному интеллекту, и в компьютерной лингвистике, было использование ставших классическими диагностических высказываний [Cruse, 1986]. Например, если понятие Y является видом понятия X, то можно сказать, что «Y — это X», «Y, Z и другие X», «к числу X относятся Y».

Однако позже выяснилось, что одни и те же выражения естественного языка (и в частности, применяемые диагностические тесты) могут с онтологической точки зрения соответствовать значительно различающимся отношениям между сущностями внешнего мира, в том числе обладающими совсем другими свойствами [Guarino, 1998]. Поэтому многие методические руководства по разработке понятийных ресурсов рекомендуют осуществлять дополнительные проверки для устанавливаемого отношения *класс-подкласс.*

---

[1] Новое определение из современной англоязычной литературы по информационным системам: «Таксономия — структура, отражающая способ классификации документов по группам для облегчения поиска, обучения и размещения (Gartner, Delphi Group...)», см. [Lambe, 2007; Stewart, 2011]. Как правило, в данном смысле используется среди практиков.

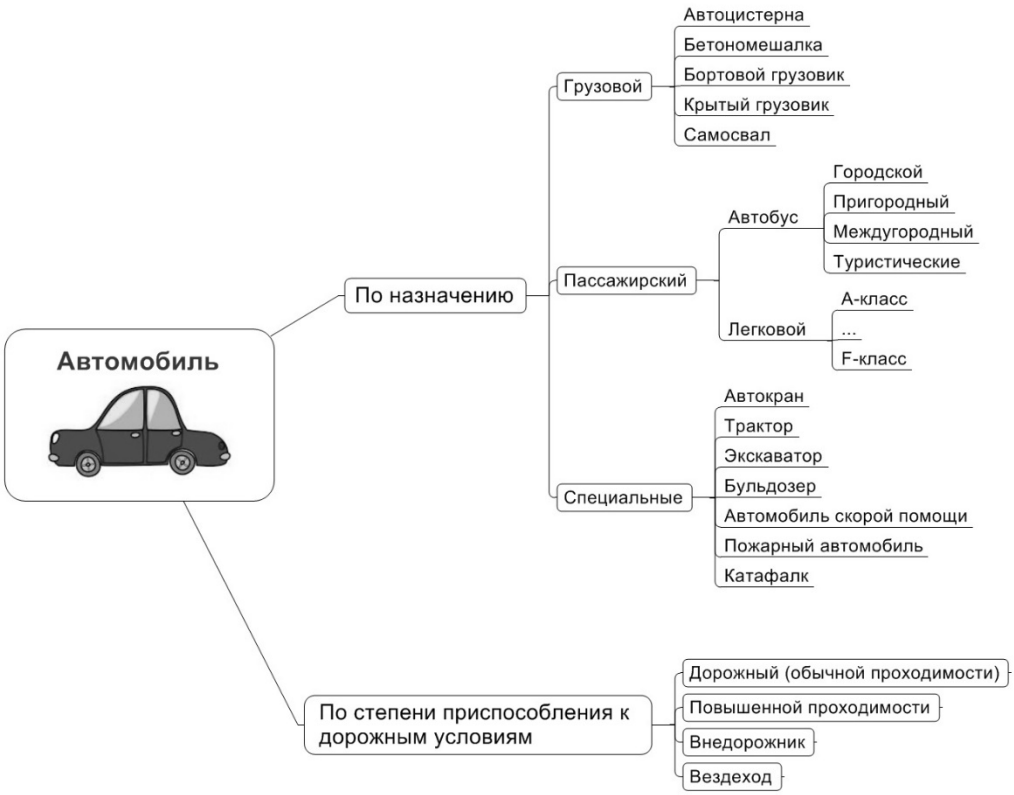


Рис. 5.19
Таксономия автомобиля

Наиболее распространенной рекомендацией для проверки правильности установления отношений *класс-подкласс* является ответ на вопрос: если объект является экземпляром одного класса, то будет ли он обязательно (т. е. по определению) экземпляром некоторого другого класса: *Если класс A — надкласс класса B, то каждый экземпляр класса B также является экземпляром A [Noy, McGuinness, 2001; Gomez-Perez et al, 2004].*

Подробнее на данную тему см. главу 6 (Установление отношений в онтологиях. Отношение класс-подкласс) и главу 7 (Описание ролей в компьютерных ресурсах) в [Лукашевич, 2011].

Отношение часть-целое и партономия

Отношение *часть-целое* играет существенную роль во многих предметных областях. Для компьютерных приложений особое значение представляет такое свойство отношения *часть-целое,* как транзитивность, на основе которой может строиться многошаговый логический вывод. Для некоторых типов отношений *часть-целое* может выполняться наследование свойств и операций от части к целому и от целого к части, что также может стать базой для логического вывода. Концептуальная структура или фрагмент онтологии, основанный на отношении часть-целое называется *партономией*. На рисунке 5.20 представлены части автомобиля — партономия автомобиля.

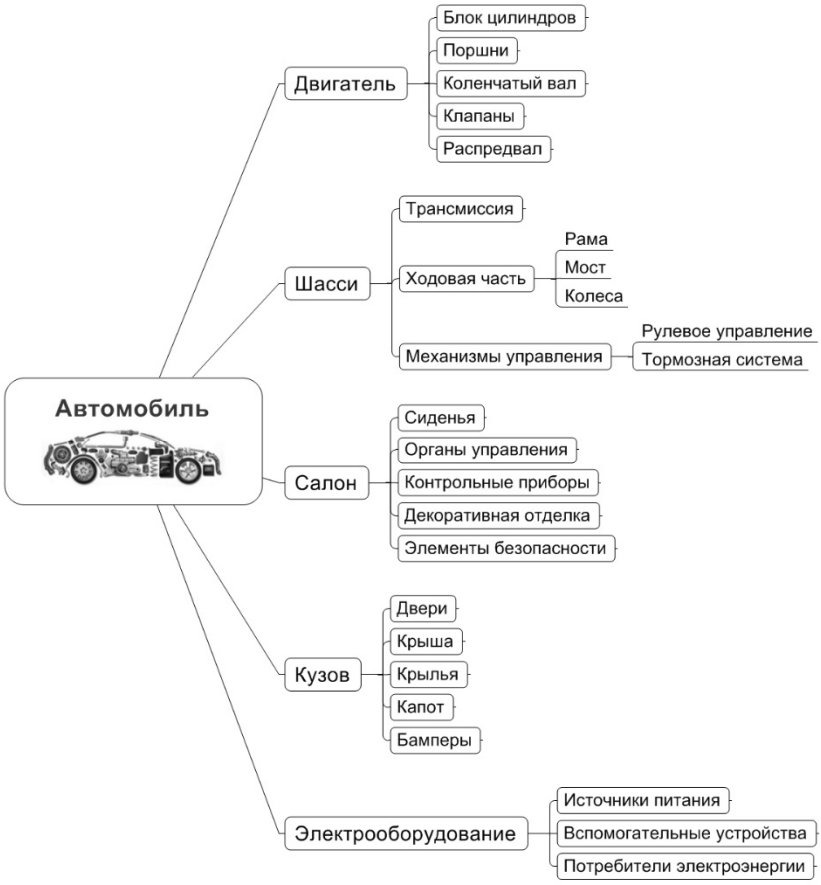


Рис. 5.20
Партономия автомобиля

Специфической особенностью этого отношения является его разнообразие — отношение может быть установлено между сущностями различных семантических типов: физическими объектами, процессами и действиями, географическими регионами, свойствами и состояниями, коллекциями и множествами, абстрактными сущностями, такими как числа. Изучающая это отношение отрасль философии — мереология — не накладывает никаких ограничений на типы части и целого, лишь постулируя три основных аксиомы [Varzi, 2006; Gangemi и др., 200l] (см. далее).

Другой стороной явного отсутствия ограничений на участников отношения *часть-целое* является то, что его описание осложнено наличием достаточно разнообразных подвидов этого отношения, а также существованием совокупности отношений, которые могут быть ошибочно отождествлены с этим отношением. Разработчики компьютерных ресурсов в конкретных предметных областях стал-

киваются с достаточно сложной ситуацией такого описания отношений *часть-целое*, которое сохранило бы его важнейшие свойства.

В данном разделе мы рассмотрим основные проблемы описания отношения *часть-целое*, его определения в рамках философии и лингвистики, а также примеры его моделирования в конкретных компьютерных ресурсах.

Для отношения часть-целое обычно постулируются три аксиомы [Varzi, 1996; Varzi, 1998):

1) Транзитивность. Части частей являются частями целого:
Если A – часть B и B – часть C, то A – это часть C.

2) Рефлексивность. Все является частью самого себя:
A – это часть A.

3) Несимметричность: ничто не является частью своих частей:
Если A – часть B и A не совпадает с B, то B – это не часть A
[Varzi, 2006, Gangemi и др., 200l].

Поскольку отношение обладает свойством рефлексивности, выделяются еще собственно-части, т. е. части, не равные своему целому.

В лингвистике для определения отношения *часть-целое* широко используются лингвистические тесты, т. е. некоторые заданные предложения, в которые подставляются анализируемые сущности. При этом *часть* обычно называется меронимом, а *целое* — холонимом. Естественным тестом для определения меронимии является предложение *X – это часть Y,* которое должно звучать нормально для X и У, интерпретируемых как родовые понятия, *палец* — это часть руки, страница — это часть книги *[Cruse, 1986].*

Другими возможными лингвистическими тестами для отношения *часть–целое* являются следующие:

Части У включают X, Z и т. д.: Части слова включают корень, приставку, суффикс.

X и другие части У: Суффикс и другие части слова.

Однако многие авторы отмечают, что если применять лингвистические тесты, то возникают серьезные проблемы с транзитивностью отношения *часть-целое.* Рассмотрим, например, следующую совокупность утверждений: *рука — это часть дирижера*, *дирижер — это часть оркестра,* но странно было бы сказать, что *рука — это часть оркестра.*

Многие исследователи указывают, что отношение *часть-целое* представляет собой скорее совокупность несколько отличающихся отношений, чем четкое отделяемое отношение:

- наиболее центральный тип этого отношения представляют физические объекты;
- сущности, длящиеся во времени, могут иметь части, которые называются стадиями, фазами или этапами;
- такие сущности, как группы *(племя, команда),* классы (*пролетариат, буржуазия)* и коллекции *(куча),* состоят в отношении меронимии со своими элементами;
- если и *целое,* и *часть* являются неисчислимыми, то можно говорить об отношении ингредиентов, например, *заварной крем и молоко;*
- если *целое* — исчислимое, а часть — неисчислимое, то это так называемое отношение объект – материал: *бокал – стекло*;

- если *целое* неисчислимое, а *часть* — исчислимое, то говорят об отношении вещество – частица: *песок – песчинка, снег – снежинка, дождь – капля* [Cruse, 1986];
- не изменяющиеся во времени состояния и свойства могут быть описаны как имеющие части: *самоконтроль — это часть зрелости*; другими периферийными меронимами могут быть названы характеристики событий, состояний. На них возможна ссылка как на части. Однако более нормально их называть признаками. *Рождественский пудинг — это часть (признак) Рождества* [Cruse, 1986].

Предложены различные классификации отношений *часть-целое.*

Основная идея авторов работы [Winston et al, 1987] состоит в том, чтобы классифицировать различные способы, которыми части соотносятся к своему целому, введением шести различных типов отношений меронимии, различаемых на основе признаков функциональности, гомеомерности и отделимости:

- *функциональные части* ограничены своей функцией в их пространственном или временном положении. К примеру, ручка чашки может быть расположена в ограниченном числе позиций, если она должна функционировать как ручка;
- *гомеомерные части* представляют собой объект того же типа, что и их целые: к примеру, кусок-пирог, тогда как негомеомерные части отличны от своих целых (дерево-лес);
- *отделимые части* могут, в принципе, быть отделенными от целого, к примеру, ручка-чашка, тогда как неотделимые не могут (сталь-велосипед).

На основе комбинации этих трех признаков можно выделить следующие типы отношений *часть-целое*.

1. *Компонент/Интегральный объект:* Интегральные объекты характеризуются своей структурой, тогда как их компоненты отделимы и имеют специальную функциональность. Например: *Колеса — это части автомобиля, фонология — это часть лингвистики.*

2. *Член/Коллекция:* Члены не играют никакой функциональной роли по отношению к своему целому, они отделимы*: Дерево — часть леса.*

3. *Порция/Масса:* Целое рассматривается как гомогенный агрегат и его порции подобны целому: *Кусок — это часть пирога.*

4. *Материал/Объект:* Материал, из которого сделан объект, не отделим от этого объекта, не имеет функциональной роли и негомеомерный: *Сталь — часть велосипеда.*

5. Фаза/Деятельность: Фаза как компонент имеет функциональную роль, но неотделима: ложка — часть процесса еды, глотание — часть процесса еды.

6. *Место/местность*: часть гомеомерна, так как каждая часть пространственного региона подобна целому региону, но неотделима: *Оазис — это часть пустыни.*

Классификация отношений *часть-целое* на основе других принципов приводится в работе [Gerstl, Pribennow, 1996].

В философии постулируется транзитивность отношения *часть-целое.* Это свойство отношения является существенным для многих компьютерных приложений. Однако разные авторы приводят многочисленные примеры нарушения транзитивности этого отношения [Cruse, 1986; Winston и др., 1987; Johansson, 2004], например:

Несомненно, что ручка двери является частью двери, дверь является частью дома, *но странно говорить, что* ручка двери является частью дома.

Несомненно, что ветка является частью дерева, дерево является частью леса, *но странно говорить, что* ветка является частью леса.

Рассматривая разные виды отношения *часть-целое*, авторы обычно подчеркивают, что проблемы с транзитивностью связаны со смешением разных видов отношений *часть-целое.* В работе [Winston и др., 1987] проблемы с транзитивностью объясняются следующим образом: пока используется один тип отношения, *часть-целое* всегда транзитивно. Однако, когда смешиваются различные отношения меронимии, то возникает проблема с транзитивностью.

В работе [Cruse, 1986] подчеркивается, что правильно сформированная иерархия состоит из элементов одного и того же типа. Так, если элемент меронимии — физический объект, то и все другие элементы меронимии должны быть такими же (например, вес тела не должен фигурировать среди его частей). Если один элемент является географической областью, то и другие должны быть такими же; если один элемент — абстрактное существительное, то и другие должны быть такими же.

В работе [Motschnig-Pitrik, Kaasboll, 1999] предлагается выделить те отношения *часть – целое*, которые, комбинируясь, дают приемлемые результаты транзитивности, и отделить те отношения *часть-целое*, которые могут привести к ошибочным транзитивным заключениям. Если моделировать такие отношения, как *член/коллекция, материал/объект* отношениями, отличными от отношений *часть – целое*, то авторы утверждают, что оставшиеся типы отношений демонстрируют транзитивное поведение, даже если комбинируются произвольным образом. Таким образом, группа отношений компонент/объект, порция/масса, фаза/деятельность, место/местность может быть названа базовыми отношениями *часть-целое.* В рамках любой комбинации базовых отношений *часть-целое* действует правило транзитивности, независимо от комбинации конкретных видов отношений.

Подробнее на данную тему см. главу 8 (Отношение часть-целое) в [Лукашевич, 2011].

Кроме рассмотренных выше базовых отношений для разных областей и для разных задач могут быть актуальны свои типы отношений. Например, исследователи в области биомедицинских знаний выделяют следующие основные типы отношений в биомедицинских онтологиях: базовые (is_a, part_of), пространственные (located_in, contained_in, adjacent_to), темпоральные (transformation_of, derives_from, preceded_by), отношения участия (has_participant, has_agent) [Smith B. et al, 2005]. Идентификация данных отношения позволила четко прописать смысл каждого отношения, его свойства и области применения, что, в свою очередь, позволило повысить качество и согласованность онтологий в области медицины и биологии.

5.5. МЕТОДЫ РАЗРАБОТКИ ОНТОЛОГИЙ

Методология разработки онтологий для новичков

В первую очередь, рассмотрим базовую методологию создания онтологий, предложенную Натальей Ной и Деборой МакГиннесс для начинающих инженеров по знаниям [Noy, McGuinness, 2001]. Данная методология состоит из 7 шагов.

Шаг 1. Определение области и масштаба онтологии.

Шаг 2. Рассмотрение вариантов повторного использования существующих онтологий.

Шаг 3. Перечисление важных терминов в онтологии.

Шаг 4. Определение классов и иерархии классов.

Шаг 5. Определение свойств классов (слотов).

Шаг 6. Определение ограничений на значения свойств (слотов).

Шаг 7. Создание экземпляров.

Рассмотрим данные шаги подробнее.

Шаг 1. Определение области и масштаба онтологии

Н. Ной и Д. МакГиннесс предлагают начать разработку онтологии с определения ее области и масштаба. Т. е. с ответа на несколько основных вопросов:

1. Какую область будет охватывать онтология?
2. Для чего мы собираемся использовать онтологию?
3. На какие типы вопросов должна давать ответы информация в онтологии?
4. Кто будет использовать и поддерживать онтологию?

Ответы на эти вопросы могут измениться во время процесса проектирования онтологии, но в любой заданный момент времени они помогают ограничить масштаб модели.

Предположим необходимо разработать онтологию вина и еды. Область охвата онтологии — представление еды и вин. Мы собираемся использовать эту онтологию для приложений, которые будут предлагать хорошие сочетания вин и еды. Конечно, в нашу онтологию будут включены понятия, описывающие различные типы вин, основные виды еды, понятие хорошего и плохого сочетания вина и еды. В то же время, маловероятно, что онтология будет включать понятия для управления инвентарем на винном заводе или служащими в ресторане, даже хотя эти понятия отчасти связаны с понятиями вина и еды.

Если онтология, которую мы проектируем, будет использоваться для помощи при обработке естественного языка статей в журналах о винах, то, возможно, понадобится включить в онтологию синонимы понятий и информацию о частях речи. Если онтология будет использоваться для того, чтобы помочь посетителям ресторана решить, какое вино заказать, нам нужно будет включить информацию о розничных ценах. Если она будет использоваться для помощи покупателям вина в создании запасов в винном погребе, то могут понадобиться сведения об оптовых ценах и о наличии вин.

Один из способов определить масштаб онтологии — это набросать список вопросов, на которые должна ответить база знаний, основанная на онтологии, т. е. **вопросы для проверки компетентности**. Эти вопросы будут служить лакмусовой бумажкой: Содержит ли онтология достаточно информации для ответа на эти типы вопросов? Требуется ли для ответов определенный уровень детализации онтологии? Эти вопросы для проверки компетентности задают ориентиры и не обязаны быть исчерпывающими.

В области вина и еды возможны следующие вопросы для проверки компетентности:

1. Какие характеристики вина мне следует учитывать при выборе вина?
2. Вино Рислинг красное или белое?
3. Хорошо ли сочетается Каберне Совиньон с морскими продуктами?
4. Какое вино лучше всего подойдет к жареному мясу?
5. Какие характеристики вина влияют на его сочетаемость с блюдом?

Судя по этому списку вопросов, онтология будет включать информацию о различных характеристиках вина и типах вин, классификациях еды, которые нужно учесть при выборе подходящего вина, рекомендуемых сочетаниях вина и еды.

Шаг 2. Рассмотрение вариантов повторного использования существующих онтологий

Почти всегда стоит учесть, что сделал кто-то еще, и проверить, можем ли мы улучшить и расширить существующие источники для нашей конкретной предметной области и задачи. Повторное использование существующих онтологий может быть необходимым, если нашей системе нужно взаимодействовать с другими приложениями, которые уже вошли в отдельные онтологии или контролируемые словари. Многие онтологии уже доступны в электронном виде и могут быть импортированы в используемую среду проектирования онтологии.

В литературе и всемирной паутине существуют библиотеки повторно используемых онтологий. Например, общая библиотека онтологий http://protegewiki.stanford.edu/wiki/Protege_Ontology_Library, библиотека биомедицинских онтологий http://www.obofoundry.org/. Существуют инструменты поиска онтологий: http://watson.kmi.open.ac.uk/WatsonWUI/ и http://swoogle.umbc.edu/.

Однако будем считать, что соответствующих онтологий еще не существует, и начнем разрабатывать онтологию с нуля.

Шаг 3. Перечисление важных терминов в онтологии

Полезно составить список всех терминов, о которых мы хотели бы сказать что-либо или которые хотели бы объяснить пользователю. Какие термины мы бы хотели рассмотреть? Какие свойства имеют эти термины? Что бы мы хотели сказать об этих терминах? Например, в число важных терминов, связанных с винами, входят вино, виноград, винный завод, местоположение, цвет вина, его крепость, вкус и содержание сахара; различные виды еды, такие как рыба и мясо; типы вина, такие как белое вино и т. д. Вначале важно получить полный список терминов, не беспокоясь о пересечении понятий, которые они представляют, об отношениях между терминами, о возможных свойствах понятий или о том, чем являются термины — классами или свойствами.

Следующие два шага — разработка иерархии классов и определение свойств понятий — тесно переплетены. Сложно выполнить сначала один из них, а потом — другой. Обычно в иерархии мы даем несколько формулировок понятий и затем описываем свойства этих понятий и т. д. Также эти два шага — самые важные шаги в процессе проектирования онтологии.

Шаг 4. Определение классов и иерархии классов

Существует несколько возможных подходов для разработки иерархии классов:

- процесс **нисходящей** разработки начинается с определения самых общих понятий предметной области с последующей конкретизацией понятий. Например, мы можем начать с создания классов для общих понятий Вино и Еда. Затем мы конкретизируем класс Вино, создавая его подклассы: Белое вино, Красное вино, Розовое вино;
- процесс **восходящей** разработки начинается с определения самых конкретных классов, листьев иерархии, с последующей группировкой этих классов в более общие понятия. Например, сначала мы определяем класс вина Ахашени, а затем поднимаемся вверх и создаем более общие классы Красное полусладкое вино и Грузинское вино;
- процесс **комбинированной** разработки — это сочетание нисходящего и восходящего подходов: сначала мы определяем более заметные понятия, а затем соответствующим образом обобщаем и ограничиваем их.

Ни один из этих трех методов не лучше других по своей сути. Выбор подхода в большой степени зависит от личного взгляда на предметную область. Если разработчик склонен к рассмотрению предметной области сверху вниз, то ему, возможно, больше подойдет нисходящий метод. Часто для многих разработчиков онтологий самым простым является комбинированный метод, так как понятия,

находящиеся «посередине», имеют тенденцию быть самыми наглядными понятиями в предметной области.

Если вы склонны делать сначала самую общую классификацию, то вам больше подойдет нисходящий метод. Если бы вы начали приводить конкретные примеры, то более подходящим является восходящий метод.

Какой метод мы бы ни избрали, обычно мы начинаем с определения классов. Из списка, составленного в Шаге 3, мы выбираем термины, которые описывают объекты, существующие независимо, а не термины, которые описывают эти объекты. В онтологии эти термины будут классами и станут точками привязки в иерархии классов. Мы организуем классы в иерархическую таксономию, задавая контрольный вопрос: если объект является экземпляром одного класса, будет ли он обязательно (т. е. по определению) экземпляром некоторого другого класса?

Проверка правильности иерархии классов подробнее рассмотрена в разделе 5.4, типы отношений — в онтологиях.

Шаг 5. Определение свойств классов (слотов)

Классы сами по себе не предоставляют достаточно информации для ответа на вопросы проверки компетентности из Шага 1. После определения некоторого количества классов мы должны описать их внутреннюю структуру.

Мы уже выбрали классы из списка терминов, который мы создали на Шаге 3. Большинство оставшихся терминов, вероятно, будут свойствами этих классов. Эти термины включают, к примеру, цвет вина, его крепость, вкус и содержание сахара, а также местоположение винного завода.

Для каждого свойства из списка мы должны определить, какой класс оно описывает. Эти свойства станут свойствами (слотами), привязанными к классам. Таким образом, у класса Вино будут следующие свойства (слоты): цвет, крепость, вкус и сахар. А у класса Винный завод будет свойство «местоположение».

Все подклассы класса **наследуют** свойства этого класса. Например, все свойства класса Вино будут унаследованы всеми подклассами этого класса, включая Красное Вино и Белое Вино. К классу Красное Вино мы добавим дополнительное свойство уровень танина (низкий, средний или высокий). Данное свойство будет унаследовано всеми классами, представляющими красные вина.

Свойство должно быть привязано к самому общему классу, у которого может быть данное свойство. Например, крепость и цвет вина нужно будет привязать к классу Вино, так как это самый общий класс, чьи экземпляры будут иметь крепость и цвет.

Шаг 6. Определение ограничений на значения свойств (слотов)

Свойства (в некоторых системах используется понятие «слоты») могут иметь различные ограничения, которые описывают тип значения, разрешенные значения, число значений (мощность, кардинальность) и другие характеристики значений, которые может принимать свойство. Например, значение свойства «название» (как в «название вина») — одна строка. Т. е. «название» — это свойство с типом значения Строка. Свойство «производит» (как в выражении «винный завод производит эти вина») может иметь множественные значения, которые являются экземплярами класса Вино. Т. е. «производит» — это свойство с типом значения Экземпляр, и разрешенным классом является Вино.

Опишем несколько общих ограничений на свойства классов.

Мощность (кардинальность) свойства

Мощность свойства определяет, сколько значений может иметь свойство. В некоторых системах различаются только единичная мощность (возможно

только одно значение) и множественная мощность (возможно любое число значений). Крепость вина будет свойством единичной мощности (вино может иметь только одну крепость). Вина, производимые на конкретном заводе, заполняют свойство множественной мощности «производит» класса Винный завод. Некоторые системы позволяют определить минимальную и максимальную мощность для того, чтобы более точно описать количество значений свойства. Минимальная мощность N означает, что слот должен иметь не менее N значений.

Тип значения свойства

Вот список наиболее общих типов значений: **Строка**, **Число**, **Булевы** свойства (простые флаги «да — нет»), **Нумерованные** свойства (список конкретных разрешенных значений свойства). Например, мы можем установить, что свойство «вкус» может принять одно из трех возможных значений: сильный, умеренный и мягкий. Объектные свойства (в соответствии с языком представления онтологий RDFS) позволяют определить отношения между экземплярами классов (индивидами). Для объектных свойств также должен быть определен список разрешенных классов, экземпляры которых можно использовать в качестве значений свойства. Например, свойство «производит» класса Винный завод в качестве значений может иметь экземпляры класса Вино. Разрешенным классом для значений этого свойства является класс Вино.

Домен свойства и диапазон значений свойства

Разрешенные классы для оъектных свойств часто называют **диапазоном значений** свойства. Например, Вино является диапазоном значений свойства «производит». Некоторые системы позволяют ограничить диапазон значений свойства, если свойство привязано к определенному классу. Классы, к которым свойство привязано, называются **доменом** свойства. Класс Винный завод — домен свойства «производит». В системах, где мы *привязываем* свойства к классам, домен свойства обычно составляют классы, к которым привязано свойство. С одной стороны, нужно стараться сделать домен как можно более общим. С другой стороны, необходимо гарантировать, что каждый класс, к которому мы привязываем свойство, на самом деле имеет свойство.

Шаг 7. Создание экземпляров

Последний шаг — это создание отдельных экземпляров классов в иерархии. Для определения отдельного экземпляра класса требуется (1) выбрать класс, (2) создать отдельный экземпляр этого класса и (3) ввести значения свойств.

В случае необходимости получения более формальной онтологии, могут быть добавлены шаги с описанием ограничений классов (в том числе задание формальных определений) — см. раздел «Классификация онтологий по степени формальности» выше. Данные шаги подробно рассмотрены в руководстве по использованию редактора онтологий Protege [Horridge M. et al., 2011].

Когнитивный подход к разработке онтологий

Если в результате анализа области и предназначения онтологии выявлено, что онтология будет использоваться для коммуникаций людей, то следует учесть некоторые когнитивные аспекты онтологий. В частности, на этапе формирования иерархии классов представляется целесообразным применить некоторые результаты гештальт-психологии [Гаврилова, 2005; Гаврилова, Воинов, 2007]. Автор гештальт-психологии Макс Вертгеймер так сформулировал основной ***принцип хорошего гештальта*** (хорошей формы) или закон прегнантности [Вертгеймер, 1987]:

«Организация любой структуры в природе или в сознании должна быть настолько хороша (регулярна, полна, сбалансирована или симметрична), насколько позволяют существующие условия».

Также полезными могут быть и другие когнитивно-перцептивные законы:

- закон близости — визуальные стимулы (объекты), находящиеся близко друг от друга воспринимаются как единое целое;
- закон сходства — вещи, обладающие одинаковыми свойствами, обычно воспринимаются как нечто единое (цельное);
- закон включения В. Келера — есть тенденция воспринимать только большую фигуру, а не ту меньшую, которую она включает;
- закон парсимонии — самый простой пример является самым лучшим, известен как принцип «бритвы Оккама»: «не нужно умножать сущности без необходимости».

Переформулируем эти законы для практического использования инженером по знаниям. Основная гипотеза: «Гармония = концептуальный баланс + ясность» [Гаврилова, 2005].

При этом концептуальный баланс подразумевает, что

- понятия одного уровня иерархии на одной диаграмме связываются с родительским концептом одним и тем же типом отношения (например, «класс – подкласс» или «часть – целое»);
- глубина ветвей онтологического дерева должна быть примерно одинаковая (±2);
- общая картинка должна быть довольно симметричной;
- перекрестные ссылки должны быть по возможности исключены.

Ясность включает

- минимизацию. Так максимальное число концептов одного уровня или глубина ветви не должна превышать знаменитое число Ингве-Миллера (7±2);
- прозрачность для чтения. Тип отношений на диаграмме должен быть по возможности очевиден, так чтобы не перегружать схему онтологии лишней информацией и опускать названия отношений.

Иллюстрация принципа хорошего гештальта представлена на рисунке 5.21. Трудно не согласиться с более «правильной» формой онтологии *A* по сравнению с онтологией *B*. Резюмируя, можно сказать, что в случае использования онтологии для коммуникации с людьми следует стремиться к концептуальному балансу и ясности создаваемых концептуальных моделей. В случае, если требования к онтологии и предметная область не позволяют этого обеспечить (например, поставлена задача «по-крупному» проработать всю предметную область и подробнее разобрать один раздел или разные разделы предметной области находятся на разных уровнях зрелости/освоения), то следует при демонстрации онтологии человеку разделить ее на сбалансированные и ясные фрагменты (например, общий вид без деталей + отдельная проработанная ветка).

Обзор методологий разработки онтологий

Методы разработки онтологий начали развиваться в 90-х годах, в частности, возникли следующие методологии: Сус-метод (1990); метод Ушолда и Кинга (1995), основанный на разработке EnterpriseOntology, т. е. онтологии для моделирования предприятий; методология Грюнингера и Фокса (1995), основанная на опыте разработки онтологии TOVE, также предназначенной для бизнес-моделирования; метод KACTUS (1996), разработанный в рамках европейского про-

екта Esprit, одной из целей которого являлось исследование возможности переиспользования знаний в сложной технической системе и роль онтологии в поддержке этого; методология METHONTOLOGY (1997), базирующаяся на основных видах деятельности по разработке программного обеспечения; метод SENSUS (1997); методология On-To-Knowledge (2002). Обзор данных методологий можно найти в работе [Загорулько, 2008].

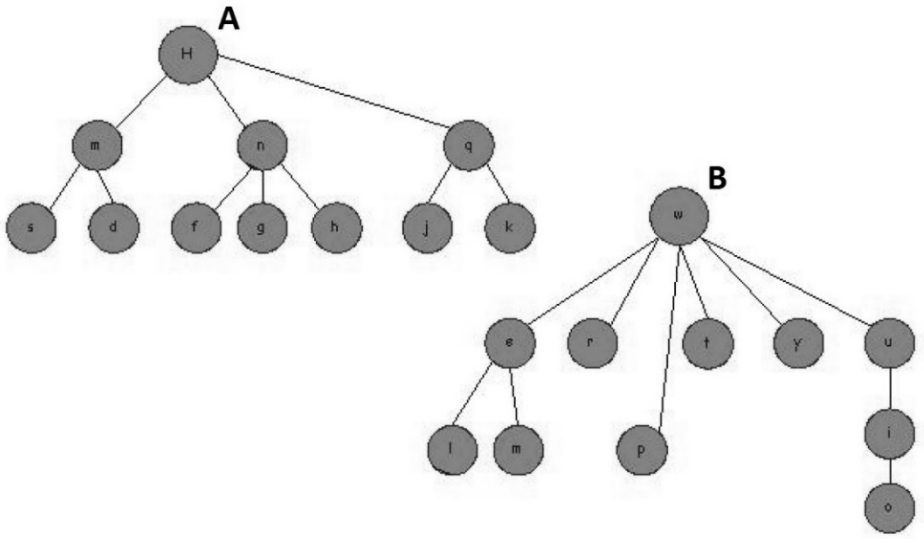


Рис. 5.21

Принцип хорошего гештальта

В последнее время все большую роль в методах проектирования и разработки начинает играть поддержка совместной работы, поэтому рассмотрим подробнее более современные методологии DOGMA-MESS и DILIGENT, в которых заложена коллективная разработка.

DOGMA-MESS [de Moor et al, 2006; De Leenheer, Debruyne, 2008] — это методология разработки и развития онтологий, основанная на сообществе (DOGMA — сокр. от Developing Ontology-Grounded Methods and Applications). Методология подчеркивает важность разработки общей концептуальной модели, особенно когда процесс совместной разработки онтологий пересекает границы одной организации. Разработчики онтологии расширяют согласованную (consensual) верхнеуровневую онтологию в соответствии с предварительно заданными ограничениями — политиками повторного использования (reuse policies [De Leenheer et al, 2007]). Если ограничения/политики нарушаются, то возникает конфликт.

DOGMA-MESS предлагает общую модель для межорганизационного процесса разработки онтологий. В рамках данного процесса знание движется по восходящей спирали от индивидуального уровня через организационный уровень к межорганизационному. Межорганизационная онтология состоит из различных взаимосвязанных частных онтологий. Циклический процесс разработки включает 4 фазы.

1. Закладка основ сообщества (Community Grounding): Процесс разработки начинается с формирования общей верхнеуровневой онтологии (OBO, upper

common ontology), которая содержит понятия и ограничения общие для всей рассматриваемой предметной области. Данная онтология разделяема сообществом.

2. Проработка частных вопросов/перспектив (Perspective Rendering): Далее каждый участник разработки расширяет данную онтологию (ОВО), например, путем добавления более детальных элементов (specializing). В результате создаются частные дивергентные концептуальные модели различных участников. Эксперты в предметных областях на данном этапе выражают свои знания в более привычных для них терминах. Для стимулирования использования ОВО применяются соответствующие политики (reuse policy).

3. Объединение частных перспектив (Perspective Unification). Частные перспективы и ОВО объединяются в общую нижнеуровневую онтологию (ОНО, lower common ontology). Используется простое правило: все определения должны быть полной специализацией шаблонов ОВО. Однако такое упрощение часто не работает, конструктивистская парадигма требует нарушения политик повторного использования и создания новых понятий в общей онтологии, которые не являются специализацией ОВО и вносят в нее новые идеи.

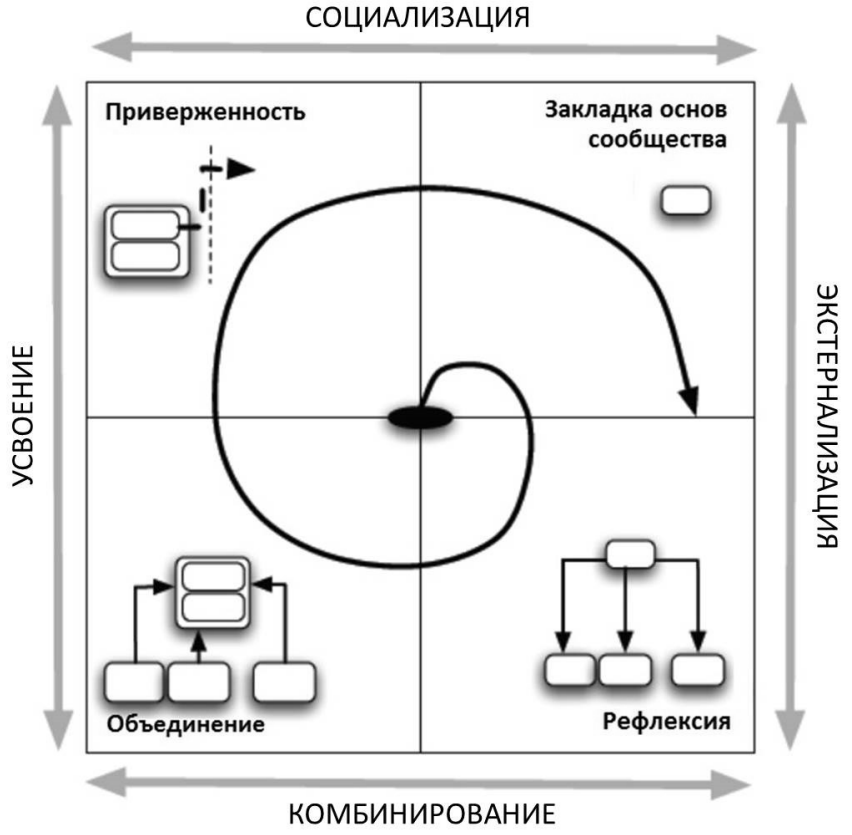


Рис. 5.22

Цикл развития онтологии в методологии DOGMA-MESS [De Leenheer, Debruyne, 2008]

4. Институционализа́ция общих взглядов (Perspective Version Commitment): Часть ОНО, которая согласована сообществом, формирует лигитимную ОВО для следующего цикла онтологического инжиниринга. Все участвующие в разработке организации усваивают новую онтологию и приводят свою базу экземпляров в соответствие с новой ОВО.

Некоторые методологии совместной разработки онтологий основаны на использовании модели аргументации для достижения консенсуса. Например, HCOME [Kotis, 2008] и DILIGENT [Tempich et al, 2005, 2007] являются методологиями, которые следуют формальной модели аргументации. DILIGENT использует онтологию аргументации для описания и хранения (в качестве экземпляров классов) различных цепочек обсуждений. Благодаря этому, легко могут быть найдены и проанализированы предпосылки и обоснования принятых в прошлом решений, касающихся разрабатываемой онтологии.

Методология DILIGENT — методология разработки и сопровождения (evolution) онтологии в условиях множественности и распределенности экспертов. Процесс разработки онтологии в соответствии с данной методологией включает в себя следующие шаги:

1. Создание. На данном этапе быстро создается первая версия онтологии, которая может быть передана для использования заинтересованным сторонам.

2. Локальная адаптация. Пользователи применяют исходную онтологию и локально адаптируют ее под свои потребности.

3. Анализ. Контрольная комиссия оценивает изменения, предлагаемые пользователями онтологии, анализирует локальные онтологии, запросы и предложения пользователей. В результате такого анализа комиссия пытается выявить сходства в онтологиях пользователей.

4. Ревизия. Комиссия принимает решения об изменениях в онтологии. Комиссия должна регулярно проводить ревизию разделяемой онтологии, чтобы локальные онтологии существенно от нее не отличались.

5. Локальное обновление. Пользователи онтологии обновляют свои локальные онтологии с учетом измененной версии общей онтологии.

Предлагаемые изменения и принимаемые решения об изменениях сопровождаются аргументами и основаны на соответствующей методологии аргументации. Методология аргументации DILIGENT (The DILIGENT argumentation framework) позволяет зафиксировать размышления, которые распределены по дискуссиям, связанным с разрабатываемой онтологией. Это позволяет сделать процесс поиска консенсуса более эффективным. Методология аргументации DILIGENT поддерживает взаимодействие между инженером по знаниям и экспертом с помощью эксплицитного представления аргументов, что помогает согласовать различные мнения и получить по-настоящему *разделяемую* онтологию.

Методология аргументации DILIGENT состоит из процесса и формальной модели (онтологии) аргументации, технологически методология поддерживается основанным на вики инструментом — *coefficient Makna* (collaborative, ontology-engineering-efficient Makna) и Cicero [Dellschaft et al, 2008], которая является плагином к среде онтологического инжиниринга NeOn (http://neon-toolkit.org/wiki/Cicero).

Процесс аргументации состоит из следующих шагов: 1. Выбор модератора (человека внутри команды разработки онтологии); 2. Выбор процедуры принятия решений (например, способ голосования и условия для начала голосования); 3. Спецификация спорных вопросов/предметов спора (issues); 4. Генерация аргу-

ментов и идей (для повышения эффективности предлагается перечень стандартных типов аргументов — детализация, оценка, альтернатива, пример), 5. Принятие решений по спорным вопросам и идеям.

Онтология аргументации включает следующие понятия: спорный вопрос/предмет спора (*issues*), идея, аргументы. С целью поиска консенсуса разработчики онтологии высказывают свои мнения с помощью аргументов, относящихся к спорным вопросам, идеям или элементам онтологии. Обоснования (*Justification)* — это аргументы в поддержку спорного вопроса или идеи. Возражения (*Challenge)* — аргументы против спорного вопроса или идеи. Онтология дополнительно разделяет поддерживающие аргументы на примеры и оценки (*evaluations*), а возражения на контрпримеры и альтернативы. Внутри дискуссий участники могут также выражать свою позицию (*position*) через согласие или несогласие. После того, как аргументы и позиции участников доступны, команда разработчиков может принять решение (связанное со спорным вопросом, идеями и элементами онтологии).

В основе онтологии DILIGENT лежат более общие идеи систем поддержки решений, позволяющие множеству людей участвовать в обсуждении процесса проектирования (системы типа IBIS: Issue-Based Information System [Kunz et al, 1970]).

Традиционно методологии коллективной разработки онтологий ориентированы на централизированный подход сверху-вниз (DILIGENT, DOGMA-MESS), в котором основная копия онтологии поддерживается и обновляется группами разработчиков, занимающимися локальными адаптациями.

Исследователи из Мадрида и Карлсруэ предлагают комплексный подход к коллективной разработке, который также поддерживает децентрализованную коллективную разработку [Palma et al, 2011]. Он основан на формализации процесса коллективной разработки онтологий с помощью эксплицитных редакторских потоков работ (editorial workflows). Указанные потоки работ координируют заявки на изменения в онтологиях. Данный подход поддерживается специальными моделями, методами и стратегиями управления изменениями в онтологиях в распределенных средах. В частности: форма представления изменений в онтологиях, организованная по уровням для обеспечения независимости от онтологических языков, а также методы и стратегии для их обработки, управления версиями, приобретения, хранения и сопровождения. Кроме того, данный подход включает набор стратегий распространения изменений (change propagation), который позволяет синхронизировать распределенные копии одной онтологии.

Методология NeOn

Наиболее развитой, проработанной и комплексной на текущий момент является методология NeOn **(www.neon-project.org)**, в которой интегрированы достижения предшествующих методологий [Suárez-Figueroa et al, 2012].

Особенности методологии NeOn:

- повторное использование знаний: неонтологических ресурсов (глоссарии, тезаурусы, таксономии, классификационные схемы...), готовых онтологий, шаблонов (ontology design patterns);
- контекстуализация онтологий (кастомизация, модуляризация, локализация);
- поддержка всего жизненного цикла работы с онтологиями;
- динамика (эволюция онтологий);
- совместная работа.

Методология NeOn основана на следующих элементах.

Сеть онтологий (Ontology network) — комплекс интегрированных онтологий, созданный для конкретного приложения.

Типовые задачи онтологического инжиниринга — в рамках методологии NeOn выделено 53 типовые задачи работы с онтологиями (концептуализация онтологии, отображение онтологии, развитие онтологии...), для которых разработаны согласованные определения, предлагаются специальные методы, техники и инструменты. (русский перевод см. http://ontology.ipi.ac.ru/index.php/Категория: Список_терминов).

Модель жизненного цикла сети онтологий (ontology network life cycle model) — обобщенная схема, выбранная организацией для систематизации и отбора типовых задач онтологического инжиниринга с целью получения собственного (для организации) жизненного цикла онтологической системы.

Возможные модели жизненного цикла онтологий:

- каскадная (Waterfall),
- пошаговая (Incremental),
- итеративная (Iterative),
- развития (эволюция) прототипа (Evolving Prototyping),
- спиральная (Spiral).

Жизненный цикл сети онтологий (ontology network life cycle) — последовательность задач онтологического инжиниринга для реализации конкретного проекта, выполняемая инженерами по знаниям и разработчиками программных продуктов. Данная последовательность получается путем отображения типовых задач онтологического инжиниринга на выбранную модель жизненного цикла онтологической системы.

В методологии онтологического инжиниринга NeOn предлагается создавать жизненный цикл онтологической системы под конкретную ситуацию.

Последовательность разработки жизненного цикла сети онтологий.

1. Определение требований к разработке сети онтологий.
2. Выбор модели жизненного цикла сети онтологий.
   - Выявление характеристик и ограничений для процесса разработки сети онтологий;
   - Выявление всех возможных моделей жизненного цикла сети онтологий (для выбора модели жизненного цикла сети онтологий предлагается дерево решений);
   - Оценка различных моделей жизненного цикла сети онтологий на основе прошлого опыта и возможностей организации;
   - Выбор модели жизненного цикла сети онтологий, которая наиболее соответствует характеристикам и ограничениям для процесса разработки онтологической системы.
3. Выбор требуемых задач онтологического инжиниринга из списка типовых задач. Часть задач является обязательной для любой ситуации (required), а часть задач выполняется при определенных условиях (If applicable). Для выбора задач используются диагностические вопросы.
4. Отображение выбранных задач онтологического инжиниринга на выбранную модель жизненного цикла сети онтологий.
5. Определение порядка выполнения задач — жизненный цикл сети онтологий под конкретную задачу конкретной организации.

Наиболее распространенные сценарии создания сети онтологий (ontology network, см. рис. 5.23).

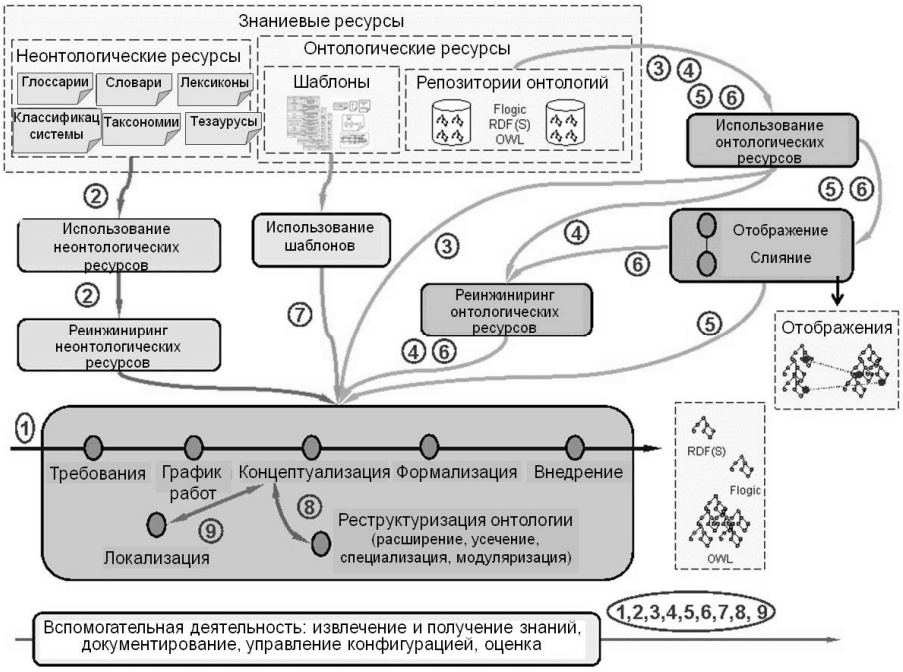


Рис. 5.23

Сценарии создания сети онтологий в методологии NeOn

Сценарий 1: Создание сети онтологий «с нуля» без использования существующих ресурсов.

Сценарий 2: Создание сети онтологий путем повторного использования «неонтологических» ресурсов (поле знаний, не доведенное до уровня онтологии или ее фрагментов).

Сценарий 3: Создание сети онтологий путем повторного использования существующих онтологий или модулей.

Сценарий 4: Создание сети онтологий путем повторного использования и реинжиниринга существующих онтологий или модулей.

Сценарий 5: Создание сети онтологий путем повторного использования и объединения существующих онтологий или модулей.

Сценарий 6: Создание сети онтологий путем повторного использования, объединения и реинжиниринга существующих онтологий или модулей.

Сценарий 7: Создание сети онтологий путем реструктуризации существующих онтологий или модулей.

Сценарий 8: Создание сети онтологий путем локализации существующих онтологий или модулей.

5.6. ПРОЧИЕ МЕТОДЫ ОНТОЛОГИЧЕСКОГО ИНЖИНИРИНГА (ОТОБРАЖЕНИЕ, ВИЗУАЛИЗАЦИЯ, ОЦЕНКА)

Методы отображения (ontology mapping, matching) онтологий

Проблема отображения неоднородных онтологий является актуальной с самого начала использования онтологий при создании информационных систем.

Отображение онтологий нужно для интеграции информации, находящейся в различных хранилищах и информационных системах, обеспечения доступа к различным источникам информации, а также для коммуникаций людей и обмена знаниями. Говоря о неоднородных онтологиях, мы подразумеваем, что две (или более) онтологии по-разному описывают одну и ту же предметную область или близкие предметные области с точки зрения разных сообществ. Онтология задает подразумеваемую семантику для понятий предметной области и определяет онтологический контекст, в котором работает сообщество. Разные контексты использования онтологий, созданных разными сообществами, отражаются на особенностях подходов к спецификации понятий, что становится одной из причин неоднородности. В результате семантика понятий в контекстах, описанных разными онтологиями, может быть сходной при различных подходах к описанию их структуры: составу, ограничениям и степени детализации.

Под отображением онтологий мы понимаем процесс, при котором понятия одной онтологии выражаются через понятия другой. Часто термин отображение корректно рассматривается в двух ракурсах: как процесс отображения одной онтологии в другую или как результат [Euzenat, Shvaiko, 2013] такого процесса, т. е. множество функций отображения понятий одной онтологии в понятия другой. Отображение онтологий является неотъемлемой частью большинства задач согласования [Euzenat, Shvaiko, 2013] онтологий, таких как слияние (merging), выравнивание онтологий (alignment), модификация одной онтологии для достижения однородности с другой и так далее.

Сложность отображения онтологий объясняется рассмотренными выше проблемами омонимии, полисемии и синонимии.

1. Сущности (классы, свойства, связи, объекты), имеющие одинаковые имена, могут иметь разный смысл.

2. Сущности (классы, свойства, связи, объекты), имеющие одинаковый смысл, могут иметь разные имена.

Отображение онтологий разделяется на 2 подзадачи.

1. Локальное отображение сущностей.

2. Глобальное отображение сущностей с учетом структуры всей онтологии. Под глобальным отображением подразумевается пересмотр (пересчет) локальных отображений с учетом отображений всех остальных элементов. Например, локальное отображение некоторых сущностей A1 и A2 может быть установлено на основе соответствия между их родовыми сущностями A01 и A02, которое изначально может быть вероятностным/недоопределенным. В процессе формирования глобального отображения сущностей соответствие между A1 и A2 пересматривается с учетом дополнительной информации по соответствию A01 и A02.

Пример отображения онтологий в среде онтологического инжиниринга NeOn представлен на рисунке 5.24.

Рассмотрим более подробно процесс установки локального отображения сущностей. Современные методы отображения онтологий носят междисциплинарный характер. Среди них можно выделить 4 группы: терминологические, экстенсиональные, структурные и логические (формальные, семантические) методы [Euzenat, 2004]. Для обеспечения точности отображения сущностей обычно используются одновременно несколько методов.

Терминологический анализ

На данном этапе определяется сходство между сущностями на основе сравнения имен сущностей (оценка количества совпадающих символов, общие части слов,

например, «Цели» и «Целевые установки») или путем анализа синонимичных терминов. Для выявления синонимичных терминов могут использоваться существующие словари общей и профессиональной лексики, тезаурусы. Данный вид анализа можно считать исходным для установления соответствия между сущностями.

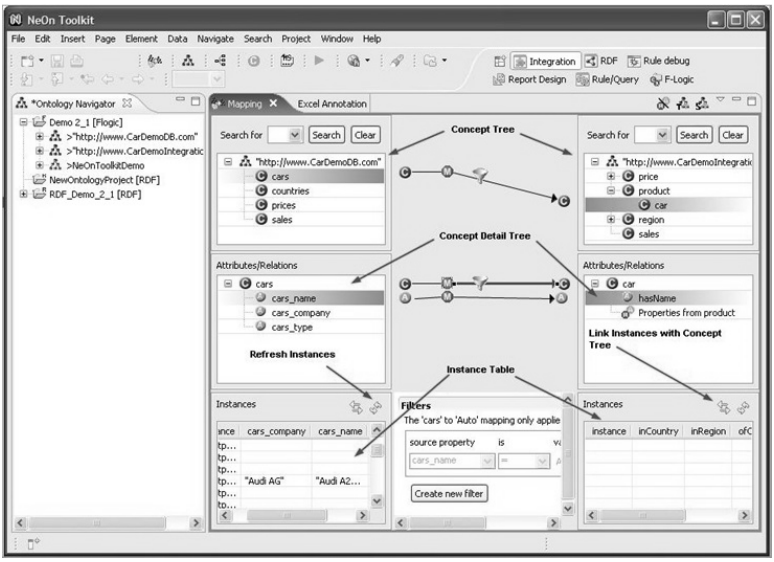


Рис. 5.24

Пример отображения онтологий в системе NeOn

Структурный анализ

Анализ внутренней структуры

В данном случае оценка сходства производится на основе анализа атрибутов сущностей или, говоря на языке OWL, свойств-значений (DatatypeProperty). Сходство сущностей характеризуется сходством областей допустимых значений атрибутов. Методы анализа внутренней структуры иногда называются методами на основе ограничений. Сущностей со схожей внутренней структурой, а также свойств с похожими доменом и областью значений может быть достаточно много, поэтому данные методы используются только для формирования кластеров сходных понятий и требуют сочетания с другими методами.

Анализ внешней структуры

Анализ сходства по иерархическим связям

Оценка схожести двух сущностей двух онтологий может быть основана на позициях данных сущностей в иерархии классов. Если две сущности двух онтологий схожи, то их «соседи» также как-то схожи. Такое утверждение может использоваться по-разному и порождает ряд возможных критериев (признаков) для сходства двух сущностей.

1. Их прямые суперсущности (или все суперсущности) уже являются схожими.

2. Их сущности-братья (или все их сущности-братья) уже являются схожими.

3. Их прямые сущности-потомки (или все их сущности-потомки) уже являются схожими.

4. Все их сущности-листья (сущности, не имеющие потомков, находящиеся в дереве, корнем которой является рассматриваемая сущность) уже являются схожими.

5. Все (или большинство) сущности на пути от корня к рассматриваемой сущности уже являются схожими.

Конечно, использование данных критериев должно сопровождаться другими критериями.

Анализ сходства по перекрестным связям

Определение сходства между сущностями может быть основано также на анализе связей сущностей. Если класс A1 связан с классом B1 связью типа R1 в одной онтологии, а класс A2 связан с B2 связью типа R2 в другой онтологии, и если известно, что B1 и B2 — схожи, R1 и R2 — схожи, можно предположить схожесть A1 и A2. Подобным образом можно говорить и о сходстве типов связей — R1 и R2, если известно, что A1 и A2 — схожи, B1 и B2 — схожи. Например, классы «Компания» и «Университет» будут оценены как схожие, поскольку они имеют схожую связь типа «имеет подчиненного» с классом «Сотрудник» и классом «Профессор», которые были признаны схожими.

Экстенсиональный анализ

Для оценки экстенсионального соответствия классов используются существующие экземпляры классов. Для установки соответствия между сущностями используются следующие диагностические правила:

C1 эквивалентен C2 — *невозможно найти объект O1:C1, такой что не O1:C2 и наоборот.*

C1 подкласс C2 — *невозможно найти объект O1:C1, такой что O1:C2 и C1 не эквивалентен C2.*

Анализ экстенсионала позволяет также идентифицировать классы-роли, когда возникает два разных класса для описания одного экстенсионала.

Логический анализ

Логический анализ основан на выявлении родовых классов сопоставляемых классов и анализе наложенных на них ограничений.

Например, в одной онтологии может существовать класс «Микрокомпания», который является видовым классом для класса «Компания» с наложенным ограничением на «Число сотрудников» <5. В другой онтологии может существовать класс «Малое предприятие», который является видовым классом для класса «Фирма» с наложенным ограничением на «Число работников» < 10. В случае сходства родовых классов сравниваются наложенные на них ограничения. Для этого сравниваются свойства классов «Число сотрудников» и «Число работников», если данные свойства схожи, то проводится сравнение наложенных ограничений <5 (сотрудников) и < 10 (работников). В результате делается заключение, что «Микро-фирма» $\subset$ «Малое предприятие».

Ограничением данного метода является потребность в «якорях»-сущностях, которые либо заведомо эквивалентны в двух сопоставляемых онтологиях, либо являются разделяемыми сущностями в некоторой сторонней онтологии.

Выбор методов локального отображения онтологий

Выбор методов локального отображения сущностей зависит от этапа процесса отображения онтологий:

- на начальном этапе при поиске классов с высокой вероятностью взаимного соответствия основными методами локального отображения являются терминологический и экстенсиональный анализ;
- по мере накопления отображений между сущностями все большую роль начинают играть структурный и логический виды анализа.

Особенности отображения классов-ролей

При установлении соответствия между классами, одним из которых является класс-роль [Лукашевич, 2011], может неявно задаваться связь «роль-исполнитель». Классы-роли всегда присутствуют в масштабных онтологиях и позволяют установить связь онтологии предметной области с прагматикой (например, с онтологией решаемых задач), отразить локальные точки зрения на предметную область. Класс-роль — класс, определяющий способ участия (роль) сущности в контексте. Понятие «Роль» рассматривается посредством двух связанных с ним понятий: «Исполнитель», который выполняет роль и «Контекст», в котором он ее выполняет. Для отображения классов-ролей необходимо эксплицитное представление контекста, поскольку исполнители могут выполнять одновременно несколько ролей в разных контекстах.

Класс-роль «Альтернатива» широко распространен в методах принятия решений. В контексте задачи «Разработка стратегии развития предприятия» класс-роль «Альтернатива» будет соответствовать классу «Проект» и/или «Целевая программа», в контексте задачи «Подбор персонала» — классу Сотрудник, а в контексте «Выбора и закупки оборудования» — классу «Основные фонды».

Глобальное отображение сущностей

После получения локальных соответствий между сущностями определяется глобальное соответствие. Практические рекомендации по расстановке приоритетов между результатами различных способов локального анализа.

При наличии баз знаний, включающих в себя экземпляры отображаемых онтологий, приоритетное значение имеют результаты экстенсионального анализа.

При наличии «якорей» в отображаемых онтологиях приоритетное значение имеют результаты логического анализа.

Однако результаты любого анализа следует согласовывать с результатами, полученными с использованием других видов анализа.

Перспективы развития методов отображения онтологий см. в [Shvaiko, Euzenat, 2013].

Методы визуализации онтологий

Визуализация онтологий нужна для более эффективного восприятия пользователями онтологий и связанных с ними данных. Рассмотрим методы визуализации онтологий, детальный обзор которых был сделан в 2007 году в работе [Katifori et al, 2007], в рамках которой выделяется шесть категорий методов визуализации онтологий и рассматриваются их достоинства и недостатки:

Иерархический список (Indented List): многие инструменты визуализации онтологий, в том числе Protege, OntoEdit, Kaon и некоторые другие, предлагают в качестве основного способа визуализации представления в стиле «Проводника» в Windows.

Узлы-связи и деревья (Node-link and Tree): методы этой категории отображают онтологию как набор соединяющихся узлов, представляющих иерархию классов в виде вертикально- или горизонтально-размещенного дерева. Пользователю обычно предоставляется возможность разворачивать и сворачивать узлы и поддеревья для управления детализацией информации. Двухмерная реализация данного метода осуществляется в OntoViz, IzaVis, SpaceTree, трехмерная в OntoSphere (рис. 5.25).

Масштабируемые (Zoomable): методы данной категории отражают иерархию классов и экземпляры через визуальную вложенность объектов (диаграммы Вена, «меньшие коробочки размещаются в больших»). При выборе объекта он увеличивается в размере (*zoom-in)*, позволяя более детально рассмотреть свои

компоненты (подклассы и экземпляры). Примеры систем и соответствующих методов: Grokker (рис. 5.26), Jambalaya, CropCircles.

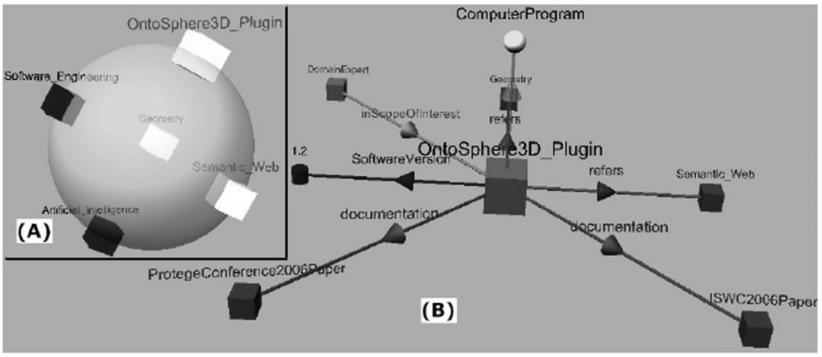


Рис. 5.25

Визуализация с помощью OntoSphere (2 возможных вида A и B)

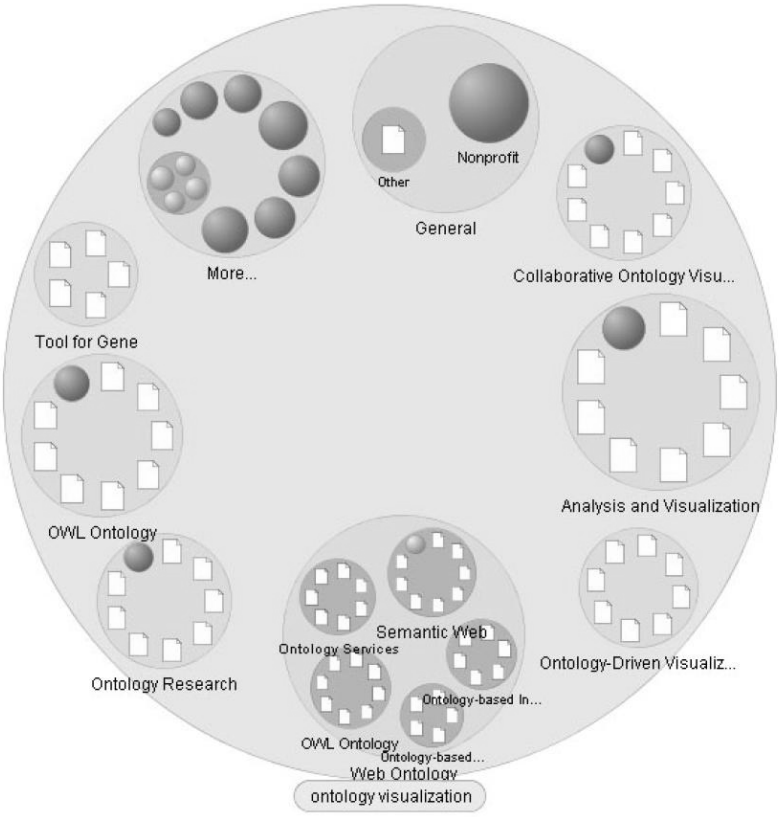


Рис. 5.26

Визуализация результатов поиска по запросу «ontology visualization» (визуализация онтологий):
Примечание: кружки — классы, «белые документы» — найденные ресурсы.

Заполнение пространства (Space-filling): идея данной группы методов заключается в использовании всего пространства экрана путем разделения пространства каждого узла между его потомками в зависимости от свойств узла, например, количества потомков и т. п. Примеры: TreeMaps, Information Slices.

Фокусирование + контекст (Focus + Context): данная группа техник основана на понятии искажения (distorting) графа с целью фокусировки на определенном узле и представления его контекста (окружения). Фокусный узел отображается в центре, а связанные с ним элементы располагаются вокруг него и их размер уменьшается по мере удаления от фокуса. Для увеличения какого-либо узла пользователь должен сделать его фокусным.

Примеры: двухмерные Touchgraph и TGVizTab (рис. 5.27), 3D Hyperbolyc Tree (рис. 5.28).

Трехмерные информационные ландшафты (3D Information Landscapes).

В статье [Katifori et al, 2007] также дается сравнительный анализ представленных методов.

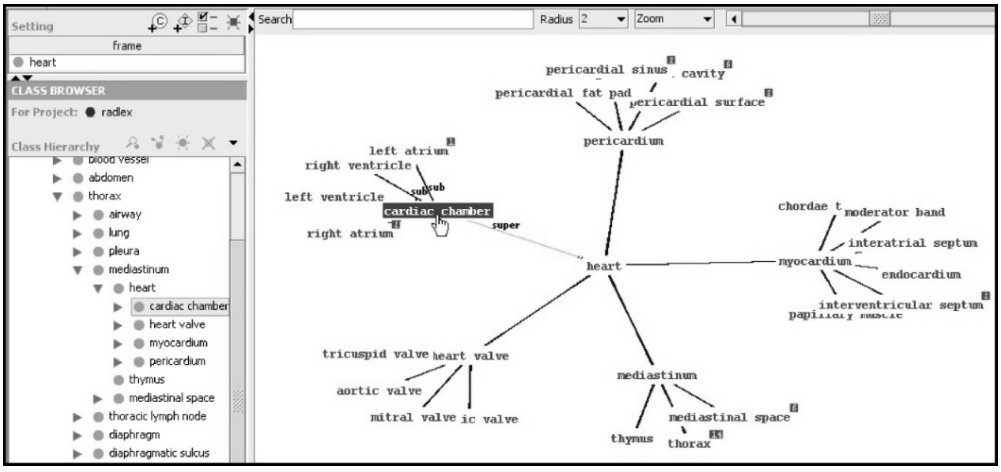


Рис. 5.27

Пример визуализации взаимосвязей понятий в TGVizTab

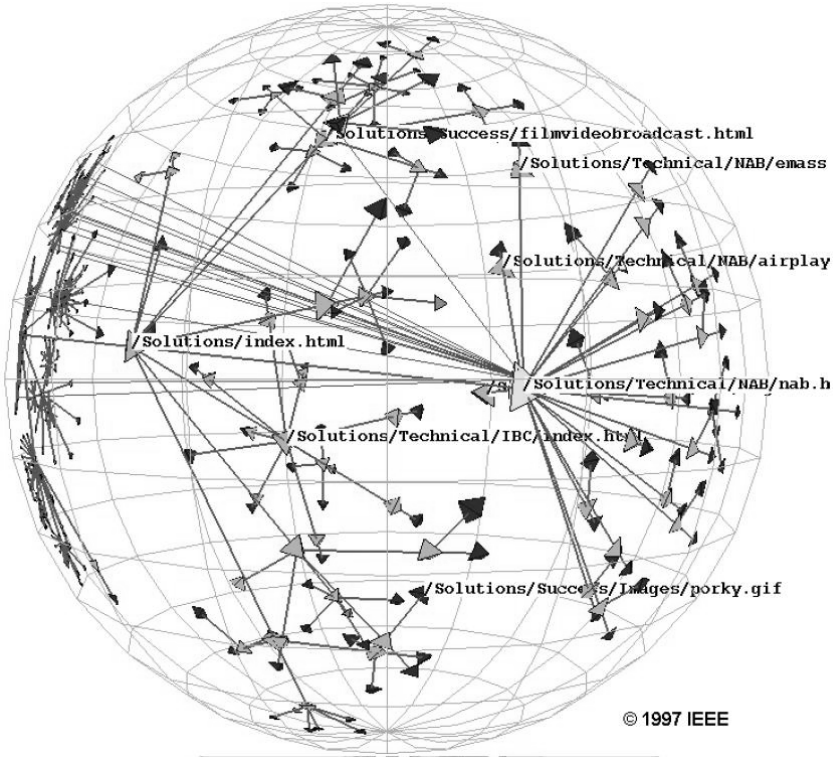


Рис. 5.28

Пример 3D-визуализации взаимосвязей между ресурсами с помощью гиперграфа

Методы оценки онтологий[1]

Вопрос оценки качества создаваемых онтологий является одной из актуальных проблем современного онтологического инжиниринга. Эта часть процесса разработки онтологий важна в практическом плане, поэтому так велик интерес к этой проблеме. В настоящее время разными группами специалистов разработано более десятка различных подходов в области оценки онтологий. Тем не менее, задача выбора подходящей методики оценки для решения конкретной задачи остается сложной теоретически нерешенной проблемой.

Хорошие обзоры существующих методов и подходов в оценке онтологий изложены в работах [Hartmann, 2005], [Brank, 2005]. Качественные критерии оценки с позиций гештальт-психологии предложены в работах Гавриловой [Гаврилова, 2005, 2007].

Обобщенный взгляд на существующие подходы предлагает модель классификации методов оценки онтологий на рисунке 5.29, предложенная в [Горовой, 2007].

Существующие методы оценки онтологий ставят перед собой одну из следующих целей:

- полнота и точность словаря предметной области;
- адекватность структуры с точки зрения таксономии, отношений и т. п.;

[1] На основе материала В. Горового и Е. Болотниковой.

- воспринимаемость (с когнитивной точки зрения);
- производительность при использовании в приложениях;
- выбор лучшей онтологии из нескольких имеющихся.

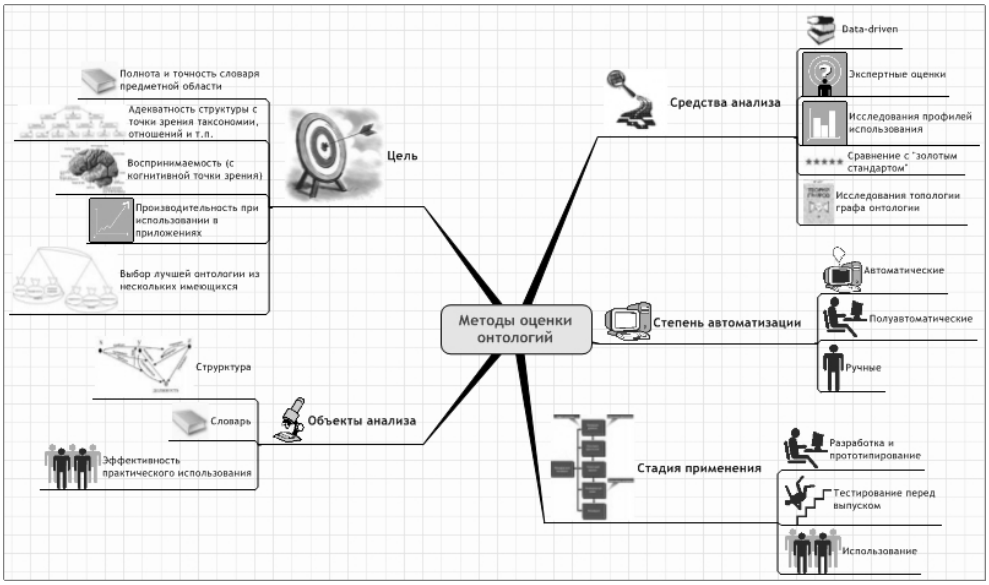


Рис. 5.29

Модель классификации методов оценки онтологий

Оценка онтологий может применяться на разных стадиях разработки и использования онтологий (разработка и прототипирование, тестирование перед выпуском, использование).

По степени автоматизации все методы оценки онтологий можно разделить на три группы: автоматические, полуавтоматические, ручные.

Объектами для анализа у существующих методов могут быть один или несколько из перечисленных ниже объектов: структура, словарь, эффективность практического использования.

По средствам, используемым для анализа качества и зрелости онтологий, можно все методы разделить на несколько классов: основанные на данных (Data-driven); экспертные оценки; исследования профилей использования; сравнение с «золотым стандартом»; исследования топологии графа онтологии.

Вопросы, относящиеся к когнитивной эргономике при оценке онтологий, рассматриваются в работах [Гаврилова и др., 2009].

5.7. ПРЕДСТАВЛЕНИЕ ОНТОЛОГИЙ И ЯЗЫКИ СЕМАНТИЧЕСКОГО ВЕБА

Для представления онтологий и индексирования информационных ресурсов в настоящее время используются языки семантического веба. В 1997 году консорциум W3C определил спецификацию RDF (Resource Description Framework). RDF предоставляет простой, но мощный язык описания ресурсов, основанный на триплетах (triple-based) «Субъект-Предикат-Объект» и спецификации URI. В 1999 г. RDF получает статус рекомендации. Этот шаг в направлении улучшения функциональности и обеспечения интероперабельности (т. е. возможности

обмениваться данными, несмотря на их разнородность) в Сети считается одним из важнейших. Концептуально RDF дает минимальный уровень для представления знаний в Сети. Спецификация RDF опирается на ранние стандарты, лежащие в основе Web:

- Unicode служит для представления символов алфавитов различных языков;
- URI используется для определения уникальных идентификаторов ресурсов;
- XML и XML Schema — для структурирования и обмена информацией и для хранения RDF (XML синтаксис RDF).

Кроме RDF был разработан язык описания структурированных словарей для RDF — RDF Schema (RDFS). Он предоставляет минимальный набор средств для спецификации онтологий. RDFS получил статус рекомендации W3C в 2004 году. Однако препятствием для Semantic Web стало то, что документов, написанных на языке RDF/RDFS, было относительно мало. В период с 2001 по 2004 годы шла интенсивная работа по созданию программных средств для обработки и автоматической генерации RDF-документов.

Результатом в 2004 году стал язык GRDDL (Gleaning Resource Descriptions form Dialects of Languages). Его назначение состоит в предоставлении средств для извлечения RDF-триплетов из XML и XHTML данных (в особенности это относится к документам, автоматически генерируемым из закрытых баз данных). Развивалось и программное обеспечение для Semantic Web. В области создания библиотек классов и построения логических выводов над RDF-графами была создана библиотека Jena Framework, в области создания модулей расширения для браузеров — Simile для Firefox. В области создания визуальных сред редактирования большое число редакторов онтологий стали поддерживать RDF.

В 2004 г. статус рекомендации получил язык OWL (Web Ontology Language). OWL имеет несколько «диалектов»: устаревший и очень упрощенный OWL Lite, наиболее часто используемый OWL DL (Description Logic), и богатый по возможностям OWL Full. Одно из различий между OWL DL и OWL Full состоит в том, что для OWL DL гарантируется вычислимость любого логического выражения, а для OWL Full — нет. С другой стороны, OWL Full позволяет, например, сделать какую-либо сущность классом и индивидом одновременно, что бывает необходимо в сложных моделях. OWL фактически является надстройкой над RDF/RDFS и поддерживает эффективное представление онтологий в терминах классов и свойств, обеспечение простых логических проверок целостности онтологии и связывание онтологий друг с другом (импорт внешних определений). Многие формализмы описания знаний могут быть отображены на формализм OWL (два из его диалектов — OWL Lite и OWL DL — соответствуют двум дескриптивным логикам, имеющим разную выразительную силу). Наконец, существует новая версия стандарта — OWL 2. В ней определены так называемые профили OWL, налагающие на возможности языка различные ограничения: OWL RL, OWL QL и др. Большое число создаваемых в настоящее время онтологий кодируются на OWL; уже существующие онтологии транслируются в него. На этом работа по обеспечению Semantic Web необходимыми стандартами не остановилась. В 2005 году началась работа над форматом обмена правилами — RIF (Rule Interchange Format). Его назначение — соединить в одном стандарте несколько формализмов для описания правил, по которым может осуществляться нетривиальный логический вывод: логику клауз Хорна, логики высших порядков, продукционные модели и т. п. Также для представления правил предложен стандарт Semantic Web Rule Language (SWRL), который интегрирован с OWL.

Язык SPARQL — язык запросов к RDF-хранилищам — в январе 2008 г. приобрел статус официальной рекомендации Консорциума W3C. Синтаксически он очень похож на SQL. Он уже широко используется разработчиками информационных систем.

На рисунке 5.30 представлена диаграмма, называемая иногда стеком (или даже «слоеным пирогом») семантического веба (Semantic Web), которая интегрирует отдельные языки.

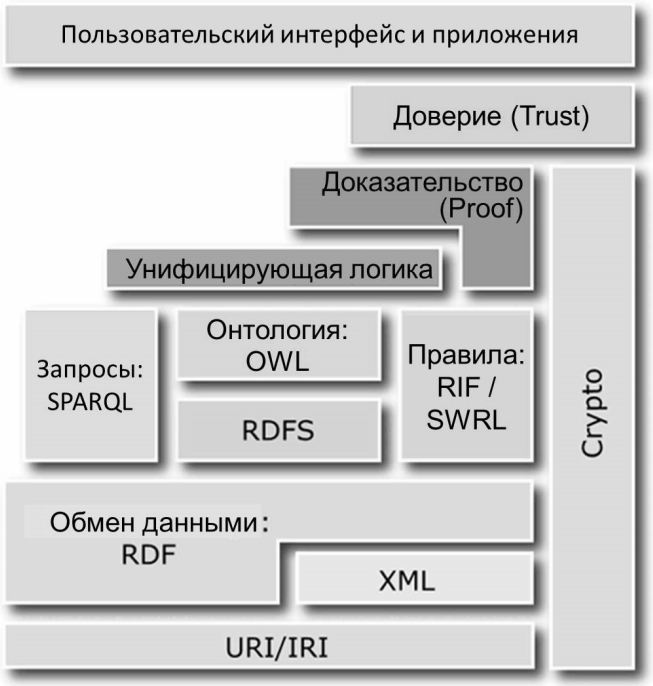


Рис. 5.30

«Слоеный пирог» семантического веба (Источник: http://www.semanticfocus.com/)

Все основные уровни диаграммы были описаны выше. Уровням «Ontology vocabulary» и «Logic» соответствуют OWL и RIF. Уровни Доказательств (Proof) и Доверия (Trust) на данный момент еще не незатронуты никакими стандартами. Здесь и возникает одно из существенных препятствий к реализации всей идеи: поддержка автоматической проверки корректности и правдивости информации. В самом деле, у многих поставщиков семантических описаний может возникнуть соблазн «обмануть» программу-агента, предоставив информацию, не соответствующую действительности, либо навязчивую рекламу, как это в настоящее время проделывается с поисковыми машинами, спам-фильтрами и т. п.

Рассмотрим более подробно отдельные стандарты.

RDF (Resource Description Framework)

RDF — язык представления информации о ресурсах WWW. В частности, RDF служит для представления метаданных, связанных с ресурсами Сети, таких как «заголовок», «автор», «дата последнего изменения страницы». Но RDF может ис-

пользоваться и для представления информации о ресурсах «второго типа», на которые можно только ссылаться (или идентифицировать в Сети при помощи URI), но невозможно непосредственно получить к ним доступ через Сеть.

Может оказаться, что в некоторых случаях для управления метаданными достаточно использовать XML и XML Schema (либо вообще ограничиться подэлементом HEAD элемента HTML). Но этот подход слабо масштабируется: при увеличении объема метаданных, усложнении их структуры управление метаданными, построенными на основе XML Schema, становится трудоемкой задачей, для решения которой и предназначен RDF.

Модель данных RDF. RDF-граф

Базовой структурной единицей RDF является коллекция троек (или триплетов), каждая из которых состоит из субъекта, предиката и объекта (S, P, O). Набор триплетов называется RDF-графом. В качестве вершин графа выступают субъекты и объекты, в качестве дуг — предикаты (или свойства). Направление дуги, соответствующей предикату в данной тройке (S, P, O), всегда выбирается так, чтобы дуга вела от субъекта к объекту (рис. 5.31).



Рис. 5.31
RDF-тройка

Каждая тройка представляет некоторое высказывание, увязывающее S, P и O.

Первые два элемента RDF-тройки (субъект и предикат) идентифицируются при помощи URI. Объектом же может быть как ресурс, идентифицируемый при помощи URI, так и RDF-литерал (значение). Пример представления информации с помощью RDF о курсе про онтологии и предствление знаний:

1) Тройки {субъект, предикат, объект}

{KR, имя, «Онтологии и представление знаний»}

{KR, читается, БК}

{KR, homepage, https://www.lektorium.tv/course/22781}

{БК, name, «Борис Конев»}

2) RDF-граф (рис. 5.32)

3) В XML-синтаксисе:

Пояснение: Каждый тег description описывает ресурс и каждый вложенный элемент или атрибут является свойством ресурса

<rdf:Description rdf:about="#KR">

<csclub: читается rdf:resource="#БК"/>

<csclub:имя>Онтологии и представление знаний</csclub:name>

<csclub: домашняя страница
rdf:resource="https://www.lektorium.tv/course/22781"/>

</rdf:Description>

<rdf:Description rdf:about="#БК">

<csclub:имя>Борис Конев</csclub:имя>

</rdf:Description>

4) В N3-синтаксисе (удобный для чтения человеком и расширяющий исходную модель данных RDF)

<ex:index.html> < csclub:имя > «Онтологии и представление знаний».

<ex:index.html> < csclub:читается> «БК»".

<ex:index.html> < csclub:домашняя страница >
«https://www.lektorium.tv/course/22781».



Рис. 5.32
Пример RDF-графа

RDFS

RDFS является семантическим расширением RDF. Он предоставляет механизмы для описания групп связанных ресурсов и отношений между этими ресурсами. Все определения RDFS выражены на RDF (поэтому RDF называется «самоописывающимся» языком). Новые термины, вводимые RDFS, такие как «домен», «диапазон» свойства, являются ресурсами RDF.

Система классов и свойств языка описания RDF-словарей похожа на систему типов объектно-ориентированных языков программирования, например, Java. Но RDF отличается от большинства таких систем тем, что здесь центральным аспектом является определение свойства, а не класса. Свойства в RDF определяются как пары (домен, диапазон). При этом домен представляет некоторое множество классов RDF, к которым данное свойство применимо, диапазон определяет допустимое множество ресурсов — значений свойства. Для сравнения: в Java определение класса имеет законченную форму (свойства класса выражаются в полях и методах класса). В RDF, напротив, описание класса всегда остается открытым (набор свойств класса определяется вне самого класса).

Пример. Определим свойство «Автор» с доменом «Документ» и диапазоном «Человек». В случае появления дополнительной информации о свойствах «Документа» нет необходимости изменять описание класса «Документ». Достаточно добавить новое свойство с соответствующим доменом.

Пример a-la RDF:
Класс ("Документ");
Класс ("Человек");
Свойство ("Автор", "Документ", "Человек").
Пример "a-la Java":
Класс "Документ"
{"Человек" "Автор"}

Можно заметить, что при изменении смысла свойств изменять придется именно их. При этом все классы, зависящие от изменяемых свойств, косвенно изменят свою семантику.

Основное преимущество такого подхода — в легкой расширяемости: добавление/удаление свойств интуитивно проще, чем управление множеством классов,

обладающих каждый своим индивидуальным набором свойств (как в ООП). Фактически, любой может расширять описание существующих ресурсов (лозунг Web: «Кто угодно может сказать что угодно о чем угодно!»).

Ресурсы могут объединяться в группы, называемые классами. Члены класса (здесь наиболее близкий термин — «экземпляры» или «объекты»" ООП) называются экземплярами класса. Сами классы также являются ресурсами и идентифицируются ссылками RDF-URI. Чтобы указать, что ресурс является экземпляром класса, используется свойство rdf:type ("rdf" здесь применен как префикс пространства имен).

В RDF определение класса или свойства (т.н. интенсионал) отделено от множества экземпляров класса и значений свойства (т. н. экстенсионала). Так, два класса с одинаковыми экстенсионалами считаются различными, если они имеют разные наборы свойств (интенсионалы). Рассмотрим множества

A = {0, 2, 4, 6, 8},

B = {x | x = 2k, k = 0..4, k — целое},

C — множество неотрицательных четных чисел, меньших 10.

В этом примере множество A полностью описывается своим экстенсионалом, множества B и C описываются интенсионалами, т. е. с использованием характеристических свойств данного множества. Множества, имеющие бесконечное число элементов, могут быть описаны только своим интенсионалом. Однако при использовании интенсионала могут возникнуть парадоксы[1]. Чтобы избежать их, в теории множеств вводятся дополнительные аксиомы. Примечательно, что RDF нарушает эти аксиомы. Классу RDF не запрещено быть экземпляром самого себя.

Группа ресурсов, являющихся классами, в RDFS описывается термином rdfs:Class.

На множестве классов определено отношение ПОДКЛАСС-НАДКЛАСС, описываемое RDFS-свойством rdfs:subClassOf. Семантика данного отношения состоит в том, что экстенсионал любого подкласса данного класса C целиком содержится (как множество) в экстенсионале самого класса C. Другими словами, если ресурс i является экземпляром класса C*, а класс C* является подклассом класса C, то i является экземпляром класса C.

Любой класс RDFS по определению является подклассом самого себя.

В спецификации по RDFS определены также списки, коллекции и контейнеры ресурсов, текстовые пометки и комментарии для создания удобных для чтения примечаний к ресурсам.

Проблемы с RDFS:

1. Недостаточно выразителен:
   - нет локализованных ограничений на домен и множество значений (множество значений свойства hasChild является людьми, когда применяется к человеку и слонами когда применяется к слонам);
   - нет ограничений на существование (*у всех* людей есть мать, которая тоже человек) и мощность (у матери-героини *больше двух детей*);
   - не определяются транзитивные, обратные или симметричные свойства (свойство isPartOf транзитивное, свойство hasPart обратно isPartOf, свойство *прикасаться* симметрично);
   - ...

[1] Например, парадокс Рассела: пусть множество *M* — множество всех множеств, не содержащих себя. Содержит ли *M* само себя? Если содержит, то оно не удовлетворяет своему определению – интенсионалу; если *M* не содержит себя, то оно удовлетворяет определению и, следовательно, должно себя содержать.

2. Трудности с логическим анализом:
- нет средств, поддерживающих нестандартную семантику;
- структурные алгоритмы.

OWL

OWL (Web Ontology Language, в аббревиатуре буквы намеренно переставлены местами, чтобы получилось английское слово «сова») — язык представления онтологий в Web. Фактически это словарь, расширяющий набор терминов, определенных RDFS. OWL-онтологии могут содержать описания классов, свойств и их экземпляров. Создание OWL — это ответ на необходимость представления знаний в Сети в едином формате. Исторически предшественником OWL был язык DAML+OIL, объединивший 2 инициативы: проект DAML (DARPA Agent Markup Language) и проект OIL (Ontology Inference Layer). Наиболее ранним проектом представления онтологий в Web был SHOE (Simlpe HTML Ontology Extensions). OWL с 2004 г. является рекомендацией W3C и объединяет лучшие черты своих предшественников.

В 2008 году появился язык OWL 2, представляющий собой доработанный язык OWL, совместимый с предыдущей версией. В OWL 2 добавлено несколько новых конструкций для расширения экспрессивности языка, в том числе конструкции по ограничению кардинальности, цепочки ролей и предикаты эксплицитных данных. OWL 2 также включает новую XML-сериализацию и набор подмножеств профилей с разнообразным применением и вычислительными свойствами. Неофициальный перевод стандарта OWL 2, выполненный Сергеем Щербаком, представлен на Web-странице http://shcherbak.net/translations/ru_owl2primer_shcherbak_net.html.

Базовые элементы OWL

Любая онтология имеет заголовок и тело. В заголовке содержится информация о самой онтологии (версия, примечания), об импортируемых онтологиях. За заголовком следует тело онтологии, содержащее описания классов, свойств и экземпляров.

Классы

В OWL введен новый термин — класс (owl:Class). Необходимость этого объясняется тем, что не все классы диалектов OWL DL и OWL Lite являются RDFS-классами (в этом случае owl:Class является подклассом rdfs:Class). В диалекте OWL Full подобных ограничений нет, и owl:Class фактически является синонимом rdfs:Class.

Для организации классов в иерархию используется свойство rdfs:subClassOf.

Особое место занимают два взаимодополняющих класса — owl:Thing и owl:Nothing. Первый из них является надклассом любого класса OWL, второй — подклассом любого класса OWL. Экземпляр любого класса OWL входит в экстенсионал класса owl:Thing. Экстенсионал класса owl:Nothing является пустым множеством.

OWL-класс может быть описан шестью способами:

1) идентификатором класса (URI);
2) перечислением всех экземпляров класса;
3) ограничением на значение свойства;
4) пересечением 2-х и более определений классов;
5) объединением 2-х и более определений классов;
6) дополнением (логическим отрицанием) определения класса.

Только первый способ определяет именованный класс OWL. Все оставшиеся определяют анонимный класс через ограничение его экстенсионала. Способ 2 явно перечисляет экземпляры класса, способ 3 ограничивает экстенсионал

только теми экземплярами, которые удовлетворяют данному свойству. Способы 4-6 используют теоретико-множественные операции (объединение, пересечение и дополнение) над экстенсионалами соответствующих классов, чтобы определить экстенсионал нового класса.

Описания класса являются строительными блоками для определения классов посредством аксиом.

Простейшая аксиома, определяющая именованный класс:

<owl:Class rdf:ID="Human"/>

Все, что постулирует эта аксиома, — существование класса с именем Human.

В OWL определены еще 3 конструкции, комбинируя которые, можно определять более сложные аксиомы классов:

- rdfs:subClassOf говорит о том, что экстенсионал одного класса (подкласса) полностью входит в экстенсионал другого (надкласса);
- owl:equivalentClass говорит о том, что экстенсионалы двух классов совпадают;
- owl:disjointWith говорит о том, что экстенсионалы двух классов не пересекаются. Иногда говорят, что таким образом определяются дизъюнктивные классы.

Свойства

В OWL выделяют две категории свойств: свойства-объекты (или объектные свойства) и свойства-значения. Первые связывают между собой индивиды (экземпляры классов). Вторые связывают индивиды со значениями данных. Оба класса свойств являются подклассами класса rdf:Property.

Для определения новых свойств как экземпляров owl:ObjectProperty или owl:DatatypeProperty используются аксиомы свойств.

Пример аксиомы:

<owl:ObjectProperty rdf:ID="hasParent"/>

Все, что постулирует данная аксиома, — существование некоторого свойства hasParent, связывающего экземпляры класса owl:Thing друг с другом.

Кроме того, OWL поддерживает следующие конструкции для построения аксиом свойств:

- конструкции RDFS: rdfs:subPropertyOf (определяет подсвойство данного свойства), rdfs:domain (определяет домен свойства) и rdfs:range (определяет диапазон свойства);
- отношения между свойствами: owl:equivalentProperty (определяет эквивалентное свойство) и owl:inverseOf (определяет обратное свойство);
- ограничения глобальной кардинальности: owl:FunctionalProperty (определяет однозначное свойство — однозначное отображение домена свойства на диапазон) и owl:InverseFunctionalProperty (обратно функциональное свойство, т. е. определяет, что свойство, обратное данному свойству, является однозначным);
- логические характеристики свойства: owl:SymmetricProperty (определяет свойство как симметричное) и owl:TransitiveProperty (определяет транзитивное свойство).

Индивиды (экземпляры классов)

Индивиды определяются при помощи аксиом индивидов (т. н. фактов). Рассмотрим два вида фактов:

факты членства индивидов в классах и факты о значениях свойств индивидов;
факты идентичности/различности индивидов.

Пример аксиом индивидов первого вида:

<Балет rdf:ID="ЛебединоеОзеро">
<имеетКомпозитора rdf:resource="#Чайковский"/>
</Балет>

Данная аксиома постулирует сразу 2 факта: (1) существует некоторый индивид класса Балет, имеющий имя ЛебединоеОзеро; (2) этот индивид связан свойством имеетКомпозитора с индивидом Чайковский (который определен где-то в другом месте). Первый факт говорит о членстве в классе, второй — о значении свойства индивида.

Аксиомы второго вида необходимы для суждения об идентичности индивидов. Дело в том, что в OWL не делается никаких предположений ни о различии, ни о совпадении двух индивидов, имеющих различные идентификаторы URI. Подобные утверждения выражаются аксиомами идентичности с помощью следующих конструкций:

- owl:sameAs постулирует, что две ссылки URI ссылаются на один и тот же индивид;
- owl:differentFrom постулирует, что две ссылки URI ссылаются на разные индивиды;
- owl:AllDifferent предоставляет средство для определения списка попарно различных индивидов.

Дополнительная информация на русском языке по практическим аспектам использования стандартов семантического веба может быть найдена в [Горшков, 2016], на сайте http://shcherbak.net/publikacii/ и http://forum.semanticfuture.net/, теоретические аспекты рассмотрены на материалах симпозиума «Онтологическое Моделирование 2010» http://ontology.ipi.ac.ru/index.php/Онтологическое_Моделирование_2010.

Глава 6

ПРИМЕНЕНИЕ ОНТОЛОГИЙ

6.1. ОБЗОР СПОСОБОВ/СЦЕНАРИЕВ ПРИМЕНЕНИЯ ОНТОЛОГИЙ

В статье [Kozaki et al, 2008] было выделено девять наиболее популярных способов/сценариев использования онтологий и проведен анализ их распространенности.

(1) Общий словарь

Онтология предоставляет общий словарь для описания информационных ресурсов, а также общения сотрудников. Общий словарь снижает неоднозначность соответствия между знаком, понятием и объектом реального мира (денотатом). Данный способ/сценарий использования был рассмотрен выше (см. главу 5).

(2) Семантический поиск

При использовании онтологий поисковая машина будет выдавать только такие ресурсы, где упоминается в точности искомое понятие, а не произвольные страницы, в тексте которых встретилось данное многозначное ключевое слово. Сегодняшние поисковые системы зачастую выдают бесчисленное множество совершенно не относящихся к запросу «хитов», обрекая пользователя на длительный ручной отбор материала. Например, если вы ввели для поиска слово «ягуар», то компьютеру совершенно непонятно, имеете ли вы в виду автомобиль, животное или напиток. Вся проблема в том, что для компьютера слово «ягуар» не имеет четкого смысла или, другими словами, семантического содержания. Онтология позволяет указать контекст для поиска. Указывая в поисковой строке слово «ягуар» необходимо дополнительно указать, идет ли речь об автомобиле, животном или напитке. На техническом языке это может называться сужением области поиска. Онтологии дают возможность производить запросы на основе понятий, а не на основе совпадения строк. Например, если пользователь задаст вопрос: *«Какие транспортные средства производятся в России?»*, то он получит из базы ответ, в который попадут автомобили (= подкласс транспортных средств), производимые во Всеволожске (= город, который находится в России).

Примером семантического поиска является система Ask (Ask.com). Ask является системой вопрос-ответ, т. е. пользователь задает вопрос, а система отвечает, как это делается при общении между людьми (рис. 6.1).

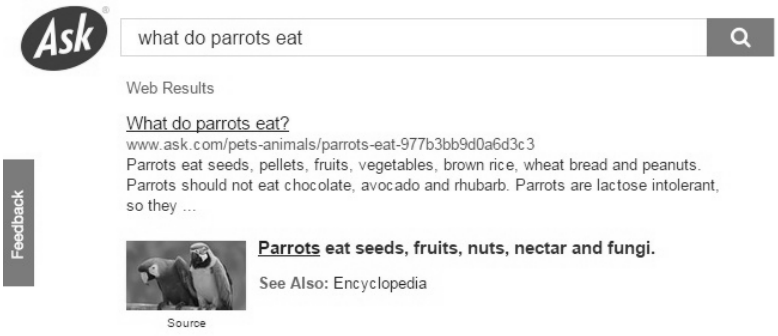


Рис. 6.1

Пример выполнения поиска в системе Ask

В тексте, который система Ask нашла на некотором сайте, выделяется та его часть, которая содержит ответ на заданный запрос, благодаря чему пользователю не надо читать всю страницу для нахождения нужной информации, как в традиционных подходах.

Система на основе онтологии может также предлагать пользователю возможные интерпретации его поискового запроса на основе анализа взаимосвязей понятий, а также распределять результаты поиска по различным интерпретациям запроса пользователя. Например, для запроса «организационное проектирование» система может уточнить интересуется ли пользователь вопросом проектирования организаций (область менеджмента) или проектированием любых объектов, но в рамках организации.

(3) Указатель (Каталог)

В данном применении онтологии используются в качестве структурированного указателя на ресурсы (рис. 6.2). Онтология предоставляет возможности для навигации, по аналогии с папками на домашнем компьютере или каталогом ресурсов. Часто пользователь не может точно сформулировать запрос, например, из-за новизны предметной области для него или нечеткости информационной потребности. В таком случае использование указателя (каталога) может быть более эффективной стратегией, нежели формирование запросов. Семантические описания также могут направлять пользователя в процессе поиска информации, предлагать детализацию, более общий взгляд, смежные области. В случае малого количества результатов поиска, иерархия понятий из онтологии может быть использована для расширения запроса добавлением более общих и более частных понятий, например, добавлением понятий «транспортное средство» и «внедорожник» к понятию «автомобиль».

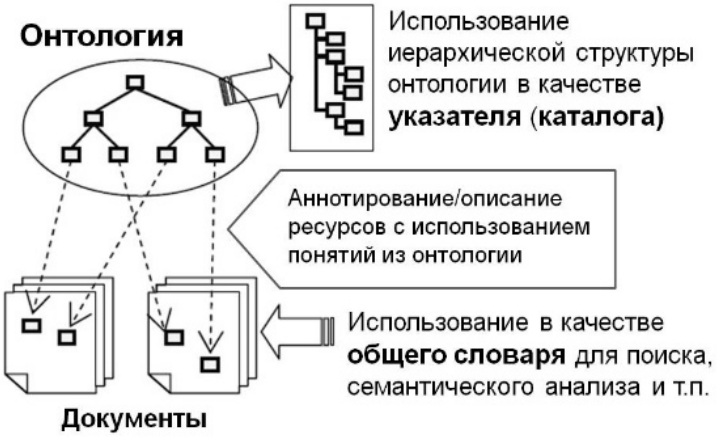


Рис. 6.2

Использование онтологии в качестве общего словаря и указателя

Конкретными примерами приложений данной группы являются семантическая система навигации Magpie [Dzbor et al, 2003] и система Ontogator [Makela et al, 2006], в которой указатель основан на представлениях (views) онтологии и интегрирован с контекстными структурами.

(4) Схема данных

Онтология может использоваться для описания и спецификации схемы базы данных (рис. 6.3).

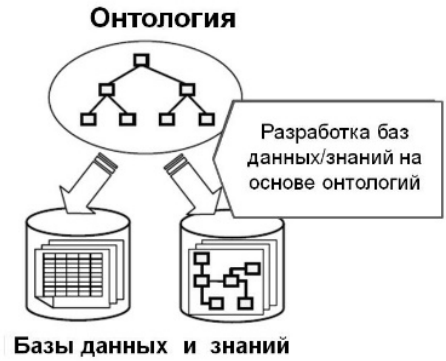


Рис. 6.3

Использование для описания схемы базы данных и структуры модели знаний

Конкретными примерами в данной группе является схема метаданных единого научного информационного пространства Российской академии наук [Бездушный и др., 2004], а также масштабные базы данных по исследованиям [Fox et al, 2006].

(5) Посредник для интеграции информации

Онтологии позволяют интегрировать информацию, находящуюся в различных источниках, — документы, базы данных и знаний, сотрудники. Потребителями интегрированной информации могут быть как люди, так и программы (или программные агенты). Возможны два варианта интеграции (рис. 6.4): (a) потребители работают с одной общей онтологией, с помощью которой описываются все информационные ресурсы, (b) потребители имеют свои собственные частные онтологии, между элементами которых установлены соответствия. В целом, доступ к информации может быть организован за счет ее совместного использования потребителями или посредством обмена информацией между ними.

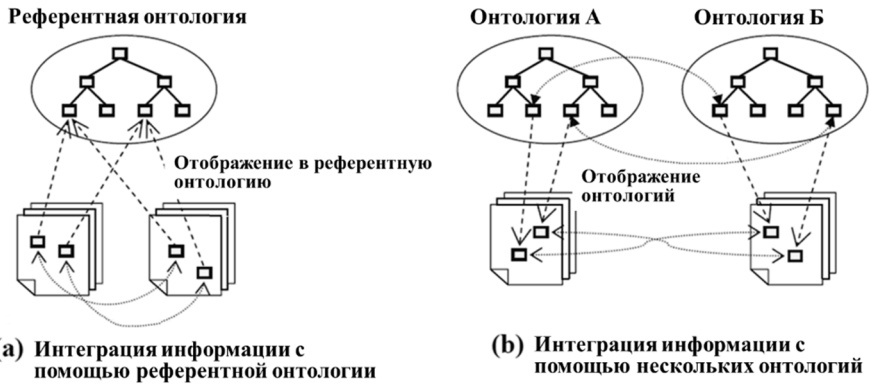


Рис. 6.4

Использование в роли посредника при интеграции информации

Задачи интеграции поддерживают специальные модули отображения (mapping) онтологий (рис. 5.24).

Так как сейчас все большее внимание уделяется распределенным информационным системам моделирования организаций, которые обмениваются знаниями друг с другом, совместно используют знания, реализуют возможности повторного использования знаний, то возникает вопрос о представлении знаний в формальной форме. Несмотря на то, что такие системы имеют стилистические и организационные различия в представлении знаний, необходимо, чтобы они сохраняли декларативное содержание этих знаний и, таким образом, могли формулировать запросы к другим информационным системам, обмениваться с ними данными и знаниями. В данном случае возникает проблема необходимости организации взаимодействия с учетом нескольких созданных онтологий. В результате, онтология становится средством предоставления возможности взаимодействия между компьютерными системами независимо от индивидуальных технологий систем, архитектуры информации и областей применения. Вопросы использования онтологий для интеграции информации подробно рассмотрены в работах [Когаловский, 2012; Тузовский, Ямпольский, 2011; Черняк, 2009]. На рисунке 6.5 представлен пример интеграции информации о сотрудниках компании с помощью онтологии.

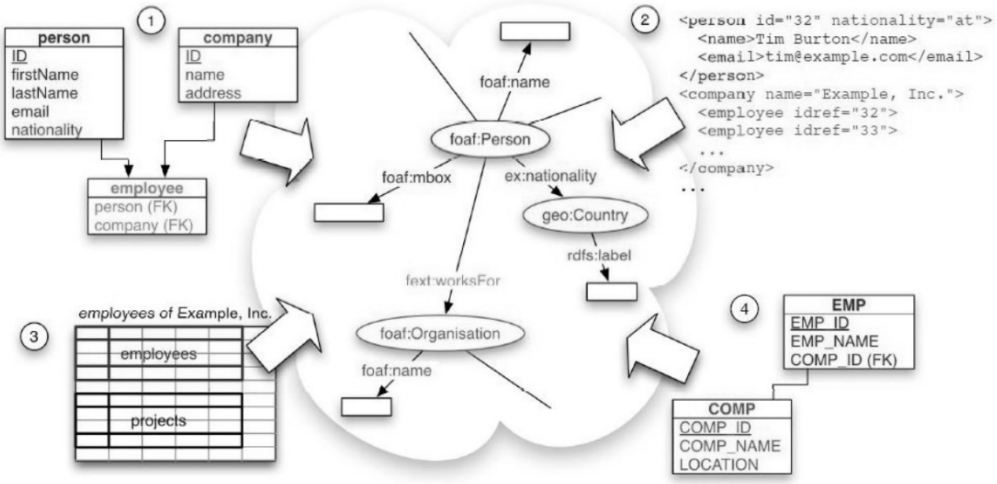


Рис. 6.5

Интеграция информации из баз данных, электронных таблиц и XML-файлов с помощью онтологии

В качестве реального примера использования онтологий для интеграции знаний можно рассмотреть интеграцию знаний о бизнес-процессах, представленных в разных нотациях. В рамках проекта SUPER (http://www.ip-super.org/) были разработаны онтологии, соответствующие стандартным нотациям (EPC, BPMN, BPEL), обобщенная онтология моделирования бизнес-процессов, верхнеуровневая онтология процессов, а также схемы соответствия между ними. Данные онтологии и схемы, заложенные в программную среду, позволяют преобразовывать модели процессов из одной нотации в другую, а также осуществлять интегрированный поиск по всем моделям. Данный пример подробнее рассмотрен в следующем разделе 6.2.

Применение онтологий для интеграции инженерных данных рассмотрено в [Агроскин и др., 2013].

(6) Семантический анализ

К знаниям, представленным с помощью онтологий, могут быть применены автоматизированный логический вывод и семантическая обработка, которые позволяют проводить семантический анализ и создавать новые знания. Одним из распространенных примеров вывода является автоматическая классификация объектов с помощью инструментов логического вывода (inference engine). Например, зная, что начинкой «American Pizza» является сыр моцарелла, система автоматически относит данную пиццу к классу сырных (chessy) (рис. 6.6).

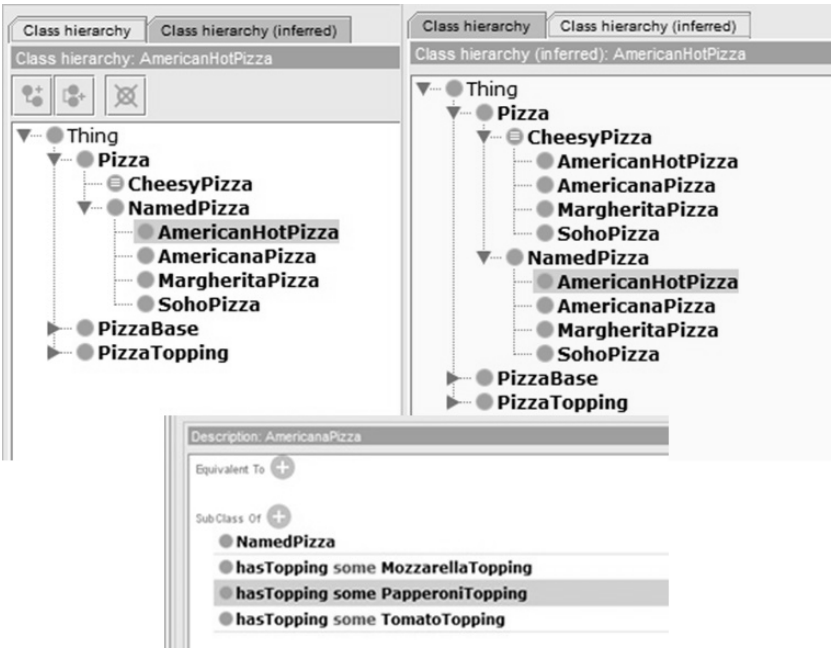


Рис. 6.6

Пример автоматизированной классификации

Интеграция онтологий с правилами позволяет проводить автоматическую классификацию, проверку соблюдения политик и может использоваться для поддержки принятия решений. Такое применение онтологий детально рассматривается в проекте OntoRule (http://ontorule-project.eu/).

Использование информации программными агентами при решении рутинных умственных задач, которое было предложено в рамках концепции семантического веба, также относится к семантическому анализу на основе онтологий. Семантический веб (Semantic Web) привносит смысловую структуру в содержание Web-страниц. Это не какая-то отдельная сеть, а расширение уже существующей, такое, что в ней информация снабжена точно определенным смыслом, позволяющим человеку и машине успешно взаимодействовать.

Сценарий работы таких программных средств представлен в основополагающей статье идеологов семантического веба — Тима Бернерса-Ли и Джеймса Хендлера, опубликованной в 2001 году в журнале Scientific American:

«Пит поднял трубку, звонила его сестра Люси из кабинета врача: „Моей маме нужно попасть на прием к врачу, а потом ей требуется пройти несколько сеансов

физиотерапии. Примерно два раза в неделю. Я сейчас поручу своему агенту записать нас на прием к врачу". Пит сразу же согласился подвезти ее маму на своей машине. В кабинете врача Люси дала указания своему электронному агенту семантического веба через свой портативный веб-браузер. Агент сразу же получил информацию о *назначенном маме лечении* от агента ее врача, просмотрел несколько перечней *медицинских учреждений*, оказывающих подобные услуги, проверил, какие из них *подходят* по ее страховому полису и находятся в *окрестности 20 миль* от ее *дома* и при этом имеют *рейтинг — «отличный»* либо *«очень хороший»* согласно достоверным рейтинговым источникам. Далее агент попытался сопоставить имеющиеся *часы приема* врачей (предоставленные агентами соответствующих медучреждений на их веб-сайтах) с расписаниями Пита и Люси. (Слова, выделенные курсивом, указывают на термины, семантика или смысл которых была предоставлена агентам посредством семантического веба.) Уже через несколько минут агент представил им план. Питу он не понравился — университетская клиника расположена на другой стороне города, и ему пришлось бы возвращаться обратно как раз в час пик. Он попросил своего агента повторить поиск с более строгими предпочтениями относительно *места* и *времени*. Тем временем агент Люси, *полностью доверяя* агенту Пита в рамках данной задачи, автоматически помогал, предоставляя тому права доступа и ссылки на уже добытую им информацию. Практически мгновенно был предложен новый план: найденная клиника была уже гораздо ближе, и часы приема более ранние».

(7) Извлечение информации (Information Extraction)

Данное применение онтологии позволяет выявлять экземпляры класса и отношения между ними в тексте на естественном языке, а также записывать выявленные факты в семантическое хранилище (базу знаний) (рис. 6.7).



Рис. 6.7

Семантическая разметка текста на естественном языке

На рисунке 6.8 указано, что в результате семантической разметки фразы «XYZ планирует инвестировать 150 млн долларов в строительство завода в Исландии» некоторые ее элементы позволяют наполнить семантическое хранилище. В свою очередь, информация в нем удобна для дальнейшей обработки и анализа (см. применения выше).

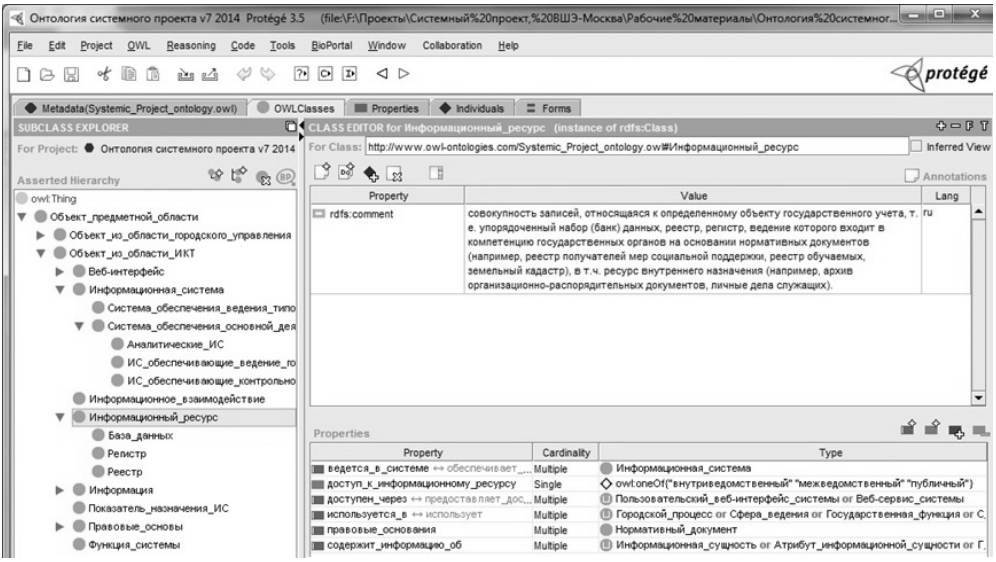


Рис. 6.8

Онтология, задающая структуру базы знаний

(8) Спецификация структуры моделей знаний

Онтология используется для спецификации структуры моделей знаний, т. е. баз знаний. На рисунке 6.8 приведена онтология, которая задает структуру для онтологической модели (модели знаний), представленной на рисунке 6.9 и для отчета из этой модели (см. рис. 6.10).

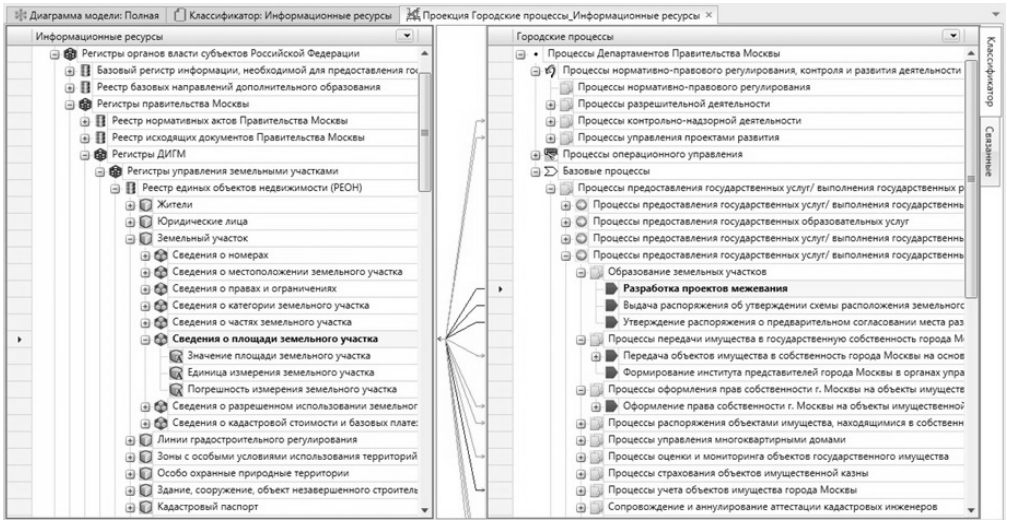


Рис. 6.9

Наполнение содержимого базы знаний: среда моделирования

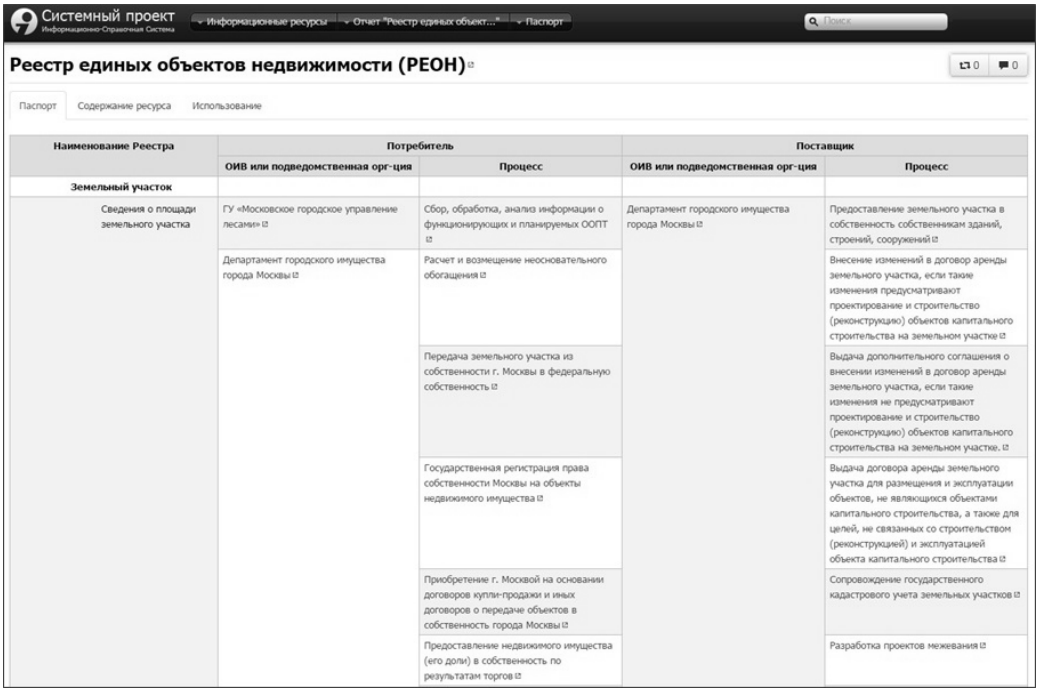


Рис. 6.10

Представление содержимого базы: табличный отчет на портале

Данный пример использования онтологий для моделирования комплексов городского хозяйства более подробно рассмотрен в разделе 6.3.

(9) Систематизация знаний

Структурировать знания об окружающем пространстве — для отдела персонала это может быть каталог компетенций сотрудников, для службы техобслуживания и ремонта — справочник оборудования и инструментов, для маркетологов — сегменты рынка, для бизнес-аналитика — справочник бизнес-процессов. Ориентиром в этой области применения онтологий на предприятии является комплексное структурированное описание архитектуры предприятия (enterprise architecture).

На практике приложение часто сочетают указанные выше примеры использования. Например, выполнения простых автоматических рассуждений и интеграция информации могут повышать качество результатов поиска информации. Онтологии дают возможность получения не заданных явно знаний из информационных хранилищ путем логического вывода — поиск «скрытой информации». Например, пользователь системы может задать вопрос: *Какие поставки продукции находятся сейчас в состоянии риска?* В ответ на такой вопрос система в одной онтологии тарифов определит, что с учетом текущих условий (например, географических или погодных) существуют риски, связанные с перевозкой овощей и фруктов. А в другой базе или онтологии деклараций по перевозке груза определит, что в декларации № А345 указаны арбузы, которые являются подклассом «Овощей и фруктов» (см. рис. 6.11). В результате система сможет дать конкретный ответ на поставленный вопрос: *Поставка COSCO #A345*

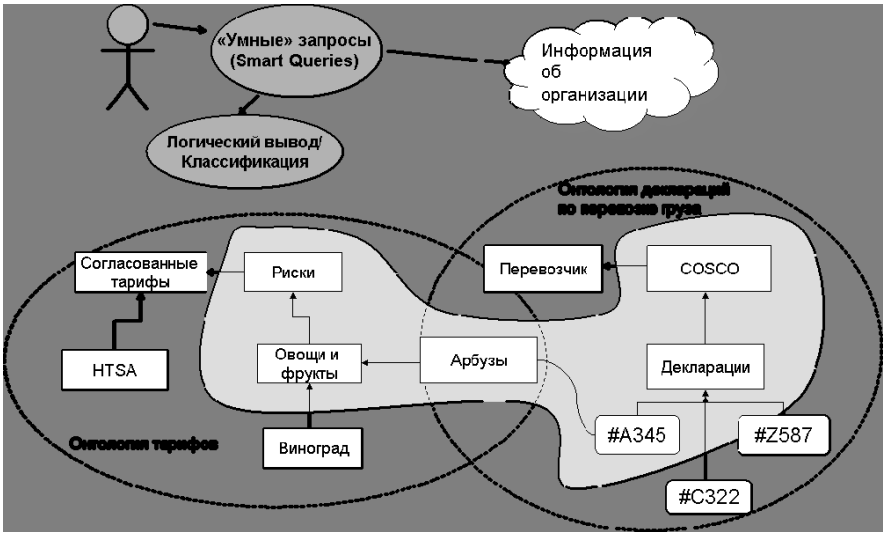


Рис. 6.11

Интеллектуальный поиск на основе логического вывода и интеграции информации

6.2. МОДЕЛИРОВАНИЕ ПРЕДПРИЯТИЙ НА ОСНОВЕ ОНТОЛОГИЙ

В соответствии с [Vernadat, 2002] моделирование предприятия — деятельность по формализации знаний об устройстве и функционировании предприятия для создания ценности для повышения эффективности и создания дополнительной ценности. В качестве конечных целей может выступать явное представление таких фактов и знаний, которые добавляют ценность для организации или могут быть распределены между пользователями и программными приложениями для улучшения деятельности организации.

К настоящему моменту существует ряд примеров использования онтологий в моделировании и управлении предприятиями.

1. Enterprise Project [Stader, 1996; Uschold et al, 1998].
2. Toronto Virtual Enterprise (TOVE) project [Fox, 1992; Fox et al, 1997; Gruninger et al, 2000].
3. SUPER Project [Hepp, 2007].
4. Система ОРГ-Мастер [Григорьев, Кудрявцев, 2012; Kudryavtsev, Grigoriev, 2011; Grigoriev, Kudryavtsev, 2013].
5. Essential project [Mayall A., Carter, 2015].

Далее приводится краткое описание указанных примеров.

Enterprise Project и Онтология Предприятия (Enterprise Ontology)

Целью проекта Enterprise (http://www.aiai.ed.ac.uk/project/enterprise/), выполненного совместно представителями университета Эденбурга, IBM, Unilever и Lloyd's Register [Stader, 1996; Uschold et al, 1998], являлось моделирование бизнес-среды для поддержки менеджеров в принятии взвешенных стратегических, тактических и операционных решений. В свою очередь основная роль моделирования в проекте Enterprise состояла в формировании комплексного взгляда на организацию. Для достижения, использования и поддержки такого комплекс-

ного взгляда на организацию требуются мощные средства интеграции, коммуникации, гибкости и поддержки. В результате проекта Enterprise был разработан Enterprise Tool Set (ETS) для информационной поддержки предоставления комплексного взгляда на предприятие.

В качестве основы моделирования предприятия использовались модели процессов, которые обеспечивают процессно-ориентированный взгляд и могут быть реализованы в исполнительной (*running*) системе. Для разработки моделей процессов был создан Procedure Builder. В большинстве организаций существует множество используемых приложений, и в проекте было решено поддерживать интеграцию существующих инструментов с минимальными их изменениями, вместо того, чтобы заменять существующие инструменты и их интерфейсы. Для реализации такого подхода был создан Agent Toolkit — архитектура, основанная на агентах (*agent-based architecture*), совмещенная с библиотекой, поддерживающей процесс добавления инструментов к системе.

Кроме поддержки интеграции инструментов, предложена поддержка реализации (исполнения) процессов. Task Manager обеспечивает как интеграцию инструментов, так и поддержку самих моделей процессов. Кроме того, он обеспечивает поддержку реализации процессов во времени (*agenda-style*).

Для достижения такой высокоуровневой интеграции и обеспечения эффективной коммуникации компонент необходимо согласование используемых понятий. Для этого была разработана онтология предприятия (*Enterprise Ontology*).

В результате состав Enterprise Tool Set следующий (рис. 6.12):

- Procedure Builder для разработки моделей процессов;
- Agent Toolkit для поддержки разработки агентов;
- Task Manager для интеграции, визуализации и поддержки реализации;
- Enterprise Ontology для коммуникации.

Роль онтологии в Enterprise Project:

- Интеграция информации, используемой разными приложениями;
- Описание функциональных возможностей пользовательских приложений;
- Онтология в качестве семантики языка моделирования бизнес-процессов (по заявлениям самих участников, фактически, язык моделирования бизнес-процессов опирается в большей степени на свои собственные, не связанные с онтологией, понятия).

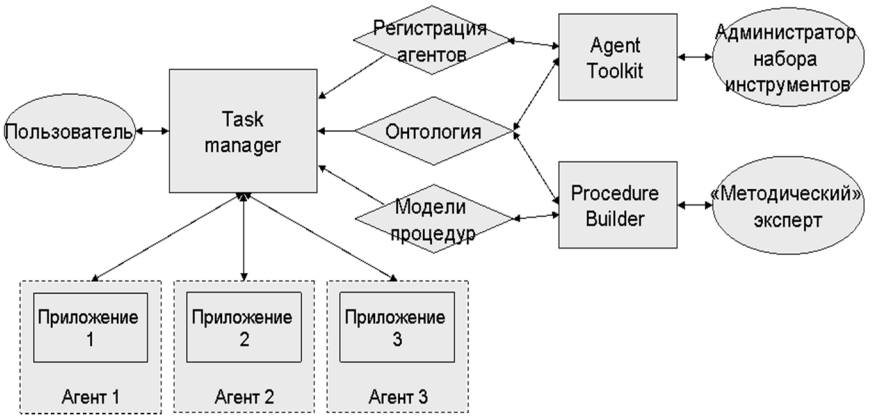


Рис. 6.12

Архитектура Enterprise Tool Set

Онтология Предприятия состоит, главным образом, из набора определенных терминов, явно представленных в онтологии, для которых даются определения, и устанавливается взаимосвязь с другими терминами онтологии.

Для лучшего понимания Онтологии Предприятия полезно знать, как термины и понятия, явно представленные в онтологии, соотносятся с терминами и понятиями, широко используемыми в другом контексте (например, в других онтологиях). Поэтому в приложении к онтологии перечисляются родственные термины, которые достаточно широко распространены, но не определены в Онтологии Предприятий. Где это возможно, указывается соотношение между этими терминами и терминами из Онтологии Предприятия. Эти родственные термины разделены на три категории: Синонимы, Близкие термины, Другие общепринятые термины.

В некоторых местах онтология дополнена аксиомами.

Все термины в онтологии попадают в 5 верхнеуровневых разделов, отражающих разные аспекты предприятия:

- Метаонтология и Время,
- Активность, План, Способность и Ресурс,
- Организация,
- Стратегия,
- Маркетинг.

В приводимой ниже таблице 6.1 перечислены все определенные в Онтологии Предприятия понятия, объединенные в основные группы.

Таблица 6.1

Общая модель Онтологии Предприятия

Активности и процессы	**Организация**	**Стратегия**	**Маркетинг**	**Время**
Активность	Лицо	Цель-назначение	Продажа	Ось времени
Спецификация активности	Машина	Иметь цель-назначение	Потенциальная продажа	Временной интервал
Выполнять	Корпорация	Планируемая цель-назначение	Для продажи	Момент времени
Выполненная спецификация активности	Партнерство	Держатель цели-назначения	Коммерческое предложение	
Т-Начала	Партнер	Стратегическая цель-назначение	Вендор	
Т-Окончания	Юридическое лицо	Задача	Фактический покупатель	
Предварительное условие	Подразделение	Виденье	Потенциальный покупатель	
Эффект	Управлять	Миссия	Покупатель	
Делатель	Делегировать	Цель-результат	Дистрибьютор	
Подактивность	Управленческая связь	Обеспечивать достижение	Продукт	
Полномочия	Право собственности	Стратегия	Запрашиваемая цена	
Владелец активности	Non-Legal собственность	Стратегическое планирование	Цена продажи	
Событие	Собственность	Стратегическое действие	Рынок	
План	Владелец	Решение	Переменная сегментации	

Продолжение табл. 6.1

Активности и процессы	**Организация**	**Стратегия**	**Маркетинг**	**Время**
Подплан (sub-plan)	Актив	Допущение	Рыночный сегмент	
Планирование	Заинтересованная сторона	Критическое допущение	Маркетинговое исследование	
Спецификация процесса	Трудовой договор	Некритическое допущение	Бренд	
Способность	Доля/Акция	Фактор влияния	Имидж	
Умение	Собственник/ Акционер	Критический фактор влияния	Характеристика	
Ресурс		Некритический фактор влияния	Потребность	
Распределение ресурсов		Критический фактор успеха	Потребность рынка	
Ресурс-заменитель		Риск	Продвижение	
			Конкурент	

Далее, для примера, рассмотрены термины из раздела «Активности и процессы».

Центральным термином является АКТИВНОСТЬ (ACTIVITY). Он охватывает понятие всего, что предполагает реальное делание, в частности, включая действие. АКТИВНОСТЬ может иметь место в прошлом и происходить в настоящем. Этим термином можно также обозначать гипотетическую будущую АКТИВНОСТЬ.

Тем не менее, существует необходимость непосредственно указывать спецификацию или план АКТИВНОСТЕЙ. Это называется СПЕЦИФИКАЦИЯ АКТИВНОСТИ (ACTIVITY SPECIFICATION). Как и рецепт, она на некотором уровне детализации описывает одну или более АКТИВНОСТЕЙ. СПЕЦИФИКАЦИЯ ВЫПОЛНЕННОЙ АКТИВНОСТИ должна иметь нечто сделанное, соответствующее АКТИВНОСТИ.

Концепция АКТИВНОСТИ тесно связана с понятием ДЕЛАТЕЛЬ (DOER), который ВЫПОЛНЯЕТ (EXECUTES) СПЕЦИФИКАЦИЮ АКТИВНОСТИ, совершая заданные АКТИВНОСТИ. ДЕЛАТЕЛЬ может быть ЛИЦОМ (PERSON), ПОДРАЗДЕЛЕНИЕМ (ORGANISATIONAL UNIT) или МАШИНОЙ (MACHINE). Эти термины определены в секции Организация и вместе могут собирательно обозначаться как [ПОТЕНЦИАЛЬНЫЕ] ИСПОЛНИТЕЛИ ([POTENTIAL] ACTORS) (или АКТОРЫ).

Возможность ПОТЕНЦИАЛЬНОГО ИСПОЛНИТЕЛЯ быть ДЕЛАТЕЛЕМ обозначается как СПОСОБНОСТЬ (CAPABILITY) или УМЕНИЕ (SKILL), если ДЕЛАТЕЛЬ — ЛИЦО. ИСПОЛНИТЕЛЬ может выполнять иные РОЛИ (ROLE) в отношении АКТИВНОСТИ, такие, например, как ВЛАДЕЛЕЦ АКТИВНОСТИ (ACTIVITY OWNER).

С АКТИВНОСТЬЮ также тесно связан РЕСУРС (RESOURCE), представляющий собой нечто, могущее быть использованным или израсходованным АКТИВНОСТЬЮ. АКТИВНОСТЬ также может иметь выходы или ЭФФЕКТЫ (EFFECTS). АКТИВНОСТЬ связана с ВРЕМЕННЫМ ИНТЕРВАЛОМ (TIME INTERVAL). АКТИВНОСТЬ может занимать малое или большое время и быть простой или сложной. Сложная АКТИВНОСТЬ может раскладываться на множество ПОД-АКТИВНОСТЕЙ (SUB-ACTIVITIES).

СПЕЦИФИКАЦИЯ АКТИВНОСТИ совместно с ПЛАНИРУЕМОЙ ЦЕЛЬЮ-НАЗНАЧЕНИЕМ (INTENDED PURPOSE), (определяемой в разделе Стратегия) называется ПЛАНОМ (PLAN). Понятие о возможности многократно ВЫПОЛНЯТЬ один и тот же ПЛАН описывается термином СПЕЦИФИКАЦИЯ ПРОЦЕССА (PROCESS SPECIFICATION).

Управление осуществлением АКТИВНОСТЕЙ важно для предприятия. Для этой цели мы определяем понятие ПОЛНОМОЧИЯ (AUTHORITY) ИСПОЛНИТЕЛЯ быть вправе выполнять одну или несколько АКТИВНОСТЕЙ (например, как указано в ПЛАНЕ).

Подробнее об Enterprise Project см. [Гурьянова и др., 2011].

Проект TOVE

В начале 1990-х годов в департаменте Industrial Engineering университета Торонто сформировалась Канадская школа исследований в области моделирования предприятий под руководством М. Фокса. Данной командой был реализован проект TOVE (TOronto Virtual Enterprise) [Fox, 1992; Fox et al, 1997; Gruninger et al, 2000]. Основные цели проекта TOVE, который был поддержан, в частности, агентством Natural Science Engineering Research Council, компаниями DEC, Micro Electronics Computer Technology Corp. и Spar Aerospace Ltd., состояли:

- в формализации знаний в области инжиниринга предприятий (Enterprise Engineering), что должно было обеспечить не только точную спецификацию интуиции практиков, но и основу для автоматизации отдельных задач инжиниринга предприятий;
- в интеграции знаний в рамках программных средств поддержки инжиниринга предприятий на базе формального представления структуры организации и ее деятельности (поведения);
- в обеспечении разработанных моделей средствами многоаспектной визуализации результатов.

Также одним из его важнейших результатов была разработка методологии онтологического инжиниринга, известной в настоящее время как методология Грюнингера и Фокса (Gruninger-Fox's methodology) [Gruninger, Fox, 1994]. Как и многие другие методологии и методы онтологического инжиниринга первого поколения, эта методология была сформирована на основе опыта разработки конкретной онтологии, ориентированной на предметную область моделирования бизнес-процессов и по существу предполагает создание онтологии как логической модели знаний.

На базе методологии Грюнингера — Фокса были разработаны:

- *Enterprise Design Workbench* — среда проектирования, позволяющая пользователю анализировать различные конструкции предприятия (enterprise design). Важной функциональной особенностью Enterprise Design Workbench была поддержка сравнительного анализа альтернативных конструкций предприятия (enterprise design) и руководство проектировщиком;
- *Integrated Supply Chain Management Project agents* — организация цепочки поставок как сети взаимодействующих интеллектуальных агентов, каждый из которых выполняет одну или несколько функций в цепочке снабжения и координирует свои действия с другими агентами.

Архитектура системы инжиниринга предприятий, предложенная и реализованная в рамках проекта TOVE, представлена на рисунке 6.13.

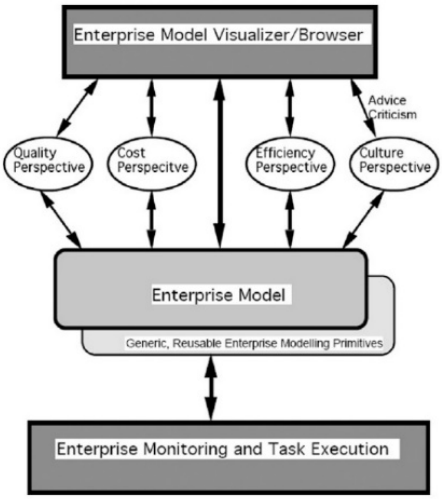


Рис. 6.13

Архитектура системы инжиниринга предприятий в проекте TOVE

Как следует из рисунка 6.13, система инжиниринга предприятий здесь состоит из четырех основных компонент: модель предприятия, основанная на здравом смысле (common-sense enterprise model), модулей управления (advisors), визуализации (visualization) и информационных агентов (information agents).

Базисом системы является модель предприятия, которая поддерживает множество основных повторно используемых знаний о предприятии. В частности, эта модель включает модели представления процессов (processes), активностей (activities), времени (time), причинности (causality), ресурсов (resources), качества (quality) и цены (cost). Модель предприятия используется всеми компонентами системы, за счет чего обеспечивается общая (разделяемая) терминология и общее множество ограничений.

В бизнесе необходимы модели различных его аспектов, таких как, например, эффективность, качество, цена и др. Поэтому любая система инжиниринга предприятий должна поддерживать хорошо определенное представление и управление разными аспектами бизнеса. В рамках архитектуры TOVE для этого предназначены модули управления (advisors), «способные» анализировать текущую ситуацию в бизнесе и принимать правильные решения в условиях наличия нескольких альтернатив.

В дополнение к представлению знаний о предприятии в системе полезно иметь средства визуализации ситуации в бизнесе с различных точек зрения, что предполагает наличие графических средств представления модулей управления и взаимодействий между ними, включая мониторинг рабочих потоков.

И, наконец, у системы должно быть окружение периода исполнения (execution environment), поддерживающее базы данных, библиотеки исполняемых модулей и т. п.

Таким образом, архитектура системы инжиниринга предприятий в проекте TOVE является развитой технологической платформой для спецификации бизнеса и моделирования протекающих здесь процессов.

Понятно, что моделирование является ключевым компонентом в спецификации предприятия. Поэтому важнейшей целью исследований, которые проводились в

проекте TOVE, были работы по созданию общих повторно используемых представлений знаний о предприятии, которые могли бы использоваться в разных конкретных предприятиях. Такие знания о предприятии представляются в Enterprise Ontology TOVE [Fox, et al., 1997], которая содержит формальное представление сущностей и их свойств, формируя таким образом общую разделяемую терминологию по объектам интереса в данной предметной области. Онтология TOVE поддерживает мощное и точное представление общих знаний (активности, процессы, ресурсы, время, причины) и знания, специфичные для бизнеса (цена, качество, оргструктура и др.). Пирамида знаний в проекте TOVE показана на рисунке 6.14.



Рис. 6.14

Пирамида знаний в проекте TOVE

Базовые сущности в модели TOVE представляются объектами с их свойствами и отношениями. Объекты объединяются в таксономии. Объекты, атрибуты и отношения специфицируются, где это возможно, в исчислении предикатов первого порядка. С учетом вышесказанного, онтология определяется путем спецификации объектов, которые представляются константами и/или переменными соответствующего языка представления знаний, и последующей идентификации свойств объектов и отношений между объектами, которые представляются в языке представления знаний предикатами.

На следующем шаге в исчислении предикатов первого порядка определяется множество аксиом, фиксирующих ограничения на объекты и предикаты в онтологии. Построенное множество аксиом формирует микротеорию и обеспечивает декларативную спецификацию различных задач, которые должны решаться в рамках разрабатываемой модели. Аксиомы микротеории «позволяют» модели выводить ответы на вопросы, на которые обычно можно ответить на основе общего понимания бизнес-процессов. Однако для теоретического обоснования этой интуиции необходимо иметь формальное доказательство свойств микротеории, что позволяет понять область охвата и ограничения используемого подхода. В проекте TOVE для этого используются вопросы компетенции (competency questions), которые служат для характеризации различных онтологий и микротеорий в модели бизнеса. Микротеория в данном подходе должна содержать необходимое и достаточное множество аксиом для представления вопросов и ответов на эти вопросы, что позволяет утверждать, что микротеория адекватна поставленной задаче. Заметим, что такой подход к обоснованию корректности разработанной онтологии был впервые предложен для инжиниринга бизнес-процессов именно в проекте TOVE.

Основными онтологиями, которые были разработаны в проекте TOVE, являются:

• онтология **Активностей и Состояний (Activities and States)**. Формализация понятия процесса и активности является критической для любых систем реинжиниринга бизнес-процессов, поскольку они суть базовые события, специфицирующие изменения в предметной области [Gruninger et al, 1994]. Состояния, в свою очередь, определяют, что должно быть истинным для того, чтобы активности могли выполняться, и что будет истинным после их завершения;

• онтология **Времени (Time)**. Активности «стартуют» в определенные моменты времени и, будучи запущенными, продолжаются в течение некоторого временного интервала. И более того, свойства состояний сохраняются во время этих

активностей. Для представления таких явлений в проекте TOVE было расширено множество темпоральных отношений [Allen, 1983];

• онтология **Ресурсов (Resources)**. Все активности требуют, чтобы некоторые объекты были доступны во время их выполнения. Это послужило основой для создания теории ресурсов, где в микротеории были аксиоматизированы понятия разделяемых vs. личных ресурсов, ресурсных ограничений и доступности ресурсов [Fadel, et al., 1994];

• онтология **Качества (Quality)**. Работы в данном направлении были связаны с созданием терминологии менеджмента качества, соответствующей стандартам ISO 9000, Baldridge Award и др. [Kim et al, 1999];

• онтология **Затрат (Cost)**. Целью данной разработки была формализация концептов, используемых в области управления затратами [Tham et al, 1994].

Подробнее о проекте TOVE см. [Гурьянова и др., 2011].

Проект SUPER — использование онтологий для интеграции моделей процессов

Основной целью проекта SUPER (*Semantics Utilized for Process Management within and between Enterprises* / Использование семантики для управления процессами внутри и между предприятиями) является перевод управления бизнес-процессами (BPM) с уровня информационных технологий на бизнес-уровень. Такая цель требует доступности системы управления процессами на уровне понятий бизнес-экспертов, способом представления которых выступают онтологии.

В рамках проекта SUPER (http://www.ip-super.org/) были разработаны онтологии, соответствующие стандартным нотациям бизнес-процессов EPC, BPMN, BPEL (см. на рисунке 6.15 пример для EPC), см. обобщенная онтология моделирования бизнес-процессов, верхнеуровневая онтология процессов, а также схемы соответствия между ними (см. рис. 6.16).

Данные онтологии и схемы, заложенные в программную среду, позволяют преобразовывать модели процессов из одной нотации в другую, а также осуществлять интегрированный поиск по всем моделям.

Методология ОРГ-Мастер

В настоящее время бизнес-организации вынуждены непрерывно развиваться и трансформироваться, чтобы реагировать на изменения рынка, среды, технологий или же для того, чтобы их инициировать. Для проведения целенаправленных, управляемых и успешных трансформаций принято выделять так называемую *архитектуру предприятия* (Enterprise Architecture, далее — EA), которая отражает организацию/устройство предприятия и предназначена для его анализа, проектирования изменений и преобразований [Зиндер, 2008; Op't Land et al, 2009; Тельнов, 2013; Кудрявцев и др., 2014]. Для моделирования и дальнейшей работы с EA используются специальные программные средства, которые в последнее время принято называть «инструменты управления архитектурой предприятия» (Enterprise Architecture Management/EAM tools, далее — EAM-инструменты) [Калянов, 2004; Matthes, 2008]. В этих инструментах активно применяется визуальное моделирование, т. е. используются «чертежи, позволяющие точно изображать хорошо структурированную информацию в понятной форме».

В России в течение последних 20 лет активно развиваются и используются на практике методология и комплекс программных средств ОРГ-Мастер [Григорьев и др., 2010; Григорьев, Кудрявцев, 2012; Григорьев и др. 2012; Григорьев и др., 2013; Kudryavtsev, Grigoriev, 2011; Grigoriev, Kudryavtsev, 2013]. ОРГ-Мастер

не является коробочным программным продуктом и представляет собой программно-методический комплекс, предназначенный для оказания сервисов в области EAM российским предприятиям, акцентируясь на бизнес-моделировании[1]. Преимуществами данного комплекса являются развитые онтологические возможности работы с репозиторием модели, позволяющие систематизировать и интегрировать множество частных моделей и точек зрения на предприятие в рамках создаваемой для него комплексной модели архитектуры. Таким образом становится возможным не только создавать красивые диаграммы для начальников, но оперативно разрабатывать сложные и детальные модели различных предприятий, отделяя уровень внутреннего (рабочего) представления от внешнего, презентационного.

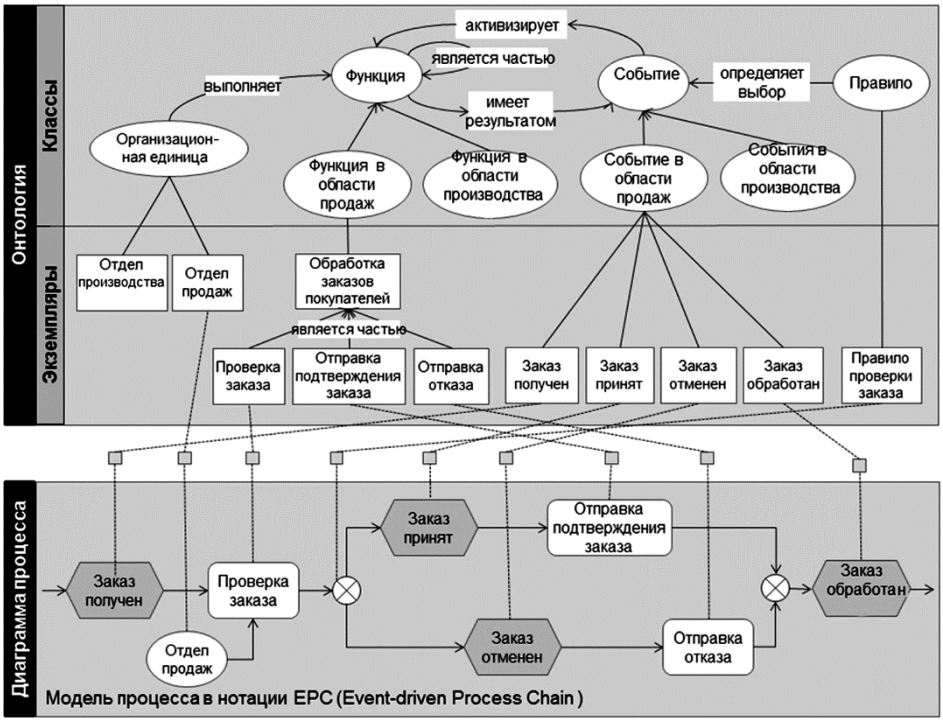


Рис. 6.15

Связь онтологии и языка моделирования бизнес-процессов EPC

Центральным элементом методологии в системе ОРГ-Мастер является онтология организации. Онтология в системе ОРГ-Мастер задает структуру модели организации, используется для настройки содержания визуальных нотаций и выходных документов, используется в коммуникациях сотрудников. Если в традиционных системах модели организации — это диаграммы, созданные с использованием различных графических языков организационного моделирования, то в системе ОРГ-Мастер организационная модель — это наполненная экземплярами онтология организации, а заменой графических языков выступают визуальные нотации, семантика которых

[1] Другим фокусом в области EAM является разработка интегральной IT-архитектуры предприятия — единого видения всех IT-систем, которые должны автоматизировать деятельность данного предприятия.

задается, опять же, онтологией организации. На рисунке 6.17 представлена упрощенная схема работы с программным инструментом ОРГ-Мастер.

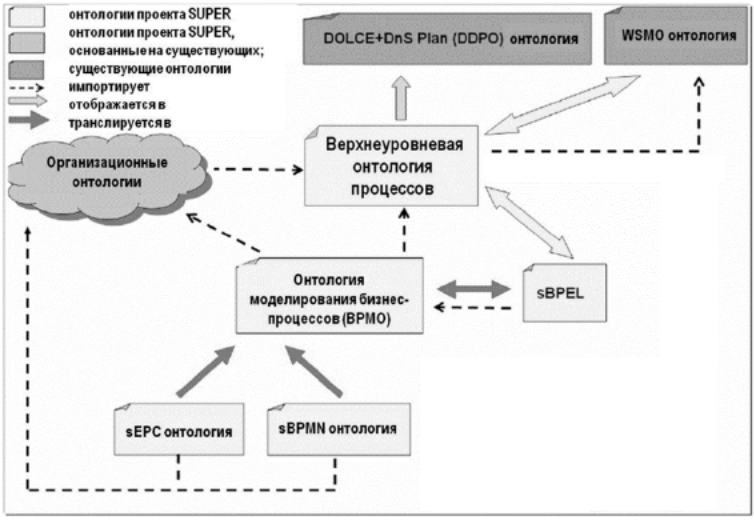


Рис. 6.16

Интеграция онтологий моделирования бизнес-процессов

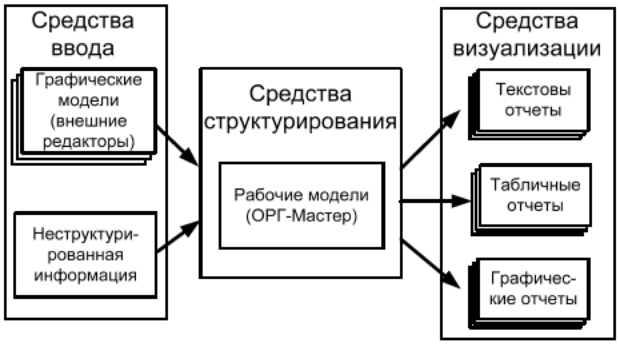


Рис. 6.17

Схема работы с системой ОРГ-Мастер

Более подробно компоненты методологии ОРГ-Мастер и их взаимосвязи представлены на рисунке 6.18. Задача организационного проектирования с точки зрения обработки информации состоит в разработке дескриптивных («как есть») и нормативных («как надо») моделей конкретной организации по правилам, заданным методиками и опирающейся на единую онтологию организации. При разработке могут использоваться референтные модели — результат обобщения конкретных моделей других организаций [Fettke, 2006; Тельнов, 2004; Кудрявцев и др., 2014]. Данную задачу, как правило, выполняют специалисты по анализу и моделированию организации (подробнее см. [Григорьев и др., 2012]).

Основными результатами моделирования являются автоматический выпуск целостной системы документов ориентированной на различные заинтересованные

стороны (руководство, менеджеры, сотрудники, сертифицирующие организации), а также создание файлов настройки информационных систем. Примерами документов являются положения о подразделениях, регламенты процессов, должностные инструкции, политики, документы системы менеджмента качества (СМК). Одним из видов документов являются диаграммы процессов в различных нотациях (IDEF0, EPC и т. п.), карты стратегий (причинно-следственные диаграммы), диаграмма взаимосвязанных цепочек создания ценности (карта бизнеса), органиграммы и т. п. Все они устанавливают определенные правила, по которым должны работать сотрудники, информационные системы и компания в целом. Полученная модель может также использоваться для ответов на ситуативные запросы при принятии организационных решений (Какие процессы влияют на достижение цели X? Какими показателями оценивается деятельность подразделения Y?) или для автоматизированной проверки организации на соответствие определенным регуляторным требованиям (ИСО, CMMI, SOX).



Рис. 6.18

Компоненты методологии и их взаимосвязь

Источником информации для разработки онтологической модели организации являются как неформализованные знания сотрудников, так и существующие документы. Также в системе ОРГ-Мастер широко используются референтные (эталонные) модели организации [Fettke, 2006; Тельнов, 2004; Кудрявцев и др., 2014], задающие нормативные структуры деятельности в различных областях, а также шаблоны документов/диаграмм, представляющие собой «образцовые» примеры заполненных документов/диаграмм. Это не является обязательным, но способствует повышению эффективности работ.

Структура модели, формы и содержание входных/выходных документов, а также методика выполнения работ по организационному проектированию фиксируются в соответствующей формализованной методологии, которая с точки зрения моделирования относится к метауровню («модель модели»). В частности, структура модели организации задается единой онтологией, а отдельные спецификации задают структуру и формат выходных документов, в т. ч. нотации диаграмм. Состав,

последовательность и входы/выходы работ по организационному проектированию формализуются в методике создания и использования модели. Один из ключевых этапов организационного проектирования методологии ОРГ-Мастер — функциональная декомпозиция организации (формирование архитектуры процессов) — рассмотрен в работе [Kudryavtsev, Grigoriev, 2011]. Вся методология фиксируется в программном средстве моделирования. Такая формализованная методология ведет начинающих аналитиков по шагам разработки модели, снижая квалификационные требования к специалистам. Формализованная методология может быть настроена под конкретную организацию с помощью специальной методики [Кудрявцев, 2009] и закреплена в виде корпоративных стандартов (в бизнесе) или нормативно-правовых актов (в органах государственной власти).

Онтология организации в методологии ОРГ-Мастер включает 12 групп понятий: деятельность (центральная группа), бизнесы и продукты, цели и показатели, нормы и правила, бизнес-концепции, внешняя среда и заинтересованные стороны, факторы влияния, информация, материальные ресурсы, финансовые ресурсы, персонал, организационная структура и инфраструктура. Рисунок 6.19 показывает верхнеуровневую систему понятий для группы «Деятельность».

Подробнее про онтологию и методологию ОРГ-Мастер см. [Кудрявцев и др., 2014].

Моделирование в системе ОРГ-Мастер основано на особом способе представления знаний, который формализован с помощью языка моделирования ОРГ-Мастер [Григорьев и др., 2013]. В основе языка лежат 2 формата ввода информации — классификаторы и проекции.

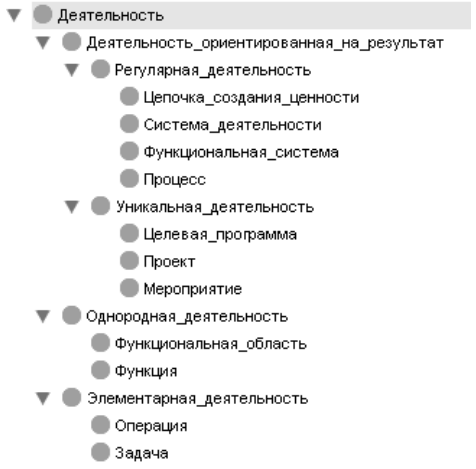


Рис. 6.19

Фрагмент онтологии организации ОРГ-Мастер

Классификатор — формат ввода нижнеуровневых классов и экземпляров онтологии, их свойств и иерархических связей между ними (рис. 6.20). Основные типы связей: «класс-подкласс», «часть-целое», «подчиняется» и т. п.

Проекция — формат ввода связей между экземплярами онтологии, перечисленными в классификаторах. Примеры типов связей: «выполняет», «обеспечивает достижение», «отвечает за». Связи в проекции могут быть направленными и ненаправленными, типизированными и нетипизированными. Проекции могут использоваться в режимах списков (рис. 6.21) или матрицы (рис. 6.22).

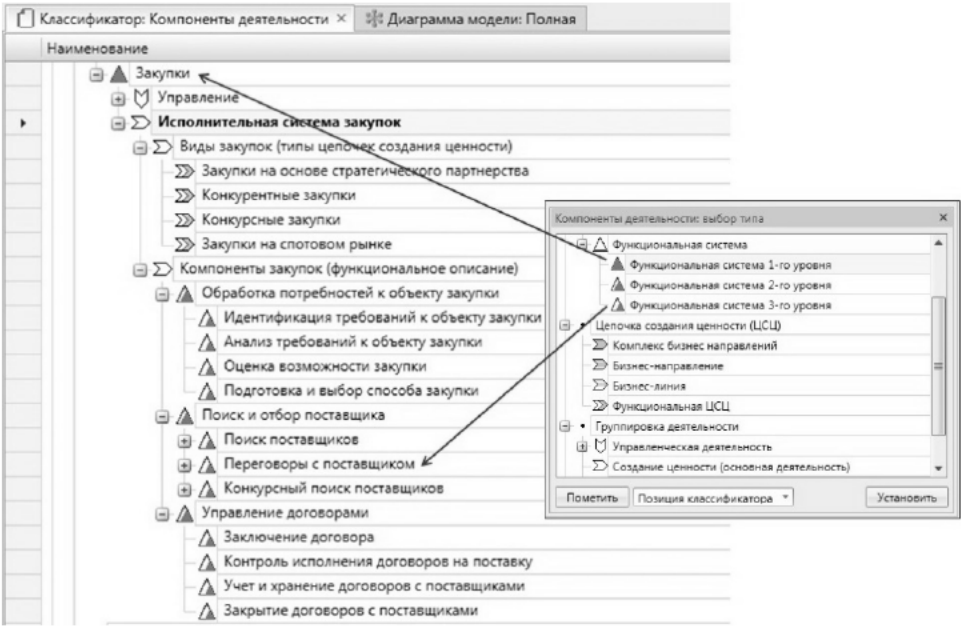


Рис. 6.20

Пример классификатора «Компоненты деятельности»

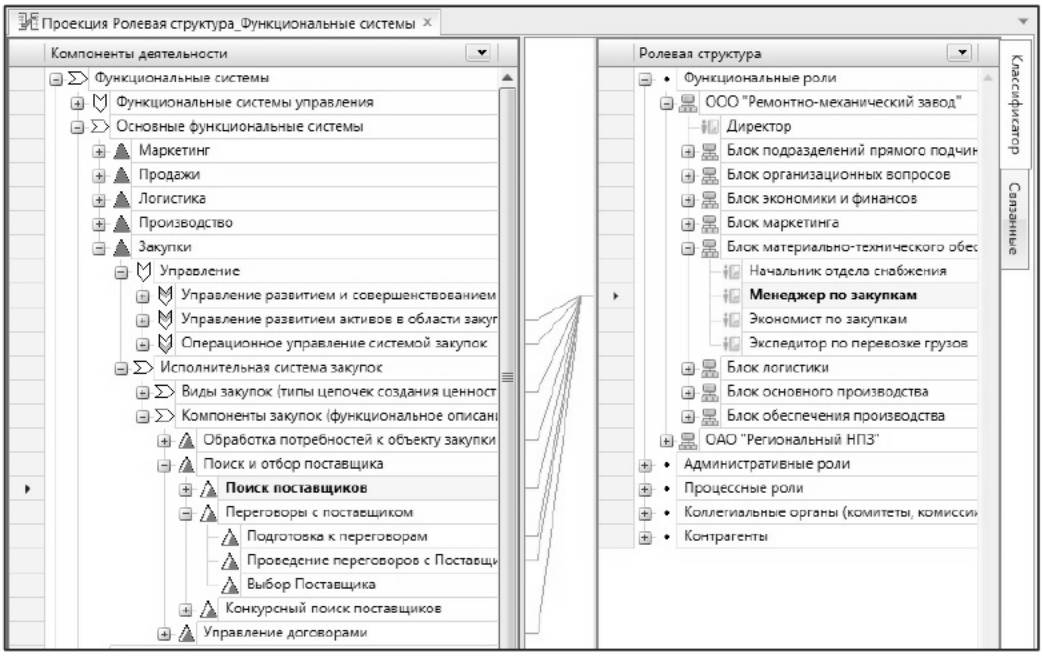


Рис. 6.21

Пример проекции, связывающей деятельность с ролевой структурой (режим списков)

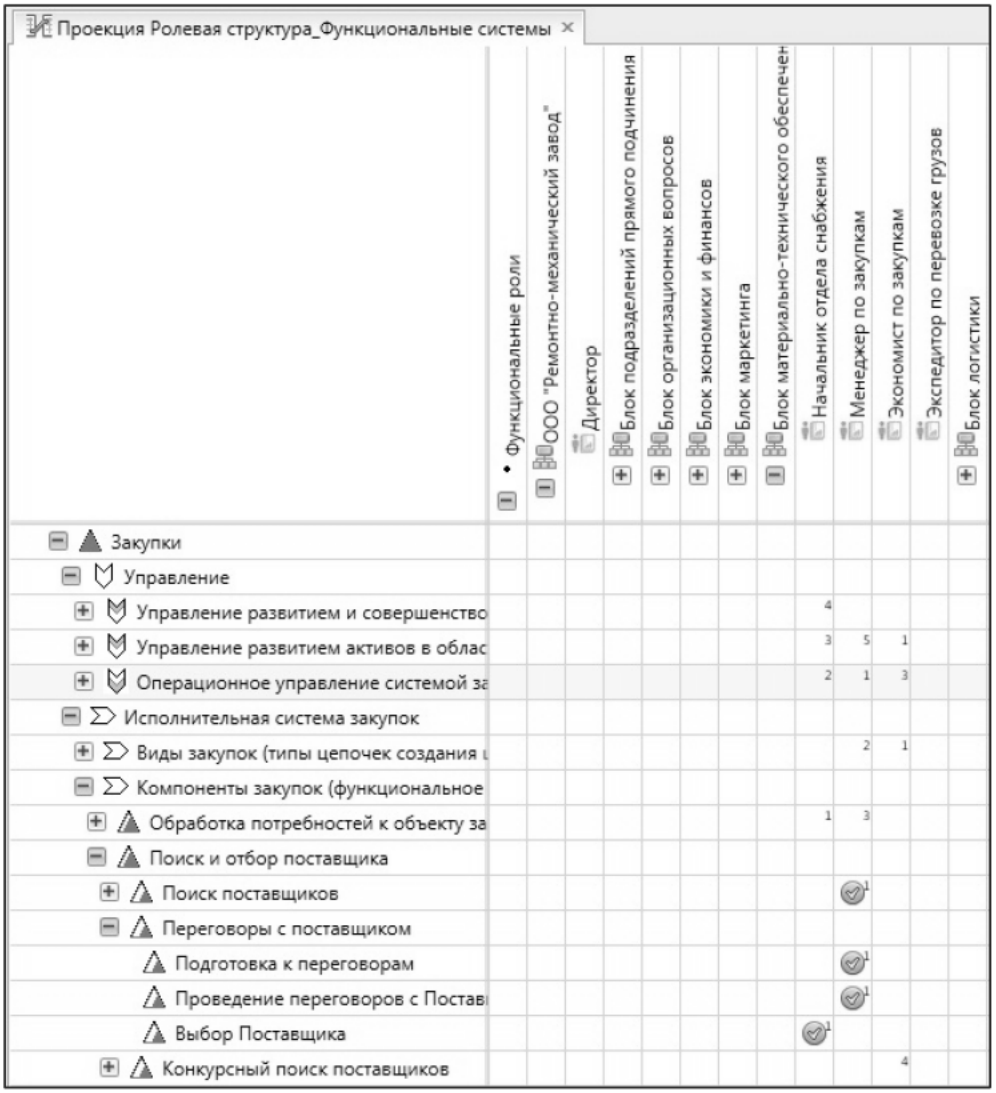


Рис. 6.22

Пример проекции, связывающей деятельность с ролевой структурой (режим матрицы)

Для строгой типизации элементов модели используется справочник типов (см. рис. 6.23), который фиксирует таксономию верхнеуровневых классов, типы связей между классами, значения свойств классов. Каждому типу ставится в соответствие пиктограмма (графический знак).

Для настройки и представления структуры модели в системе ОРГ-Мастер используется диаграмма модели (см. рис. 6.24), на которой представлены классификаторы и связывающие их проекции.

ОРГ-Мастер не использует диаграммы в качестве базовых средств моделирования. В качестве последних используется таблично-диалоговое моделирование,

и, таким образом, визуальные средства должны быть подходящим образом интегрированы с ними. При этом они оказываются существенно различными — по назначению, по функционалу, а также по способам интеграции с пакетом ОРГ-Мастер. Вот список этих средств:

- внутренние редакторы — являются частью поставки базовых средств моделирования базового EAM-пакета (в нашем случае — ОРГ-Мастера), используются для выборочного создания фрагментов моделей (установление связей, просмотр модели и ее фрагментов и т. д.);
- внешние редакторы — предназначаются для клиентов, обеспечивают удобство ввода информации в ОРГ-Мастер, могут дорабатываться для нужд конкретного EA-проекта, а также создаваться новые;
- графические отчеты — конечные диаграммные отчеты для заказчика (для вставки в отчетные документы или выгрузки на Web-портал);
- импорт из других визуальных средств (Visio, CMap и пр.).

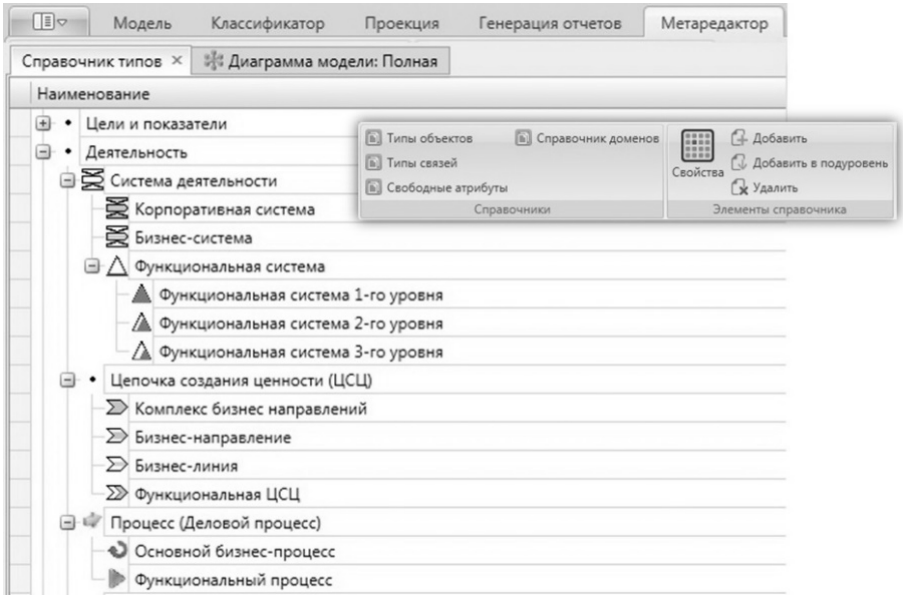


Рис. 6.23

Справочник типов в метаредакторе системы ОРГ-Мастер

Представленная методология и инструмент моделирования ОРГ-Мастер имеют 15-летнюю историю применения в проектах по организационному развитию и совершенствованию процессов коммерческих организаций и органов государственной власти. ОРГ-Мастер используется в коммерческих организациях России и стран СНГ, при этом размер клиентов-пользователей ОРГ-Мастера варьируется от небольших компаний до холдинговых структур с количеством работников до 10 000 человек.

Технология ОРГ-Мастер была использована при выполнении более 50 проектов по управленческому консультированию. Примеры организаций: Киришская ГРЭС, ИЛИМ, Северо-Западное пароходство, Группа предприятий ГОТЭК, Концерн «Силовые машины», ЗАО «Евросиб», ОАО «Иркутскэнерго», ОАО «Газаппарат», «АСТРА Холдинг», Торговый дом «Петровский», Холдинг ПЕКАР и др.

Представленные подходы легли в основу Концепции моделирования деятельности органов государственной власти в ходе административной реформы РФ. Ключевые направления использования моделирования в рамках реализации Концепции административной реформы: регламентация деятельности органов власти, оптимизация структуры и функций органов власти, внедрение механизмов управления по результатам. С использованием программного продукта было разработано более 10 административных регламентов для Федеральной миграционной службы, Федеральной регистрационной службы, Федерального агентства по физической культуре и спорту, Федеральной службы по труду и занятости и других органов исполнительной власти федерального и регионального уровней.

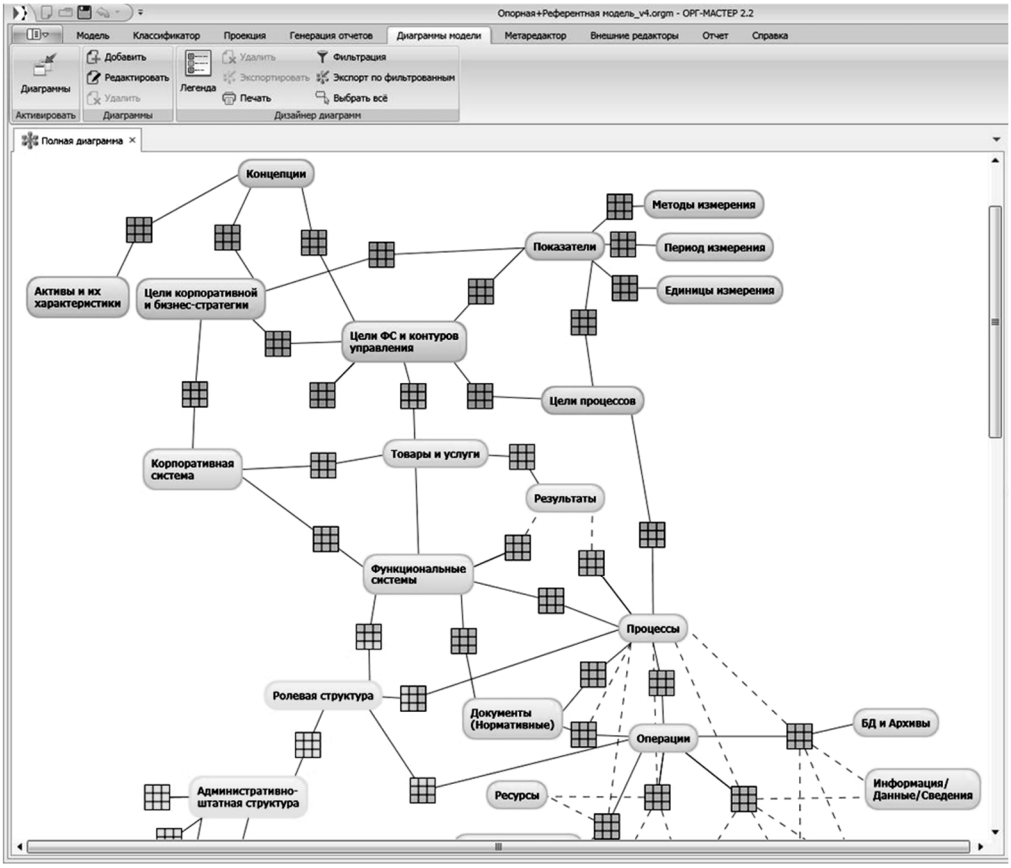


Рис. 6.24

Диаграмма модели в системе ОРГ-Мастер

Essential Project

В рамках проекта Essential [Mayall A., Carter, 2015] предложены метамодель (онтология) предприятия, а также программное средство для моделирования — Essential Architecture Manager.

Essential Architecture Manager (http://www.enterprise-architecture.org/) является программным средством, предназначенным для поддержки практик управления архитектуры предприятия. Особое внимание в продукте уделено:

- легкости поддержки;
- легкости обучения;
- возможности быстро приносить результат.

Essential Architecture Manager реализует следующий функционал:

- наличие репозитория;
- метамодель, поддерживающая различные уровни архитектуры и точки зрения;
- поддерживает отношения между объектами в репозитории с учетом точек зрения;
- поддерживает возможность импорта моделей и артефактов;
- возможность расширения репозитория.

По-крупному, вся функциональность может быть сгруппирована в две категории:

- функции поддержки моделирования предприятия;
- функции поддержки пользователей в рассмотрении этой модели с различных точек зрения, за счет отчетов и анализа.

Такая группировка обуславливает архитектуру Essential Architecture Manager, состоящего из (рис. 6.25):

- Essential Modeller — создание и поддержка архитектурной модели;
- Essential Viewer — генерация отчетов, позволяющих рассматривать модель с различных точек зрения.

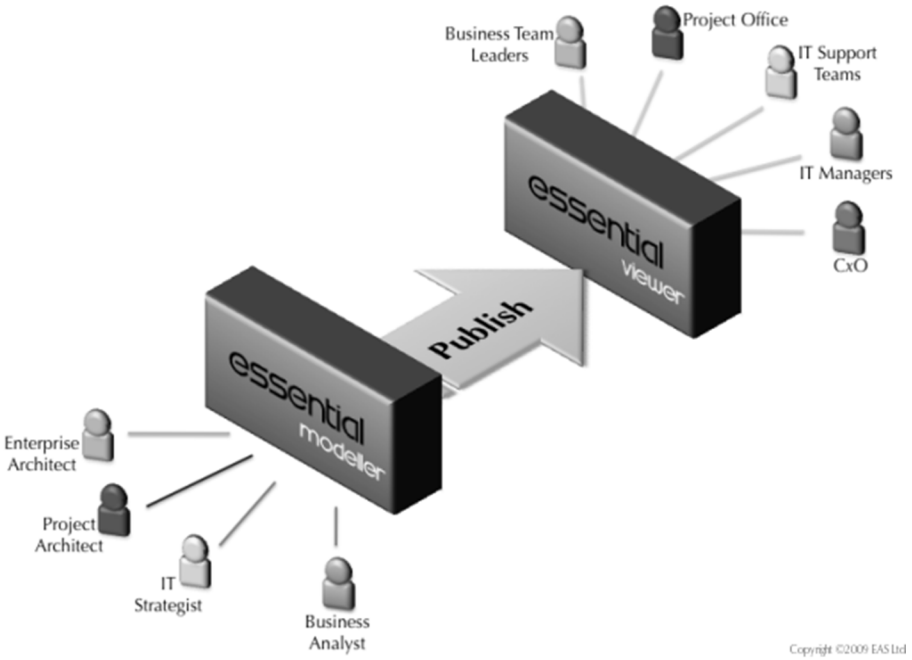


Рис. 6.25

Контекст средства «Essential Architecture Manager»

Компоненты Essential Architecture Manager разработаны на базе открытых (*open source*) технологий (рис. 6.26).

Essential Modeller имеет в своей основе редактор онтологий (Protege Ontology Editor), а также базу знаний (Knowledge Repository). Использование редактора

онтологий для определения метамодели позволяет получить управляемую, гибкую и разделяемую модель архитектуры. На рисунке 6.27 представлены некоторые примеры.

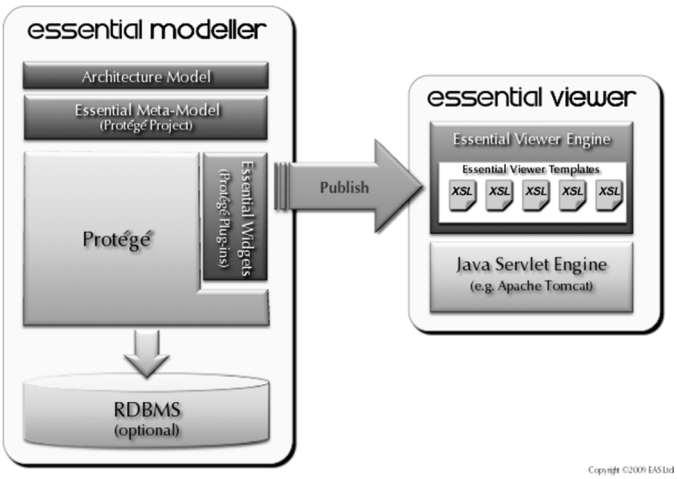


Рис. 6.26

Высокоуровневая архитектура Essential Architecture Manager

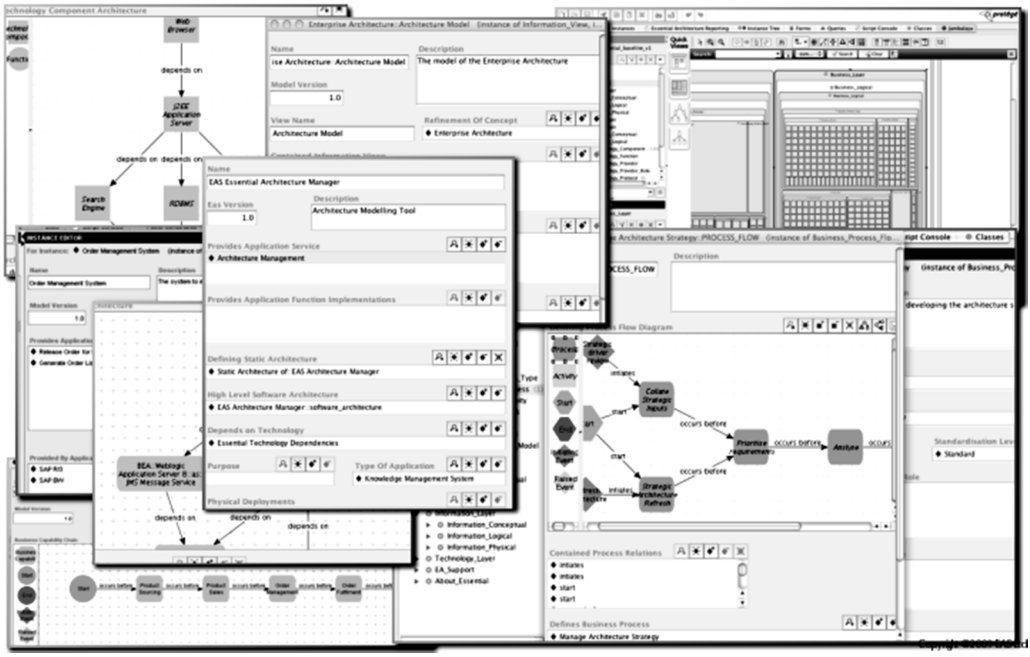


Рис. 6.27

Примеры различных представлений Essential Modeller

Essential Viewer является Java-приложением, независимым от платформы. EV предоставляет гибкий и расширяемый способ генерации различных точек зрения на модель. Существует 100 преднастроенных представлений. Предоставля-

ются расширенные возможности проектирования и генерации отчетности различного уровня сложности. На рисунке 6.28 представлены некоторые примеры.

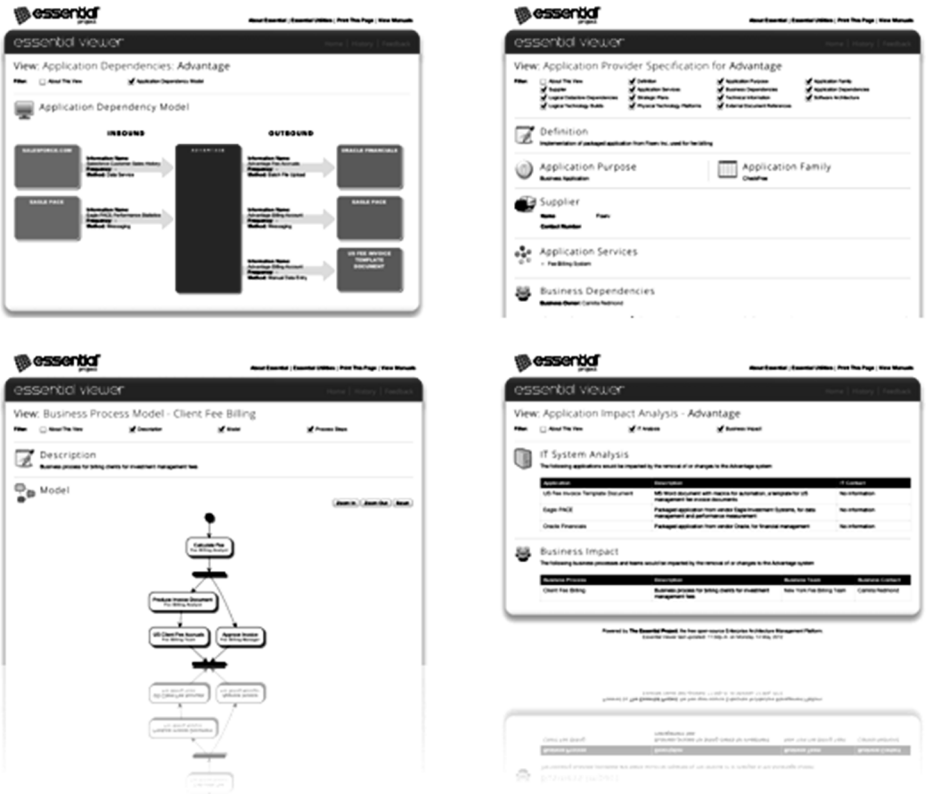


Рис. 6.28

Примеры различных представлений Essential Viewer

6.3. ПРИМЕНЕНИЕ ОНТОЛОГИЙ ДЛЯ МОДЕЛИРОВАНИЯ В ГОСУДАРСТВЕННОМ И ГОРОДСКОМ УПРАВЛЕНИИ

Моделирование органов государственной власти для задач административной реформы

Электронное моделирование органов государственной власти и местного самоуправления (ОГВМС) может быть использовано для совершенствования государственного управления и, в частности, применялось в административной реформе в Российской Федерации[1] для решения следующих задач [АНО КМЦ Бизнес-инжиниринг, 2006; Кудрявцев и др., 2009]:

- внедрение в органах исполнительной власти принципов и процедур управления по результатам;
- разработка и внедрение стандартов государственных услуг, предоставляемых органами исполнительной власти, а также административных регламентов в органах исполнительной власти;

[1] Указ Президента Российской Федерации от 23 июля 2003 г. № 824 «О мерах по проведению административной реформы в 2003 – 2004 годах».

- оптимизация функционирования органов исполнительной власти;
- повышение эффективности взаимодействия органов исполнительной власти и гражданского общества, а также повышение прозрачности деятельности органов исполнительной власти;
- модернизация системы информационного обеспечения органов исполнительной власти;
- формирование необходимого организационного, информационного, ресурсного и кадрового обеспечения административной реформы, совершенствование механизмов распространения успешного опыта государственного управления.

Для большинства из перечисленных задач применение электронного моделирования позволяет получить достаточно точную, наглядную и непротиворечивую информацию для принятия управленческих решений ценой меньших временных и финансовых затрат в сравнении с «ручным» способом. Моделирование деятельности является средством описания организационного устройства и порядка функционирования ОГВМС, предназначенным для анализа и оптимизации структур и процессов функционирования. Применение электронного моделирования позволяет быстро и экономично определить наилучший вариант преобразований структуры и функций ОГВМС в соответствии с изменением социально-экономической ситуации, появлением новой задачи или решения вышестоящих органов о перераспределении сфер ведения (управленческой ответственности).

Для методической поддержки административного моделирования в рамках НИОКР выполненной компанией АНО КМЦ «Бизнес-инжиниринг» по заказу Министерства экономического развития и торговли были разработаны[1] следующие компоненты (табл. 6.2):

- проект стандарта общей административной модели, который включает административную онтологию, являющуюся семантической основой административного моделирования, и формат представления данных общей организационной модели;
- проект стандарта представления частных административных моделей (методов описания, ракурсов), который содержит спецификации содержания и формы представления частных административных моделей (методов описания, ракурсов);
- методика по административному моделированию, описывающая основные типы административных моделей (организационно-функциональная модель, модель административных процессов, модель результативности), а также предлагаемый подход к моделированию;
- руководство по жизненному циклу моделирования, которое описывает роль программных средств в реализации предлагаемого подхода, а также уточняет практические аспекты реализации жизненного цикла моделирования;
- методики по созданию и использованию основных типов административных моделей (организационно-функциональная модель, модель административных процессов, модель результативности);
- справочники, представляющие собой существующие способы классификации и образцы заполнения отдельных частей административной модели. Использование справочников повышает качество создаваемых моделей, повышает скорость создания моделей, облегчает интеграцию моделей, позволяет передавать опыт и лучшую практику государственного и муниципального управления;
- опорные административные модели, определяющие структуру административных моделей для решения различных задач и уровней государственного управления;

[1] Данные материалы представлены здесь: http://bigc.ru/government/modeling/

Таблица 6.2

Компоненты проблемно-ориентированных решений для органов государственной власти

	Решение 1: Оптимизация структуры и функций органов власти	**Решение 2: Разработка административных регламентов**	**Решение 3: Управление результативностью**
Общие компоненты	**Инструменты / Программные продукты** ГОС-Мастер **Методические материалы** • Проект стандарта представления полной организационной модели, *который включает административную онтологию, являющуюся семантической основой административного моделирования, и формат представления данных общей организационной модели.* • Проект стандарта частных организационных моделей (человекочитаемых и машиночитаемых) для различных ролей с методиками их использования, *который содержит спецификации содержания и формы представления частных административных моделей.* • Методические рекомендации по административному моделированию, *описывающие основные типы административных моделей (модель результативности, организационно-функциональная модель, модель административных процессов), а также предлагаемый подход к моделированию.* • Руководство по жизненному циклу моделирования, *которое описывает роль программных средств в реализации предлагаемого подхода, а также уточняет практические аспекты реализации жизненного цикла моделирования.* **Справочники** Справочные классификаторы **Обучение** Тема 1. Система современных взглядов и идей организационного менеджмента Тема 2. Моделирование деятельности органов власти в контексте административной реформы		
Частные компоненты	**Методические материалы** Методика построения организационно-функциональной модели **Референтные и опорные модели** • Сводная организационно-функциональная модель (ОФМ) для органов исполнительной власти (ОИВ) государства в целом • Опорная ОФМ для Федерального ОИВ • Опорная ОФМ для Субъекта РФ • Референтная ОФМ для Федерального ОИВ (Минэкономразвития) • Референтная ОФМ для Субъекта РФ (Ямало-Ненецкий АО) **Обучение** Тема 3. Построение организационно-функциональной модели органа исполнительной власти	**Инструменты / Программные продукты** ГОС-Мастер ГРАФИКС Мастер административных регламентов **Методические материалы** • Методика создания моделей административных процессов в ГОС-Мастер ГРАФИКС • Методика построения модели административного процесса с помощью Мастера административных регламентов **Референтные и опорные модели** • Опорная процессная модель для Федерального ОИВ • Опорная процессная модель для Субъекта РФ • Референтная процессная модель для Федерального ОИВ (Минэкономразвития, Минсвязь) • Референтная процессная модель для Субъекта РФ (Ямало-Ненецкий АО) **Обучение** Тема 4. Построение модели процессов органа исполнительной власти	**Методические материалы** Методика построения модели результативности **Референтные и опорные модели** • Сводная модель результативности (для государства в целом) • Опорная модель результативности для Федерального ОИВ • Опорная модель результативности для Субъекта РФ • Референтная модель результативности для Федерального ОИВ (Минэкономразвития) • Референтная модель результативности для Субъекта РФ (Ямало-Ненецкий АО) **Обучение** Тема 5. Построение модели результативности деятельности

• референтные административные модели, являющиеся примерами наполнения административных моделей для решения различных задач и уровней управления. Референтные модели были получены в результате выполнения пилотных проектов с Минэкономразвития (уровень ФОИВ) и с Ямало-Ненецким АО (уровень Субъектов РФ).

Для программной поддержки административного моделирования было *разработано* программное средство ГОС-Мастер, поддерживающее предложенную административную онтологию и стандарты моделирования, которое предназначено для специалистов органов государственной власти Российской Федерации, занимающихся вопросами проведения административной реформы. ГОС-Мастер начал использоваться для моделирования органов государственной власти с 2005 г. и первоначально (версия 1.0) был ориентирован на поддержку формирования стандартов представления административно управленческих процессов.

Для обучения административному моделированию разработан учебный курс для государственных служащих «Моделирование деятельности органов исполнительной власти как инструмент административной реформы».

Для нормативно-правового обеспечения внедрения административного моделирования разработан комплект проектов нормативно-правовых актов и распорядительной документации.

Обзор административной онтологии

Административная онтология, разработанная в рамках НИОКР, представлена в проекте стандарта представления полной организационной модели[1]. Состав понятий верхнего уровня административной онтологии, сгруппированных по предметным областям, представлен в таблице 6.3.

Таблица 6.3

Понятия верхнего уровня административной онтологии

Предметные области	Понятия
1. Внешнее окружение	1.1. Домашние хозяйства. 1.2. Финансовые корпорации. 1.3. Нефинансовые корпорации. 1.4. Некоммерческие организации. 1.5. Государственный сектор.
2. Цели и показатели	2.1. Цели и задачи. 2.2. Показатели. 2.3. Результаты.
3. Деятельность	3.1. Сферы деятельности. 3.2. Государственные услуги. 3.3. Государственные функции. 3.4. Административные процессы. 3.5. Программная деятельность. 3.6. Приоритетные национальные проекты. 3.7. Административные действия. 3.8. Поручения. 3.9. События и состояния (Условия ветвления в процедурах). 3.10. Мероприятия.

[1] http://bigc.ru/government/modeling/standards/big/dsppom/

Продолжение табл. 6.3

Предметные области	Понятия
4. Организационная структура	4.1. ОГВМС (Органы государственной власти или местного самоуправления). 4.2. Организационные подразделения и должности. 4.3. Организационные роли.
5. Личный состав	5.1. Федеральные государственные служащие. 5.2. Государственные гражданские служащие субъекта Российской Федерации. 5.3. Муниципальные служащие. 5.4. Технические работники органов местного самоуправления.
6. Основные фонды	6.1. Материальные основные фонды (основные средства). 6.2. Нематериальные основные фонды (нематериальные активы).
7. Информация	7.1. Документы. 7.2. Информационное поле (в т. ч. Учетные данные). 7.3. Хранилища информации (в т. ч. Учетные регистры).
8. Бюджет	8.1. Доходы. 8.2. Расходы.

Данная онтология описывает состояние организационного управления органов государственной власти в виде «как есть» в соответствии с нормативно-правовыми актами, которые используются в деятельности органов государственной власти. Для всех верхнеуровневых понятий административной онтологии даны определения, источниками которых являются нормативно-правовые акты (в случае отсутствия таковых — экспертное мнение).

Между понятиями административной онтологии устанавливаются как иерархические родовидовые связи, так и перекрестные связи различных типов (рис. 6.29).

Пример иерархических родовидовых связей для целей и задач органов государственной власти:

Цели и задачи.
1. Цели и задачи федерального уровня.
1.1. Цели Правительства Российской Федерации.
1.2. Стратегические цели субъекта бюджетного планирования (СБП).
1.3. Тактические задачи СБП.
1.4. Межведомственные задачи.
1.5. Цели программ.
1.5.1. Цели Федеральных целевых программ (ФЦП).
1.5.2. Цели Ведомственных целевых программ (ВЦП).
1.6. Задачи программ.
1.6.1. Задачи ФЦП.
1.6.2. Задачи ВЦП.
2. Цели и задачи регионального уровня.
2.1. Цели субъекта Российской Федерации.
2.2. Цели СБП субъекта Российской Федерации.
2.3. Тактические задачи СБП субъекта Российской Федерации.
2.4. Цели окружных и ведомственных программ субъекта Российской Федерации.

2.5. Задачи окружных и ведомственных программ субъекта Российской Федерации.

3. Межуровневые задачи для органов исполнительной власти субъектов Российской Федерации.

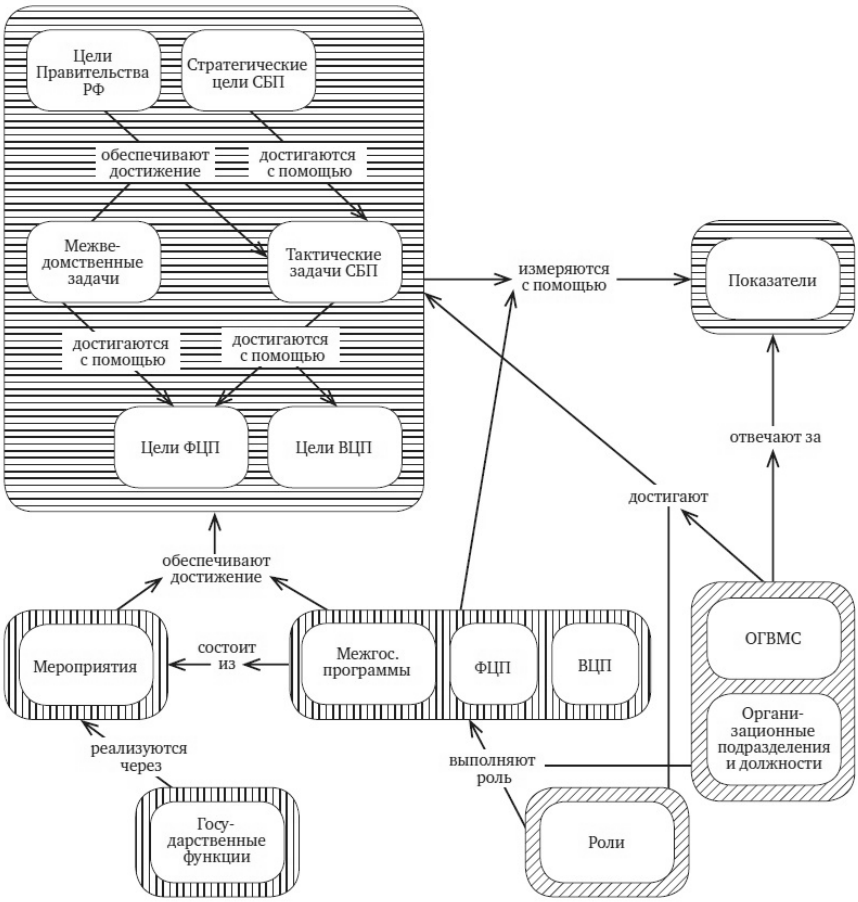


Рис. 6.29

Фрагмент взаимосвязи понятий административной онтологии в области управления результативностью

Пример иерархических родовидовых связей для государственных функций органов государственной власти:

Государственные функции.

1. Функции по принятию нормативных правовых актов.
2. Функции по контролю и надзору.
3. Функции по управлению государственным имуществом.
4. Функции по оказанию государственных услуг.

Пример иерархических родовидовых связей для программной деятельности органов государственной власти:

Программная деятельность.

1. Бюджетные целевые программы.
   1.1. Межгосударственные целевые программы.
   1.2. Бюджетные целевые программы, финансируемые из федерального бюджета.
      1.2.1. Федеральные целевые программы (ФЦП).
      1.2.2. Подпрограммы ФЦП.
      1.2.3. Ведомственные целевые программы федеральных органов исполнительной власти (ВЦП ФОИВ).
         1.2.3.1. Утверждаемая ведомственная целевая программа.
         1.2.3.2. Ведомственная целевая программа, выделенная в аналитических целях.
      1.2.4. Федеральная адресная инвестиционная программа (ФАИП).
   1.3. Бюджетные целевые программы субъекта Российской Федерации.
      1.3.1. Окружные и областные целевые программы.
      1.3.2. Подпрограммы окружной или областной целевой программы.
      1.3.3. Ведомственные целевые программы субъектов Российской Федерации.
      1.3.4. Ведомственные целевые программы, выделяемые в аналитических целях.

Состав документов и диаграмм, получаемых из административной модели

В результате анализа задач административного моделирования и информационных потребностей основных категорий получателей информации — экспертов, граждан, исполнителей и руководителей в рамках НИОКР был определен состав (табл. 6.4), структура содержания и форма представления выходных частных административных моделей (методов описания, ракурсов) генерируемых из административной модели.

Таблица 6.4

Состав частных административных моделей органов государственной власти

№ п.п.	Название частной административной модели (документа или диаграммы)	Тип
1	*Организационно-функциональная модель (ОФМ)*	
1.1.	Вопросы ОГВМС	Таблица
1.2.	Перечень государственных услуг, оказываемых ОГВМС	Текст
1.3.	Перечень государственных услуг, с указанием исполняющих органов	Текст
1.4.	Административно-штатная структура ОГВМС	Текст
1.5.	Положение об организационно-функциональной структуре ОГВМС	Текст
1.6.	Положение о функциях подразделения	Текст
1.7.	Положение о функциональных обязанностях сотрудника	Текст
1.8.	«Матрица ответственности (на уровне ОГВМС)» — по административным процессам	Таблица
1.9.	«Матрица ответственности ОГВМС» — по функциональным областям	Таблица
1.10.	Положение о функциональных областях ОГВМС	Таблица, Графика
1.11.	Положение о функциях подразделения — расширенное (Т) / Классификация функций подразделения (Тб)	Текст

Продолжение табл. 6.4

№ п.п.	Название частной административной модели (документа или диаграммы)	Тип
2	*Модель административных процессов*	
2.1.	Административный регламент	Текст
2.2.	Перечень внутренних процессов, поддерживаемых ОГВМС, и их закрепление за организационными звеньями	Текст
2.3.	Требования к документам, для предоставления государственной услуги	Текст
2.4.	Паспорт административной процедуры	Таблица
2.5.	Функционально-табличная форма представления административного процесса	Таблица
2.6.	Неструктурированное описание процесса оказания услуги	Графика
2.7.	Структурированное описание процесса оказания услуги	Графика
2.8.	Описание административного процесса по исполнителям	Графика
2.9.	Описание административного процесса в формате IDEF0	Графика
2.10.	Описание административного процесса в формате DFD	Графика
3	*Модель результативности*	
3.1.	Элементы докладов о результатах и основных направлениях деятельности для федерального и регионального уровней, в частности: • система целей, задач и результатов деятельности; • состав и структура показателей; состав и описание целевых программ; • взаимосвязи между этими сущностями. Связи с организационно-функциональной и процессной моделями Аналитические отчеты на основе выставленных взаимосвязей	Тексты, Таблицы

Конечные результаты использования административного моделирования

1. Непротиворечивость нормативно-регламентирующей документации организации.

2. Снижение трудозатрат на разработку нормативно-регламентирующей документации, аналитических и информационных материалов об организации.

3. Повышение скорости реализации организационных изменений при изменениях внешней или внутренней среды (среднее время отклонения организационных проектов от требуемого/планового времени проведения изменений).

4. Повышение управляемости процессов (минимизация вариабельности процессов).

5. Накопление и передача знаний.

6. Снижение уровня рисков, связанных с уходом сотрудников.

7. Снижение времени адаптации новых сотрудников.

Моделирование комплексов городского хозяйства для системного развития средств ИКТ города

Рассмотрим применение онтологического моделирования для задач городского управления на примере одного из проектов, реализованных в Москве [Костырко и др., 2012], результаты которого могут быть адаптированы к другим городам.

Постановлением правительства Москвы № 349-ПП от 9 августа 2011 г. была утверждена государственная программа города Москвы «Информационный город (2012–2016 годы)» (далее Программа). В качестве целей Программы обозначены:

- повышение качества жизни населения г. Москвы за счет широкомасштабного использования информационно-коммуникационных технологий в социальной сфере, в сфере обеспечения комплексной безопасности г. Москвы, в сфере управления городским хозяйством, в повседневной жизни граждан;
- повышение эффективности и прозрачности городского управления.

Для обеспечения результативности и эффективности реализации мероприятий Программы, а также обеспечения эффективности и соответствия целям реализации Программы процессов разработки и эксплуатации информационных систем города Москвы был создан «Системный проект». «Системный проект» (СП) представляет собой совокупность взаимосвязанных моделей, методик, рекомендаций, требований и других компонентов, предназначенных для интеграции процессов управления отраслями городского хозяйства с действующими и планируемыми информационными системами и ресурсами, ИКТ-компонентами, соответствующими видами программного и технического обеспечения.

СП является примером архитектурного подхода к управлению ИКТ и реинжинирингу организаций. Центральным элементом такого подхода является понятие «архитектура организации» (enterprise architecture) («архитектура электронного государства» в госуправлении) [Данилин, Слюсаренко, 2005; Christiansen, Gotze, 2006; Guijarro, 2007]. Наиболее известным и зрелым примером архитектурного подхода в управлении ИКТ органов власти является программа архитектуры федерального предприятия США (FEA — Federal enterprise architecture, http://www.whitehouse.gov/omb/e-gov/fea), а известным бизнес-примером является TM Forum Frameworx (https://www.tmforum.org/tm-forum-frameworx/).

Конечной целью разработки и реализации СП является повышение результативности и эффективности управления городским хозяйством за счет использования информационно-коммуникационных технологий. Для этого необходима разработка и внедрение новой модели управления городским хозяйством, представляющей собой взаимоувязанные процессы управления отдельными комплексами городского хозяйства, а также интеграция указанной модели с единой архитектурой информационных систем и ресурсов города Москвы.

Примером, объясняющим потребность в СП, является следующая распространенная ситуация: менеджер департамента информационных технологий (ДИТ) города, ответственный за информатизацию определенной отрасли (например, образования или здравоохранения) получает от представителя отраслевого департамента (например, департамента образования или здравоохранения) предложение о необходимости разработки новой информационной системы (ИС). Менеджер ДИТа, в свою очередь, должен принять решение о разработке и внедрении новой системы (инвестиционное решение). Обычно это происходит при формировании бюджета города на следующий год. Однако у каждого отраслевого департамента уже, как правило, есть несколько десятков (а то и сотен) информационных систем. На сопровождение каждой из них выделяются деньги из бюджета. В связи с этим у менеджера ДИТа возникают закономерные вопросы:

- Как соотносится предлагаемая ИС с существующими?
- Не будет ли предлагаемая система дублировать функциональность какой-либо из существующих систем?

- Требует ли деятельность (госфункции, процессы), которую предлагается автоматизировать, дополнительной ИТ-поддержки и в каком объеме?
- Не будет ли предлагаемая ИС собирать информацию, которая уже есть в распоряжении города и может быть получена через межведомственное взаимодействие?

Компоненты СП предназначены для ответа на эти вопросы.

Методологически работы по моделированию в рамках СП основаны на результатах НИР «Научно-методическое обеспечение эффективной интеграции процессов управления комплексами городского хозяйства с использованием средств информационно-коммуникационных технологий» [ИГМУ НИУ ВШЭ, 2011]. В данной работе проведена детализация и адаптация НИР «Моделирование деятельности органов власти, государственных и муниципальных организаций» [АНО КМЦ «Бизнес-инжиниринг», 2006; Кудрявцев и др., 2009] и НИР «Эталонные модели организации деятельности в государственном секторе» [ИПГМУ ВШЭ, 2006].

Основным инструментом СП является набор взаимосвязанных моделей комплексов городского хозяйства, включая их ИТ-архитектуру. Именно использование таких моделей позволит достичь целей «Системного проекта» и ответить на вопросы сотрудников ДИТа, представленные выше.

СП включает следующие модели:

1. Процессную модель, описывающую систему процессов, выполняемых как внутри отдельных комплексов городского хозяйства, так и пронизывающих сразу несколько отраслей. Кроме процессов, данная модель содержит декомпозицию сфер ведения, а также описание государственных функций (в том числе и функций по оказанию государственных услуг).

2. Оргструктурную модель, которая содержит справочник органов исполнительной власти (ОИВ), подведомственных организаций, а также структуру подчиненности.

3. Модель результативности, содержащую показатели для оценки состояния комплексов городского хозяйства, а также показатели результативности программы «Информационный город». Кроме показателей, в данную модель входит описание ценностей, создаваемых процессами (конечные эффекты), и непосредственных результатов выполнения процессов.

4. Информационную модель, описывающую информационные сущности, которые используются в комплексах городского хозяйства, атрибуты и связи этих сущностей, а также связь информационных сущностей и их атрибутов с процессами и ИС. Кроме того, в этой модели перечисляются информационные ресурсы (реестры, регистры), описывается их содержание и применение.

5. Модель информационных систем (ИС), содержащую перечень разрабатываемых и используемых ИС города, паспорт каждой ИС, описание и структуризацию функциональных возможностей ИС города.

Традиционно для создания подобных моделей используются визуальные методы моделирования (ARIS, IDEF, Archimate). Однако масштаб создаваемых моделей, а также многообразие требуемых нотаций и точек зрения на организацию (viewpoints) приводят к сложности создания и сопровождения интегрированных и целостных моделей. Для решения данной проблемы представляется целесообразным использовать подход к моделированию, опирающийся на разделение форматов «внутреннего» и «внешнего» представления организационных знаний [Григорьев, Кудрявцев, 2012; Kudryavtsev, Grigoriev, 2011].

«Внутренний» формат используется для централизованного хранения знаний в репозитории, а также для работы специалистов по моделированию. «Внешний»

формат используется для передачи знаний конечным пользователям в виде текстов, таблиц и диаграмм. В соответствии с данным подходом «внутренняя» структура модели задается легковесной онтологией СП (рис. 6.30) и ее содержимое (экземпляры классов и связи между ними) редактируется и хранится в специальных форматах представления информации — классификаторах и проекциях (см. подробнее [Григорьев и др., 2012; Григорьев и др., 2013]).

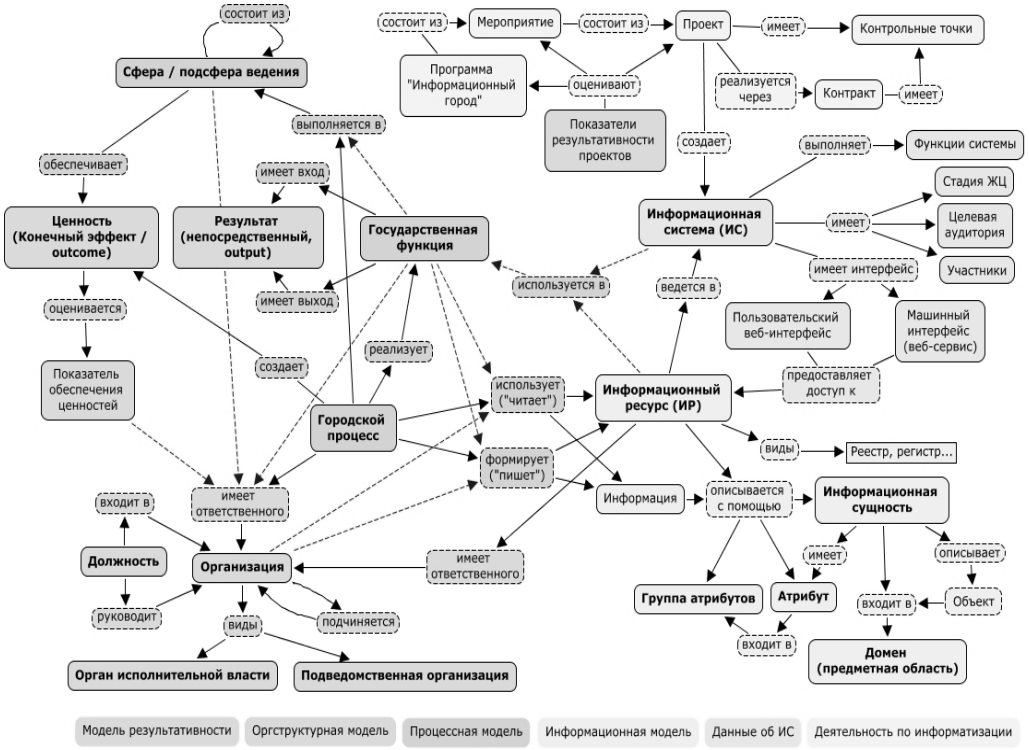


Рис. 6.30

Онтология для моделирования комплексов городского хозяйства

Онтология задает структуру модели, поддерживает интеграцию информации, а также облегчает коммуникации между различными заинтересованными сторонами. Для классов онтологии даны определения на естественном языке. Для поддержки коммуникаций онтология представляется графически в виде концептуальной карты (рис. 6.30), при задании структуры модели — в виде системы типов для моделируемых объектов (в соответствии с указанным выше «внутренним» форматом представления информации). Связи между классами, представленные синими пунктирными линиями на рисунке 6.30, в явном виде не вводятся в модели, а выводятся на основе продукционных правил вида «Если Городской процесс использует Информационный ресурс и Городской процесс имеет ответственную организацию, то Организация использует Информационный ресурс». Технически данные связи выводятся либо через «Составные проекции» (компонент формата «внутреннего» представления модели организации), либо через настройки отчетов из модели.

Для внешнего представления информации из модели служат формируемые из нее табличные отчеты и диаграммы [Гаврилова, Гулякина, 2008; Гаврилова, Кудрявцев и др., 2013; Kudryavtsev, Gavrilova, 2016] (см. табл. 6.5), размещаемые на портале информационно-справочной системы, которые позволяют представить информацию на разных уровнях обобщения: система городского управления в целом, комплекс городского хозяйства, городские процессы.

Для создания, хранения и наглядного представления информации о компонентах системного проекта была разработана специальная информационно-справочная система (ИСС).

Конечными пользователями системы являются отраслевые менеджеры и сотрудники дирекции департамента информационных технологий г. Москвы, специалисты отраслевых органов исполнительной власти, ответственные за информатизацию (функциональные заказчики), а также граждане или организация, являющиеся представителями экспертного сообщества и/или потенциальными поставщиками ИКТ для города. Наполнением ИСС занимаются специалисты по моделированию (разработчики модели), которые получают информацию от отраслевых экспертов и из нормативно-правовых документов. Часть информации попадает в ИСС из смежных систем, в частности, перечень ИС города Москвы и их комплексное описание; информация о проектах и мероприятиях по разработке, модернизации и сопровождению ИС, выполняемых в рамках программы «Информационный город».

Обобщенные функциональные возможности ИСС:

1. Моделирование ключевых комплексов городского хозяйства.
2. Интеграция информации о компонентах СП из различных источников (система моделирования, смежные системы).
3. Предоставление доступа к информации о компонентах СП для конечных пользователей.
4. Поддержка коллективной экспертной деятельности.

ИСС состоит из двух подсистем — моделера и портала (рис. 6.31). Назначение моделера: создание, редактирование, хранение и сопровождение моделей (процессной, информационной, результативности) ключевых комплексов городского хозяйства. Назначение портала: представление сведений о компонентах СП для конечных пользователей ИСС СП (в том числе из моделей ключевых комплексов городского хозяйства), а также поддержка работ по подготовке информации для наполнения модели.

На рисунке 6.32 представлен фрагмент системы типов объектов в модели, верхний уровень онтологии. На рисунке 6.33 представлен верхний уровень классификатора/справочника городских процессов, детализация онтологии. На рисунке 6.34 представлен нижний уровень классификатора/справочника городских процессов с конкретными городскими процессами (листовые элементы в иерархии). На рисунке 6.35 показана связь городского процесса с информационными ресурсами и содержащейся в них информацией (атрибуты информационных сущностей). На рисунке 6.36 приведен пример аналитического отчета из модели. На рисунках 6.37–40 представлены примеры страниц портала ИСС СП, в частности, главная страница, пример табличного отчета, графического, а также фасетный поиск объектов (отбор по нескольким основаниям).

Таблица 6.5

Систематизация отчетов и диаграмм по «Системному проекту»

Аспект / Уровень	Результативность	Деятельность	Организации	Информация	Информационные системы (ИС)
Система городского управления в целом	Ценности (конечные эффекты), создаваемые в сфере/подсфере Показатели результативности деятельности в сферах/подсферах (степень реализации ценностей) Таблица обеспечения ценностей (ценности — сферы/подсферы — ОИВ)	Сферы ведения Структура/дерево подсфер Описание подсферы Карта подсферы (взаимосвязь госфункций через результаты)	Подведомственные организации (факультативно) Таблицы атрибутов ОИВ (проекции) ОИВ — деятельность (сферы/подсферы, госфункции, процессы) ОИВ — деятельность — ценности ОИВ — деятельность — информация (информационные сущности и их атрибуты) ОИВ — деятельность — информация (информационные сущности и их атрибуты) — ИС	Домены (предметные области) Концептуальная модель (объекты реального мира, их свойства и связи) Таблицы (диаграммы) Домены — информационные сущности	Перечень ИС Классификация ИС
Комплекс городского хозяйства (отрасль)	Ценности, создаваемые госфункцией Таблица обеспечения ценностей (ценности — госфункции — ОИВ)	Классификатор государственных функций Описание госфункции (информационные входы/выходы) — контекстная диаграмма, паспорт		Информационные сущности (информационные объекты, их свойства и связи) UML-диаграмма классов Дерево (таксономия) атрибутов	ИС — госфункции ИС-поставщик — информационные объекты/атрибуты — ИС-получатель ИС — реестр (информационные объекты/атрибуты) — ИС-получатель
Городские процессы	Таблица обеспечения ценностей (ценности — процессы — ОИВ)	Классификатор городских процессов Описание процесса (контекстная диаграмма, паспорт) Диаграммы процессов (процессы — информация — реестры)		Информационные ресурсы, реестры (ИР) Таблицы (диаграммы) ИР — информационные сущности и их атрибуты	Таблицы, диаграммы ИС — реестры



Рис. 6.31

ИСС и смежные с ней системы

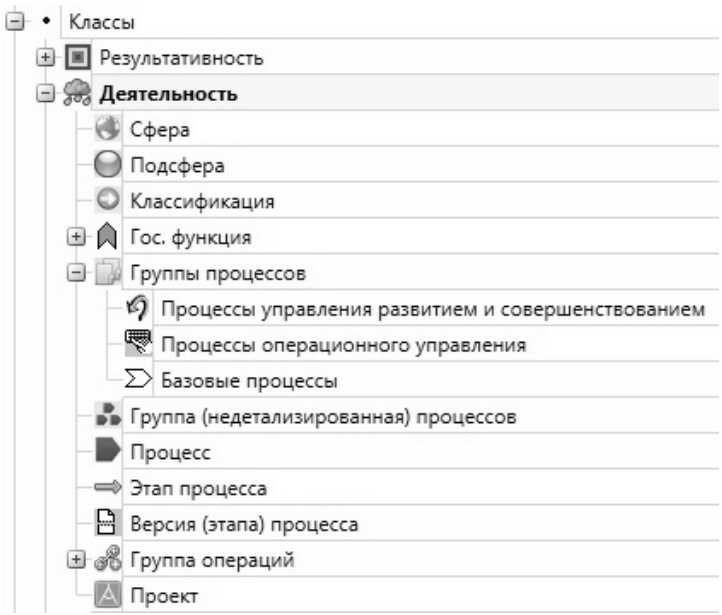


Рис. 6.32

Система типов объектов в модели, верхний уровень онтологии

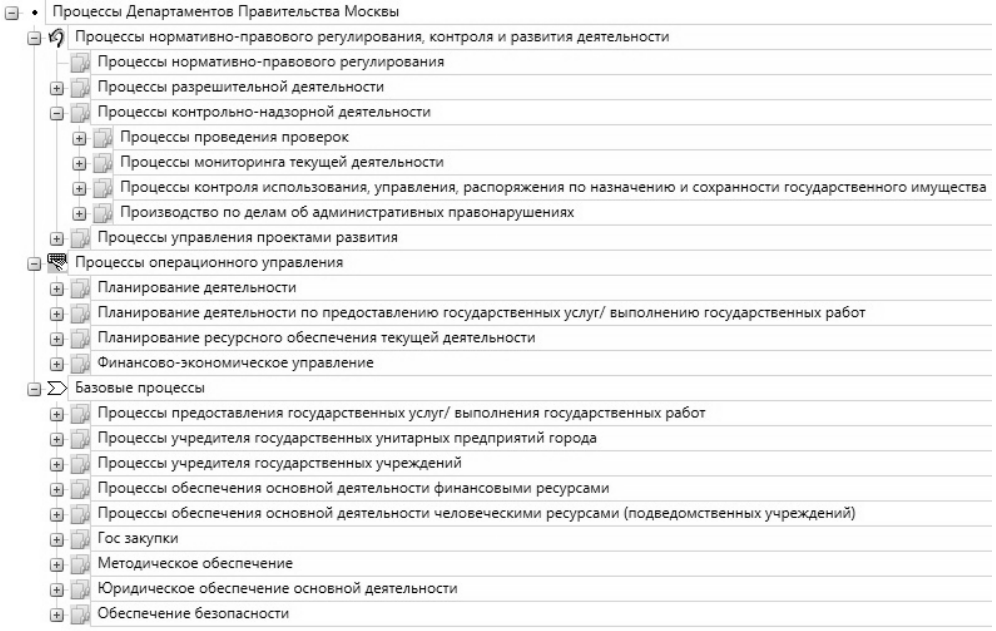


Рис. 6.33

Верхний уровень классификатора/справочника городских процессов, детализация онтологии

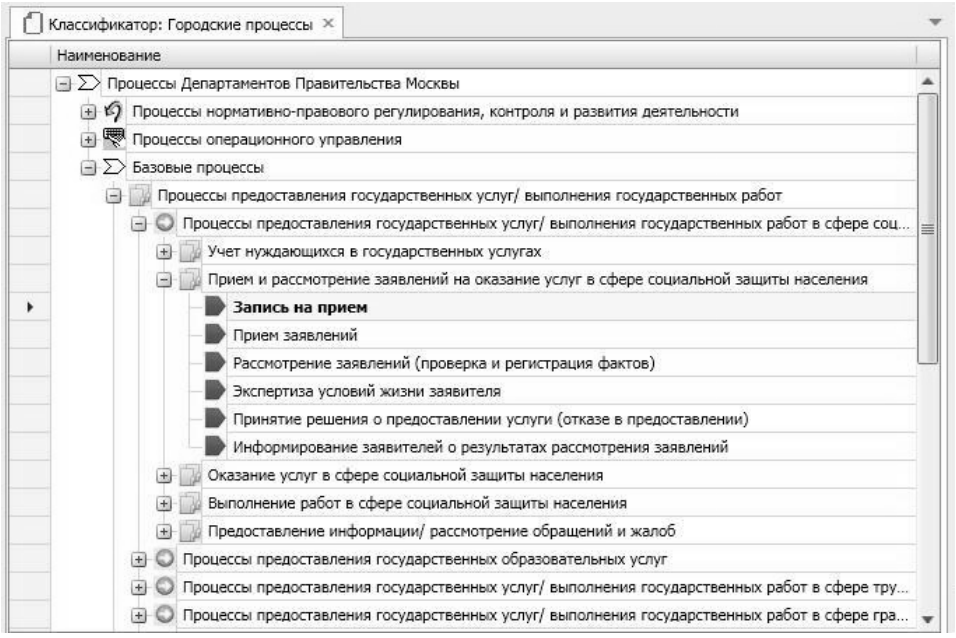


Рис. 6.34

Нижний уровень классификатора/справочника городских процессов

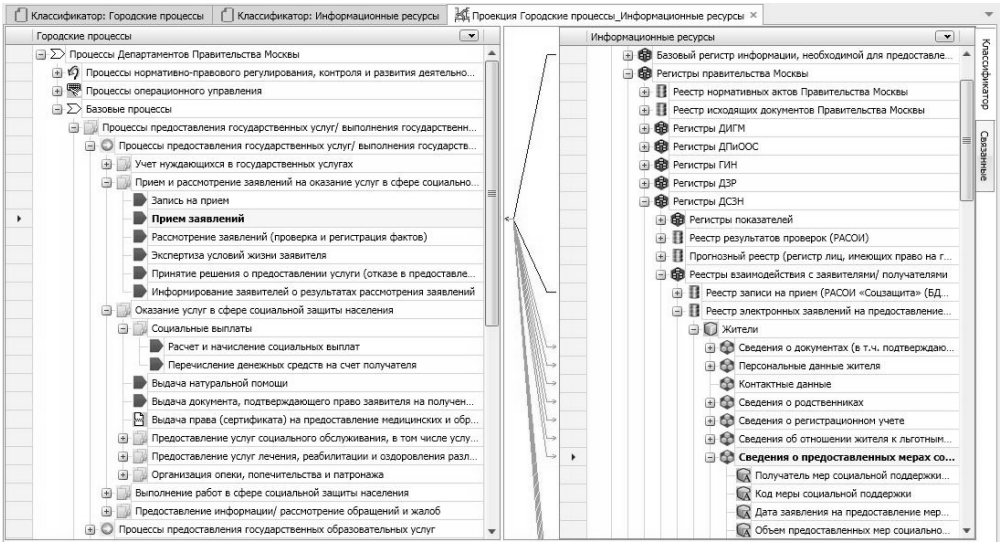


Рис. 6.35

Пример проекции Городские процессы — Информационные ресурсы в моделере ИСС СП



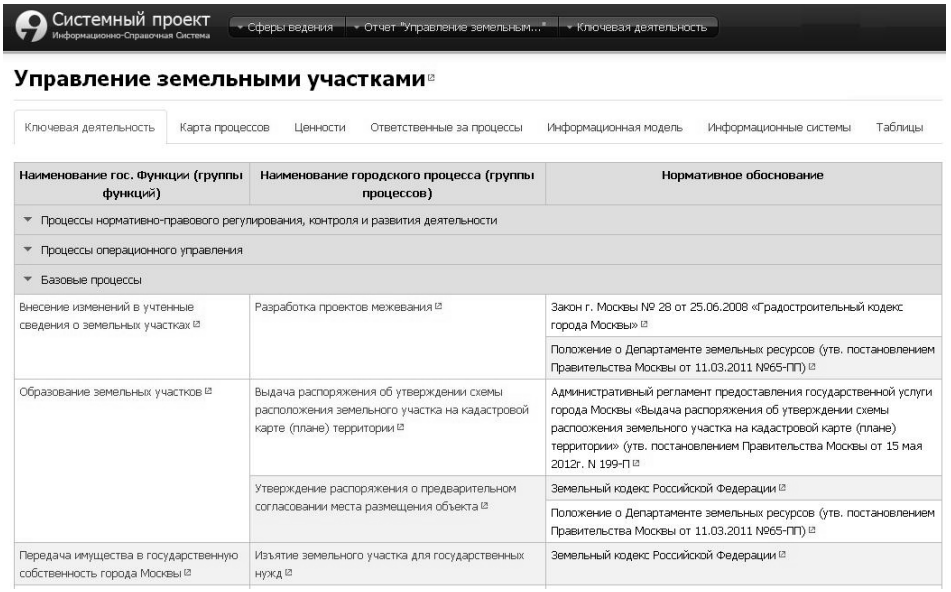
Рис. 6.36

Пример аналитического отчета из модели



Рис. 6.37

Главная страница портала ИСС СП



Наименование гос. Функции (группы функций)	Наименование городского процесса (группы процессов)	Нормативное обоснование
▼ Процессы нормативно-правового регулирования, контроля и развития деятельности		
▼ Процессы операционного управления		
▼ Базовые процессы		
Внесение изменений в учтенные сведения о земельных участках	Разработка проектов межевания	Закон г. Москвы № 28 от 25.06.2008 «Градостроительный кодекс города Москвы»
		Положение о Департаменте земельных ресурсов (утв. постановлением Правительства Москвы от 11.03.2011 №65-ПП)
Образование земельных участков	Выдача распоряжения об утверждении схемы расположения земельного участка на кадастровой карте (плане) территории	Административный регламент предоставления государственной услуги города Москвы «Выдача распоряжения об утверждении схемы расположения земельного участка на кадастровой карте (плане) территории» (утв. постановлением Правительства Москвы от 15 мая 2012г. N 199-П
	Утверждение распоряжения о предварительном согласовании места размещения объекта	Земельный кодекс Российской Федерации
		Положение о Департаменте земельных ресурсов (утв. постановлением Правительства Москвы от 11.03.2011 №65-ПП)
Передача имущества в государственную собственность города Москвы	Изъятие земельного участка для государственных нужд	Земельный кодекс Российской Федерации

Рис. 6.38

Пример табличного отчета на портале ИСС СП — Ключевая деятельность в подсфере

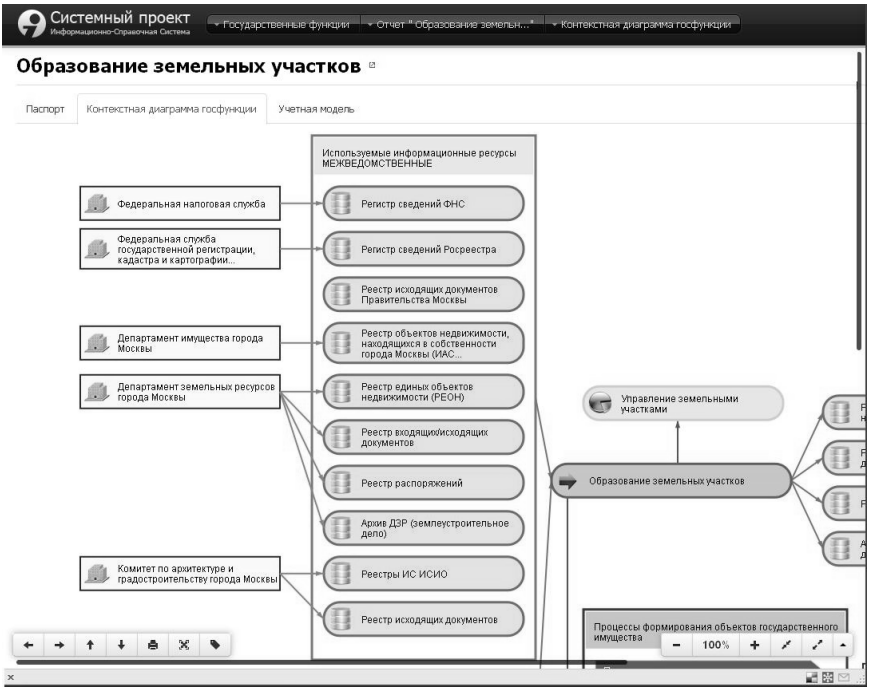
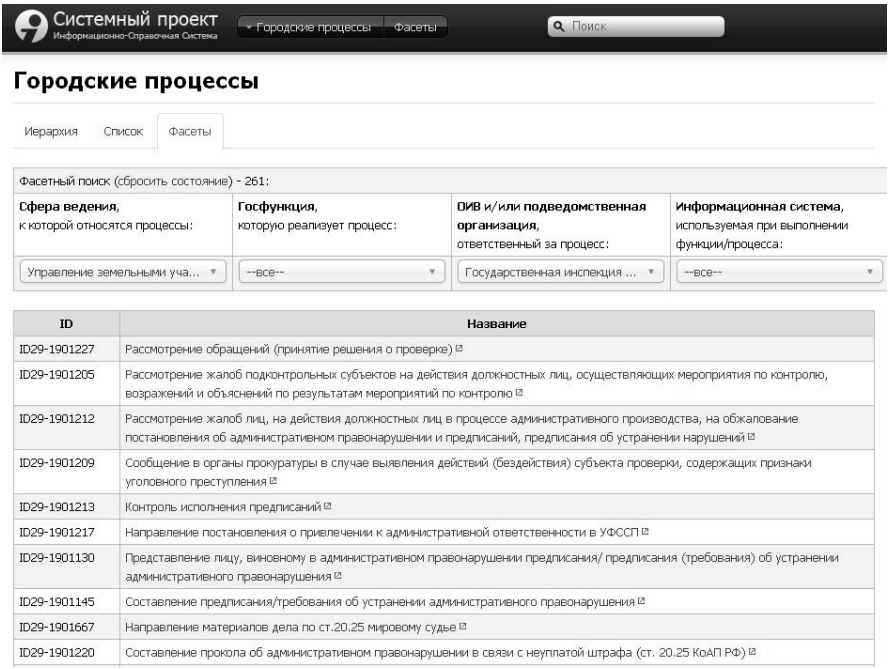


Рис. 6.39

Контекстная диаграмма госфункции на портале ИСС СП



ID	Название
ID29-1901227	Рассмотрение обращений (принятие решения о проверке)
ID29-1901205	Рассмотрение жалоб подконтрольных субъектов на действия должностных лиц, осуществляющих мероприятия по контролю, возражений и объяснений по результатам мероприятий по контролю
ID29-1901212	Рассмотрение жалоб лиц, на действия должностных лиц в процессе административного производства, на обжалование постановления об административном правонарушении и предписаний, предписания об устранении нарушений
ID29-1901209	Сообщение в органы прокуратуры в случае выявления действий (бездействия) субъекта проверки, содержащих признаки уголовного преступления
ID29-1901213	Контроль исполнения предписаний
ID29-1901217	Направление постановления о привлечении к административной ответственности в УФССП
ID29-1901130	Представление лицу, виновному в административном правонарушении предписания/ предписания (требования) об устранении административного правонарушения
ID29-1901145	Составление предписания/требования об устранении административного правонарушения
ID29-1901667	Направление материалов дела по ст.20.25 мировому судье
ID29-1901220	Составление протокола об административном правонарушении в связи с неуплатой штрафа (ст. 20.25 КоАП РФ)

Рис. 6.40

Фасетный поиск на портале ИСС СП, в данном случае городских процессов

Полученная в результате выполнения Системного проекта модель содержит описание 20 сфер ведения (отраслей городского хозяйства), более 1300 городских процессов, более 1000 информационных систем, 1200 информационных сущностей, более 800 информационных ресурсов.

В рамках работ по разработке СП были предложены критерии, позволяющие оценить степень соответствия информационно-коммуникационных систем принципам и требованиям Программы и СП, а также процедуры контроля соответствия. Оценка соответствия ИС Системному проекту состоит из следующих составляющих:

- соответствие Процессной модели Системного проекта;
- соответствие Информационной модели Системного проекта;
- соответствие Архитектурному проекту Системного проекта;
- соответствие целям реализации Государственной программы города Москвы «Информационный город (2012–2016 гг.)»;
- оценка экономической эффективности создания и эксплуатации информационно-коммуникационных систем (эффективности инвестиций в создаваемые информационные системы).

Разработанные модели, ИСС СП, а также критерии оценки соответствия были использованы для проведения экспертизы информационных систем г. Москвы.

Глава 7

СИСТЕМЫ НА ОСНОВЕ ОНТОЛОГИЙ

7.1. ИСПОЛЬЗОВАНИЕ ОНТОЛОГИЙ В ЭКСПЕРТНЫХ СИСТЕМАХ И СИСТЕМАХ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ[1]

Основным и наиболее важным компонентом экспертных систем (ЭС) является база знаний. Именно полнота и непротиворечивость представленных в ней знаний, определяют мощность экспертной системы и качество получаемых ею решений.

Онтология может составлять каркас базы знаний, создавать базис для описания основных понятий предметной области и служить основой для интеграции баз данных, содержащих фактические знания, необходимые для полноценного функционирования экспертной системы.

Кроме того, в терминах онтологии могут описываться экспертные правила, что значительно повышает их уровень описания и «понимаемость» пользователями-экспертами.

Система поддержки принятия решений (СППР) — это интерактивная автоматизированная информационно-аналитическая система, которая помогает лицу, принимающему решения (ЛПР), использовать данные и модели для решения его профессиональных слабоформализуемых задач.

СППР и ЭС — это системы практически одного класса, часто СППР в своем составе имеет несколько различных ЭС, поэтому все, что было сказано выше про роль онтологии в ЭС, относится и к СППР. Однако имеются аспекты использования онтологий, специфичные для СППР.

В виду слабой формализуемости решаемых задач очень важно иметь детальное непротиворечивое описание проблемной области, в рамках которой ЛПР решает свои задачи. Онтология является незаменимым инструментом для создания такого описания.

В большинстве типов СППР используются большие массивы разнородных данных и знаний. Благодаря тому, что онтология позволяет явно описывать семантику данных и знаний, она обеспечивает базис для их интеграции и совместного использования при решении задач.

При разработке ЭС или СППР, основанных на онтологиях, возможно использование онтологий, ранее разработанных для области знаний ЭС и СППР. Это позволяет переиспользовать уже проверенные на практике знания, что обеспечивает высокое качество создаваемых систем и их потенциальную интегрируемость с уже разработанными системами.

Онтологии могут использоваться как при проектировании и разработке ЭС и СППР (в управляемой онтологией разработке системы [Guarino, 1998, b]), так и в качестве полноценного компонента в процессе функционирования системы (в управляемой онтологией системе [Guarino, 1998, b]).

При разработке ЭС/СППР онтология может использоваться для:

- формирования и фиксации общего разделяемого всеми экспертами знания о предметной области (ПО);
- явной концептуализации ПО, позволяющей описывать семантику данных;
- обеспечения возможности переиспользования знаний;
- описания функционала ЭС или СППР (типов решаемых задач).

[1] Фрагмент из [Загорулько Ю. А., Загорулько Г. Б., 2011].

В процессе функционирования ЭС и СППР онтология может использоваться для обеспечения:

- совместного использования разнородных данных и знаний в рамках одной системы;
- процесса решения задач, составляющих функционал системы;
- лучшего понимания предметной области пользователями системы.

7.2. ОБЗОР ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ НА ОСНОВЕ ЗНАНИЙ[1]

В данном разделе приведены примеры ЭС на основе онтологий, классифицируемых по типу решаемой задачи: диагностика, поддержка принятия решений, мониторинг, интерпретация данных, прогнозирование, обучение, планирование, управление и проектирование.

1. Диагностика

Siemens PLMS (Plant Lifecycle Management Systems) COMOS[2] — система управления жизненным циклом промышленного объекта (электростанции, завода...), поддерживающая его проектирование, разработку и эксплуатацию. В работе [Legat et al., 2011] сотрудниками Siemens представлена технология и реализующий ее модуль системы COMOS для автоматической диагностики и контроля разрабатываемого и эксплуатируемого промышленного объекта. Демонстрация механизма диагностики сделана на примере разрабатываемой системы Simodrive 611 — системы управления приводом, которая используется в различных решениях по автоматизации, требующих гибкой синхронизации сложных устройств. Simodrive 611 состоит из управляющего устройства (УУ) и источника питания (ИП). Модуль диагностики идентифицирует 2 вида ошибок: конфигурации и помехи в системе, определяемых через атрибуты компонент Simodrive 611 — УУ имеет сигнал функции преобразования, а ИП имеет 2 светодиодных индикатора — статуса и загрузки. Система разработана на основе онтологий. Симптом определяется через признаки ошибки (Классы «Функция преобразования УУ», «статус ИП» и «загрузка ИП» — подклассы класса «Симптом»). Каждая возможная ошибка — подкласс класса «Диагноз». Связь между классами устанавливается через отношения (например, «isDiagnosisFor», «hasDiagnosis»). Если функция преобразования УУ неисправна и индикатор статуса ИП отключен, то диагноз — «Ошибка конфигурации». Если же отключены оба индикатора, то диагноз — «Помехи».

2. Мониторинг

iOSC3 — система поддержки принятия решений для контроля и лечения пациентов с острыми сердечнососудистыми заболеваниями. Функциональность системы обеспечивает специально разработанная онтология C3O (Critical Cardiac Care Ontology) [Martínez-Romero et al., 2013]. Знания предметной области в системе структурированы в концептуальную модель: ключевые понятия (например, частота сердечных сокращений, инфузионный насос, лекарство и др.) и отношения между ними «a kind of» (например, нитроглицерин > сосудорасширяющее средство > лекарство) организованы в иерархическую таксономию. Сама онтология состоит из 40 классов, одного свойства объекта (object property) «atPump», представляющего изменение дозы лекарства на инфузионном насосе, пяти свойств типа данных (datatype property). Логика принятия решений представлена в виде набора if-then правил, связанных с онтологией. Онтология была формализована при помощи языка OWL DL (расширенная версия языка OWL, Ontology Web Language) и

[1] Раздел подготовлен Рупасовой Викторией совместно с Кудрявцевым Д. В.

[2] http://w3.siemens.com/mcms/plant-engineering-software/en/Pages/Default.aspx

построена в редакторе Protégé, правила вывода написаны на языке SWRL (Semantic Web Rule Language). Пример таких правил на естественном языке: ЕСЛИ к пациенту подключен инфузионный насос с лекарством амин и давление пациента упало, ТО скорость инфузии в насосе должна быть увеличена.

3. Поддержка принятия решений

Situation-Based Access Control (SitBAC) [Beimel et al., 2011] — система контроля доступа к данным пациентов в медицинских учреждениях. Система спроектирована на основе онтологии, описывающей сценарии доступа к данным пациентов. Центральным понятием онтологии является «Ситуация» (SitClasses), шаблон описания которой содержит 6 основных сущностей, их связей и атрибутов: 1. Человек, делающий запрос к данным (Data-Requestor), 2. Пациент, 3. Электронная медицинская карта (Electronic Health Record, EHR), 4. Задача/Операция, на выполнение которой получен запрос (например, посмотреть метод лечения, обновить информацию), 5. Юридическое обоснование, которое авторизовывает запрашиваемую задачу (Legal-Authorization), 6. Ответ, разрешающий или запрещающий доступ к данным (Response). SitFly — класс, представляющий индивидуальный запрос пользователя. При помощи SWRL-правил на основании свойств классов и их отношений определяется сценарий (SitClasses) для запроса SitFly и принимается решение о разрешении/запрете доступа к данным. Система позволяет работать со сложными сценариями, включающими проблему интероперабельности, когда представитель одной организации хочет получить данные другой организации.

4. Интерпретация данных

Примером интерпретирующей системы является Personal Assistant (PA) — встраиваемая система интерпретации пользовательских запросов, записанных на естественном языке, способная выполнять задачи с помощью интуитивно понятного интерфейса [Paraiso et al., 2007]. Онтологии используются для семантической интерпретации полученных знаний и представления задач. Процесс интерпретации высказывания осуществляется в две стадии: а) грамматический и синтаксический анализ и б) семантическая интерпретация (задача онтологий). Подход к семантической интерпретации основан на том, что смысл высказывания можно понять, определив в нем понятия и их атрибуты. Рассмотрим фразу «Найдите все статьи об агентах». Допустим, что Personal Assistant будет встроен в некоторую базу данных. Предварительно система определяет, относится ли запрос к соответствующей теме, используя онтологию и словарь. Далее модуль грамматического и синтаксического анализа возвращает матрицу, содержащую список маркеров/слов и их синтаксические классификации. Personal Assistant определит, что «поиск» — это экземпляр класса «Действия» (используется также список синонимов, например, «искать», «выводить» или «находить» являются синонимами), «статья» — это класс, а «агенты» — это ее свойство — тема статьи. Таким образом запрос будет интерпретирован.

5. Прогнозирование

Altman Z-score — система прогнозирования банкротства предприятия на основе онтологий. Онтологии, представленные в труде [Martin et al., 2011], описаны при помощи языка OWL и разделены на несколько уровней. Онтологии верхнего уровня описывают общие понятия, как пространство, время, которые не зависят от конкретного домена; онтологии предметной области описывают словарный запас, относящийся к области домена; онтологии задач описывают лексику, связанную с задачей; онтологии приложения описывают специфичные понятия, т. е. онтология представляет собой таксономию: например, класс Активы имеет подклассы Текущие, Долгосрочные, Другие; в свою очередь класс Текущие содержит подклассы Наличные деньги и Краткосрочные инвестиции.

На вход системе подается финансовый отчет. После составления онтологии происходит кластерный анализ данных и по формуле, предложенной Эдвардом Альтманом, подсчитывается коэффициент Z — индекс банкротства предприятия. Так, например, если $Z < 2.675$, то с уверенностью в 95% можно сказать, что предприятие станет банкротом в течение года.

6. Обучение

SQL-Guide — разновидность AWBES (adaptive Web-based educational systems) адаптивной образовательной системы для тренировки умений в области SQL-запросов к базам данных [Sosnovsky et al., 2008]. Задания для студентов основаны на наборе баз данных и множестве желаемых результатов выполнения запросов, для получения которых студент должен сделать соответствующие запросы. Система оценивает ответ, учитывая правильность результата, время выполнения запроса и обеспечивает удобную обратную связь. Онтология SQL в системе описывает классы — команды и зарезервированные слова SQL (WhereClause подкласс Clause, SelectStatement подкласс Statement и т. д.) и отношения между ними (SelectStatement uses WhereClause, WhereClause isUsedIn SelectStatement), доступна по адресу http://www.sis.pitt.edu/~paws/ont/sql.owl. С помощью онтологии индексируются задания для студентов, поддерживается разработка адаптивного образовательного контента и интеграция обучающих систем (например, SQL Guide и SQL Tutor) [Brusilovsky et al, 2008; Sosnovsky et al, 2008].

7. Планирование

PASER (Planner for the Automatic Synthesis of Educational Resources) — система автоматического синтеза образовательных программ [Kontopoulos et al., 2008], способная динамически строить программы обучения из учебных объектов с учетом индивидуальных качеств учащегося, его предпочтений, потребностей и способностей. Система состоит из (а) репозитория метаданных, в котором хранятся метаданные учебных объектов (Learning Object Metadata), профили учащихся и онтология преподаваемой предметной области, (б) механизм формирования запросов и логического вывода и (с) непосредственно планировщик. Онтология в PASER разработана при помощи RDF (Resource Description Framework), обеспечивающего механизм описания компетенций и свойств. Она описывает область искусственного интеллекта и состоит из 310 компетенций, структурированных в isPartOf-иерархию. Например, корневой элемент имеет подкомпетенции, такие как Machine Learning, Planning, Knowledge Representation и т. д., которые также имеют свои подкомпетенции и т. д. Компетенции также описывают цели и достижения ученика. Устройство рассуждений устанавливает связи между объектами изучения и характеристиками ученика. План обучения строится на анализе онтологического описания накопленных знаний ученика и его целей.

8. Управление

Telehealth Smart Home — система физических и программных компонентов, обеспечивающих безопасность пожилых пациентов, потерявших когнитивную автономию [Latfi et al., 2007]. Работа системы направлена на предотвращения возможных опасных для жизни ситуаций. В контур управления также включены медсестра, родственники пациента, семейный врач, менеджер системы и др. Telehealth Smart Home использует комплекс онтологий для разработки и конфигурирования программных приложений, а также для человеко-машинного взаимодействия. Онтология Habitat описывает структуру размещения (классы комнаты, двери, окна и т. д.). PersonAndMedicalHistory онтология дает описание человека, который нуждается в уходе, и его историю болезни, а также описание человека-помощника (классы имя, возраст, болезнь и т. д.). Equipment онтология определяет оборудование умного

дома, участвующее в системе (классы мебель, бытовая техника и техническое оборудование). При помощи Software applications онтологии описаны программные модули системы. Онтология Task детализирует задачи системы. Важные аспекты в контексте дистанционного мониторинга, такие как привычки и критические физиологические параметры описаны при помощи онтологии Behavior. И наконец, онтология Decision описывает поведение системы в критической ситуации (например, вызов неотложной помощи). Основной сбор данных осуществляется с помощью датчиков (элементов класса TechnicalEquipment — подкласса Equipment). После обработки эти данные станут экземплярами онтологии Behavior и двух классов онтологии PersonAndMedicalHistory: MedicalHistory и Patient. Классы Equipment через отношения связаны с классами Habitat: это позволяет определить расположение оборудования в доме. Изучение поведения пациента (его когнитивного расстройства) осуществляют байесовские сети путем обработки данных, представленных в онтологиях. Далее, на основе полученных результатов принимается решение (Decision) о возможных действиях для обеспечения безопасности пациента.

9. Проектирование

Smart Architect (умный архитектор) — интеллектуальная система, способная автоматически генерировать архитектурное строение древнекитайского стиля [Liu et al., 2008]. Используя онтологический подход к анализу различных архитектурных стилей, система преобразует геометрии примитивов в семантические компоненты архитектуры. Онтология системы состоит из трех компонент/уровней, относящихся к 3-м основным участникам процесса проектирования: дизайнера, программиста и машины. Концептуальная модель онтологии имеет иерархическую структуру. Например, класс Architecture состоит из двух подклассов — Ancient architecture и Modern architecture. В свою очередь Ancient architecture — класс-родитель для классов Chinese ancient architecture, Indian ancient architecture, European ancient architecture и т. д. Для получения проекта здания пользователю необходимо выбрать значения 3 параметров: городские карты, семантические компоненты, описания строительных стилей и конструкций (можно изменить эти входные параметры для определения собственных целей моделирования). Эти параметры заранее определены в онтологии уровня дизайнера и через отношения связываются с интерактивным инструментарием моделирования конструкций. Далее при помощи онтологии уровня программиста совершается построение моделей, согласование конструкции с планом местности, проверка стиля. Онтология на уровне машины содержит информацию о визуализации конструкции.

7.3. ПРИМЕРЫ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ НА ОСНОВЕ ОНТОЛОГИЙ

Использование онтологии для создания автомобилей без водителя[1]

В сфере интеллектуализации транспортных систем важнейшей целью является достижение полной автономии автомобилей во всех возможных ситуациях на дороге (погодные условия, интенсивность движения и т. д.). Наивысших результатов в сфере управления автомобилем без водителя в городских условиях удалось достичь компании Google (Google driverless car, http://www.google.com/ selfdrivingcar/) в 2010 году. Несмотря на успехи в этой области по-прежнему достаточно трудно обеспечить полную автономию транспортного средства в связи с уязвимостью датчиков, которые имеют предопределенный рабочий диапазон и могут выйти из строя.

[1] Раздел подготовлен Ульяной Дудко на основе [Pollard et al, 2013], редактирование — Дмитрий Кудрявцев.



Рис. 7.1

Примеры и эволюция автомобилей без водителя

В работе [Pollard et al, 2013], являющейся частью проекта Link & Go[1], рассмотрен подход, основанный на онтологической модели представления знаний, в котором последовательно определяется степень автоматизации автомобиля, начиная от ручного управления и заканчивая полностью автоматическим. Также в примере представлена онтология для оценки ситуации (состояние датчиков, погодные условия, состояние водителя), после чего в качестве примера были определены логические правила, связывающие онтологию оценивания ситуации с уровнями автоматизации автомобиля.

На рисунке 7.2 режимы автоматизации разделены на пять слоев, соответствующих различным уровням управления автомобилем. Вертикальная шкала указывает на требования к уровню восприятия и оценки ситуации, а горизонтальная — на тип принимаемых системой решений.

1) Слой 1: Продольное управление. На данном слое водитель не обязан следить за скоростью: система может принять режим «Круиз-контроль», в котором транспортное средство поддерживает постоянную скорость. В режиме «Динамическая установка скоростного типа» должна быть оценена максимальная разрешенная скорость (P2), в соответствии с дорожными знаками и картой окружающей среды. Режим «Автономный круиз-контроль» подразумевает авторегулирования скорости автомобиля и поддержания безопасного расстояния с впереди идущим транспортным средством. Он разрешен только в том случае, если положение, скорость и ускорение последнего могут быть точно определены (P3). Режим «Остановка/Старт» доступен, когда система может самостоятельно управлять началом движения. Наконец, в режиме «Кооперативный круиз-контроль» движущееся впереди транспортное средство может транслировать информацию (C1) для того, чтобы поддерживать безопасную дистанцию.

[1] http://research.akka.eu/innovation/linkandgo.php

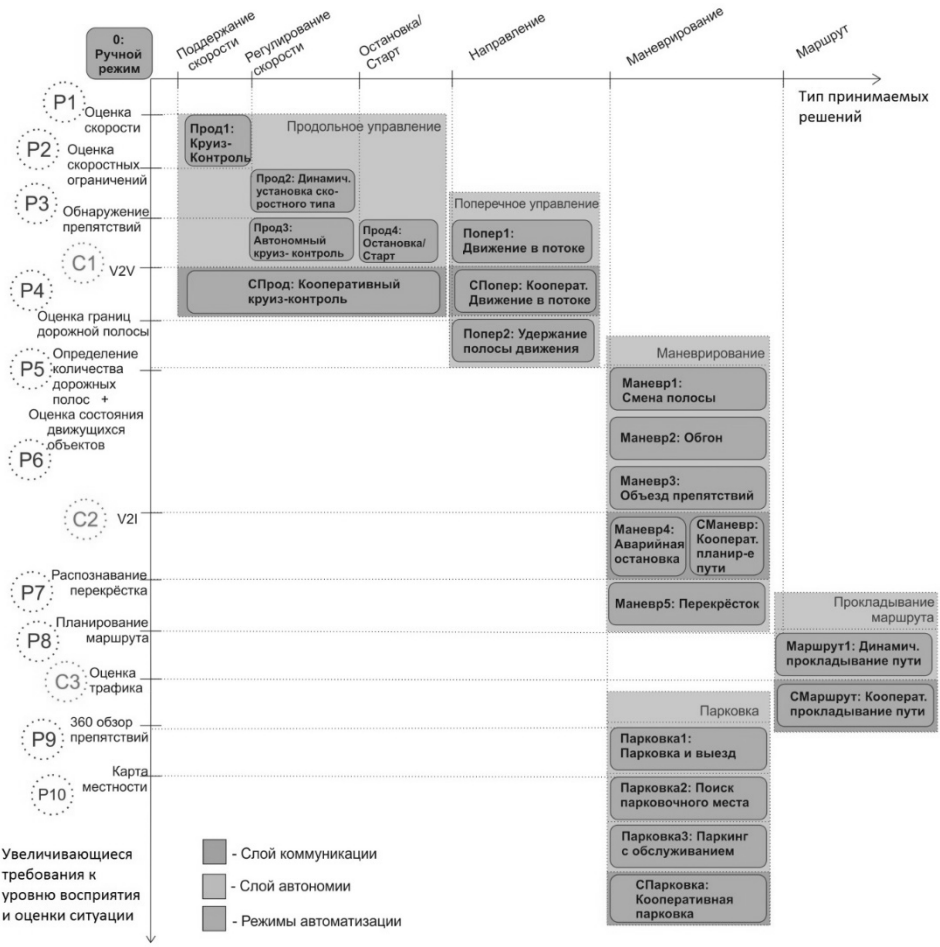


Рис. 7.2

Онтология степеней автономности/автоматизации автомобиля [Pollard et al., 2013]

2) Слой 2: Поперечное управление. Второй слой означает, что система берет на себя рулевой контроль. Для движения в транспортном потоке никаких новых датчиков восприятия не требуется. Если система способна распознать границы дорожной полосы (P4), то может быть активирован режим «Удержание полосы движения».

3) Слой 3: Маневрирование. Данный слой требует глубокого визуального анализа окружающей среды. Например, для того чтобы активировать смену полосы, система должна уметь распознавать количество полос на дороге, а также их направление (P5). Для режима обгона и объезда препятствий необходимо оценивать состояние движущихся объектов сзади и сбоку (P6). Если водитель теряет управление, например, из-за внезапной потери сознания, то «Помощник аварийной остановки» может распознать опасность и автоматически взять на себя кон-

троль для остановки автомобиля. Режим «Перекресток» является одним из самых сложных, так как необходимо иметь в виду знаки светофора, дорожной разметки, приоритет транспортных средств (Р7). Даже если обстановка тщательно исследована, иногда необходимо копировать поведение водителя на перекрестке: медленно выехать для получения лучшего обзора, пропустить поперек идущие машины, потом поехать.

4) Слой 4: Парковка. На четвертом слое важно иметь обзор 360^0 для распознавания препятствий (Р9), а также карту местности для поиска парковочного места (Р10).

5) Слой 5: Прокладывание маршрута. Динамическое прокладывание маршрута полностью автоматизировано. В кооперативном режиме автопилот получает информацию о трафике для объезда пробок.

Система рассуждений на уровнях автоматизации

Для построенной модели определены два уровня рассуждений: рассуждение на уровне слоя и рассуждение на уровне способностей восприятия. Для активации какого-либо слоя необходимо, чтобы все вышестоящие слои были активированы. Например, если слой продольного управления неактивен, то неактивен и слой поперечного управления, и слой маневрирования, и все нижестоящие. Это может быть записано так:

$$\neg\, Прод \Rightarrow \neg\, Попер \,\&\, \neg\, Маневр \,\&\, \neg\, Маршрут \,\&\, \neg\, Парковка. \qquad (1)$$

Второй уровень рассуждений определяет можно ли активировать какой-либо режим на основании полученной информации от устройств восприятия и оценки ситуации. Например, Р1 — восприятие, которое отвечает за оценку скорости автомобиля. $\neg$ Р1 означает, что данная способность не активирована. Таким образом, мы можем сказать, что если есть возможность оценить скорость (датчики работают корректно), то может быть активирован режим «Круиз-контроль» (см. рис.7.2 — «*Прод*1»):

$$\text{Р1} \Rightarrow Прод1. \qquad (2)$$

Деактивация режима означает отключение всех нижеследующих режимов. Другими словами, если не активирован «Круиз-контроль», то режимы «Динамическая установка скоростного типа», «Автономный круиз-контроль», «Остановка/Старт» также неактивны:

$$\neg\, Прод1 \Rightarrow \neg\, Прод2 \,\&\, \neg\, Прод3 \,\&\, \neg\, Прод4. \qquad (3)$$

Только если все режимы слоя активны может быть активирован и сам слой:

$$Прод1 \,\&\, Прод2 \,\&\, Прод3 \,\&\, Прод4 \Rightarrow Прод. \qquad (4)$$

Кооперативные режимы не влияют на активацию слоя в целом, однако для их включения должны быть доступны все вышестоящие режимы слоя:

$$\text{С1} \,\&\, Прод1 \,\&\, Прод2 \,\&\, Прод3 \,\&\, Прод4 \Rightarrow \text{С}Прод. \qquad (5)$$

Представление модели для оценки ситуации

Оценка текущей обстановки на дороге может быть определена через шесть модулей (рис. 7.3). Эти модули содержат подуровни и каждый элемент нижнего уровня должен быть оценен. Для числовых значений (серые блоки на рис. 7.3) не только рассчитывается значение, но и оценивается ошибка для определения точности полученных с устройства данных путем сравнения с пороговым значением. Для символьных значений должен быть предусмотрен доверительный интервал (бело-серые блоки на рис. 7.3). Далее перечислено описание каждого модуля.

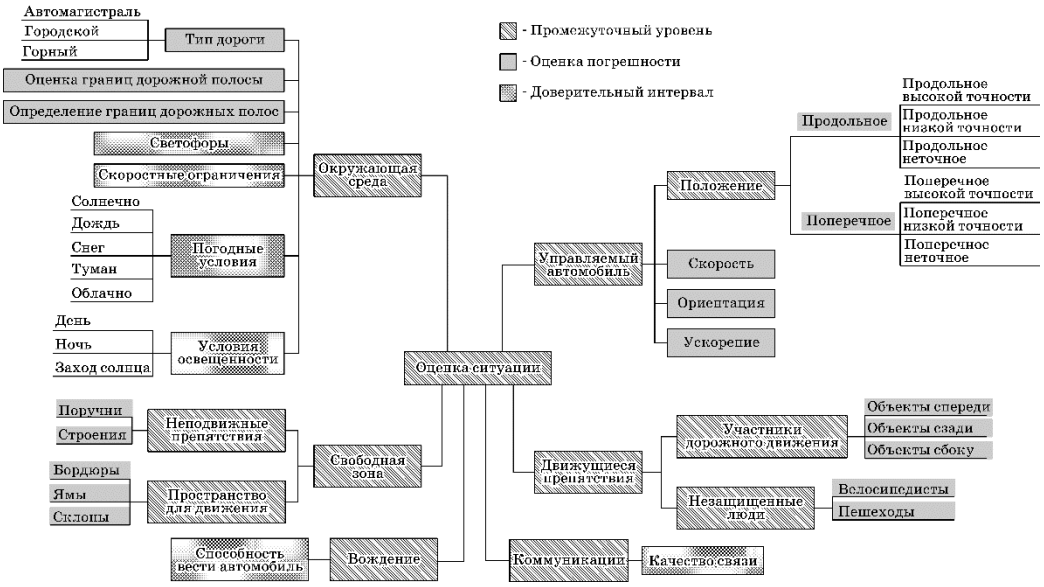


Рис. 7.3

Онтология для оценки ситуации [Pollard et al., 2013]

- Водитель. В данном случае необходимо оценить способность человека, сидящего за рулем, вести автомобиль. Это может быть определено через положение головы, рук, направления взгляда.
- Управляемый автомобиль. Помимо точного расположения машины, необходимо также оценивать погрешность ее устройств, определяющих продольную и поперечную координаты, скорость, направление, а также ускорение.
- Коммуникация. Качество связи определяется способностью транслировать данные с хорошей скоростью передачи, низким уровнем потерь данных, короткой задержкой и другими аспектами.
- Свободная зона. Данный модуль содержит в себе оценку местности на наличие неподвижных препятствий и свободного пространства для движения автомобиля. Здесь важно определять, есть ли на пути строения, поручни, бордюры, ямы и т. д.
- К движущимся препятствиям могут быть отнесены как участники дорожного движения (автомобили, автобусы), так и незащищенные люди (пешеходы, велосипедисты). Цель заключается в наилучшей оценке их состояния, положения, скорости и ускорения.
- В рамках оценки окружающей среды должны быть определены погодные условия, дорожные знаки, светофоры, скоростные режимы и т. д.

Пример:

Целью примера является оценка максимально разрешенного уровня автоматизации $\hat{M}_k$ на каждой итерации k для цикла управления и планирования маршрута.

Предположим, что автомобиль не может самостоятельно управлять движением на льду. Тогда ответственность за ведение машины ложится на плечи водителя. Это правило можно записать так:

$$\text{снежно} \Rightarrow \tilde{M}_k = 0 \quad (6)$$

В условиях дождя характеристики лазерных датчиков становятся хуже. Это означает, что обнаружение препятствий (P3) уже недостаточно точное — чем сильнее дождь, тем больше погрешность лазера, т. е. автоматизация остается на уровне второго режима:

$$\text{дождь} \Rightarrow \neg \text{P3} \Rightarrow \tilde{M}_k = \textit{Прод}2 \quad (7)$$

Туман влияет на область видимости, распознавание границ дорожной полосы (P4), скоростных ограничений (P2). Однако, если поперечная $(latHP)$ и продольная $(longHP)$ ориентации автомобиля достаточно точны, то ограничение дороги и скорости может быть также определено. Но если, к примеру, поперечная ориентация нарушена ($\neg latHP$), то некоторые функции также утрачиваются:

$$\text{туман} \;\&\neg latHP \;\& longHP \Rightarrow \neg P4 \;\& P2 \quad (8)$$

Для того чтобы включить водителя в модель, следует сделать различие между уровнем автоматизации без водителя $\tilde{M}_k$ и этим же уровнем с водителем $\hat{M}_k$. Таким образом, если часть информации, не включающая сведения о состоянии водителя, была использована для подсчета $\tilde{M}_k$, то этот максимальный режим автоматизации должен также учитывать текущее состояние водителя. Например, если текущий режим автоматизации подразумевает вмешательство водителя, а последний не готов вести автомобиль ($\neg DS$), то уровень автоматизации не может уменьшаться.

$$(\tilde{M}_k < \hat{M}_{k-1}) \;\&\neg DS \Rightarrow \hat{M}_k = \hat{M}_{k-1} \quad (9)$$

В данном случае водителю сначала дается предупреждение, что система собирается вернуть ему управление. Данный принцип представляет собой обобщенный алгоритм для соблюдения безопасности на дороге.

Система управления автомобилем без водителя была построена с помощью набора таких средств, как Protege, SWRL и Pellet (см. обзор инструментов далее).

Система OntoWEDSS[1]

Система OntoWEDSS [Ceccaroni et al, 2004] предназначена для поддержки принятия решений при управлении процессом очистки воды на водоочистных станциях. Эта система дополняет классические рассуждения на основе правил (rule-based reasoning) и на основе прецедентов (case-based reasoning) рассуждениями с использованием предметной онтологии (domain ontology). Онтология начинает «работать», когда другие системы рассуждений заходят в тупик.

Если первые две системы вывода не позволяют в полной мере моделировать микробиологию загрязнения воды и основываются только на простых правилах или прецедентах, то онтология позволяет явно моделировать эту область, в частности, задавать связи между (1) микроорганизмами и проблемными ситуациями (видами и ситуациями загрязнения) и (2) аномальными состояниями водоочистной станции и предлагаемыми решениями по приведению их в норму.

[1] Фрагмент из [Загорулько Ю. А., Загорулько Г. Б., 2011].

В системе используется признанная экспертами в данной области знаний онтология WaWO (WasteWater Ontology) [Ceccaroni et al, 2000]. Эта онтология транслируется с языка представления знаний Ontolingua в язык программирования Common LISP, на котором реализована система. После этого онтология выступает как полноправный компонент системы.

Иерархическая организация категорий онтологии WaWO (рис. 7.4) выражается с помощью языка представления знаний системы Ontolingua, а аксиомы языка KIF (Knowledge Interchange Format) используются для ответа на вопросы, анализа запросов и общих рассуждений. Аксиомы онтологии частично моделируются, но не реализуются полностью в языке KIF, поэтому основная часть вывода выполняется другими средствами.

Рассмотрим пример вывода в системе OntoWEDSS с помощью онтологии. При этом мы будем обращаться к рис. 7.5, на котором приведен фрагмент онтологии WaWO. Пользуясь терминологией Sowa [Sowa, 2000], вывод на онтологии можно объяснить как последовательность оккарентов (occurents), т. е. ситуаций или состояний каких-либо объектов. Заметим, что на рисунке 7.5 простые прямоугольники играют роль категорий (или категорий явлений, или классов) и всегда являются частью иерархии понятий, а окружности обозначают отношения.

Например, Filamentous-Bacteria (Нитчатые бактерии) — это то, что вызывает или является эффектором оккарента Filamentous-Bacteria-Excessive-Proliferation (Чрезмерный рост нитчатых бактерий). Будучи эффекторной частью таксономической (ветка Microfauna) или операционной (ветка Microthrix-Parvicella) семантических структур, этот оккарент может быть связан как с каким-нибудь классом микроорганизмов, так и с конкретным состоянием водоочистной станции (Filamentous-Dominant-AT).

Результатом этого оккарента является состояние Bulking-Sludge-Filamentous (Нитчатое вспучивание ила), которое само является эффектором оккарента Bulking-Sludge-Consequences-Avoidance (Ликвидация последствий вспучивания ила). Результат же последнего оккарента — это поиск Bulking-Solution (Решения проблемы вспучивания ила) в его ветке иерархии понятий. Аксиомы, являющиеся, по существу, правилами, управляют поиском специфического или неспецифического решения в соответствии с исходным эффектором.

Окончательным результатом вывода на онтологии может быть, например, то, что присутствие определенного количества нитчатых бактерий вызовет выработку рекомендации пользователям системы, требующую добавить определенные химикаты в сточную воду.

Следует заметить, что даже если такой же результат может быть получен в системе с помощью вывода на правилах, сама по себе структура онтологии гарантирует гораздо лучшую понимаемость, более простую проверку непротиворечивости и большую степень переиспользуемости знаний.

Введение онтологического компонента в СППР, в частности, позволяет обеспечить (1) более надежный процесс очистки воды и (2) переносимость (portability) системы. Так, если без использования онтологии система успешно диагностировала и разрешала от 60 до 73% проблемных ситуаций, то использование онтологии позволяло достичь значений этого показателя от 73 до 100%. Это происходило благодаря следующим факторам: (1) онтология WaWO позволяла разрешить ситуации, оказавшиеся «тупиковыми» для механизма вывода на правилах, благодаря тому, что имела более богатый набор аксиом по сравнению с этим механизмом; (2) онтология WaWO предоставляла дополнительные знания о микроорга-

низмах, которые не могли быть использованы в механизмах классических рассуждений.

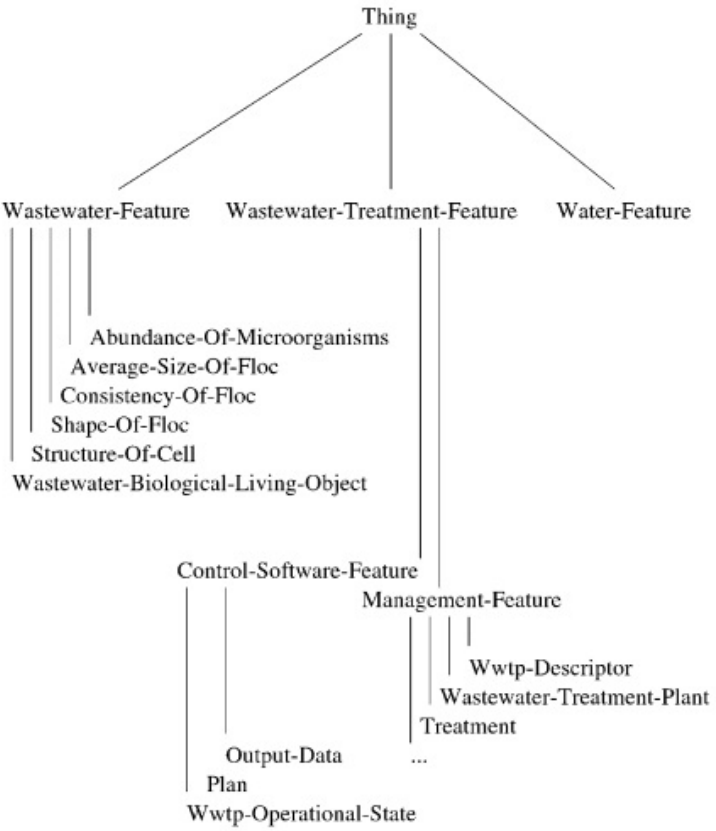


Рис. 7.4
Верхняя часть иерархии понятий онтологии WaWO

Также в системе используются и другие традиционные преимущества онтологии. В частности, возможность использования согласованной (разделяемой всеми) терминологии предметной области с точно определенной семантикой позволяет интегрировать и использовать информацию из разных источников (например, с разных водоочистных станций).

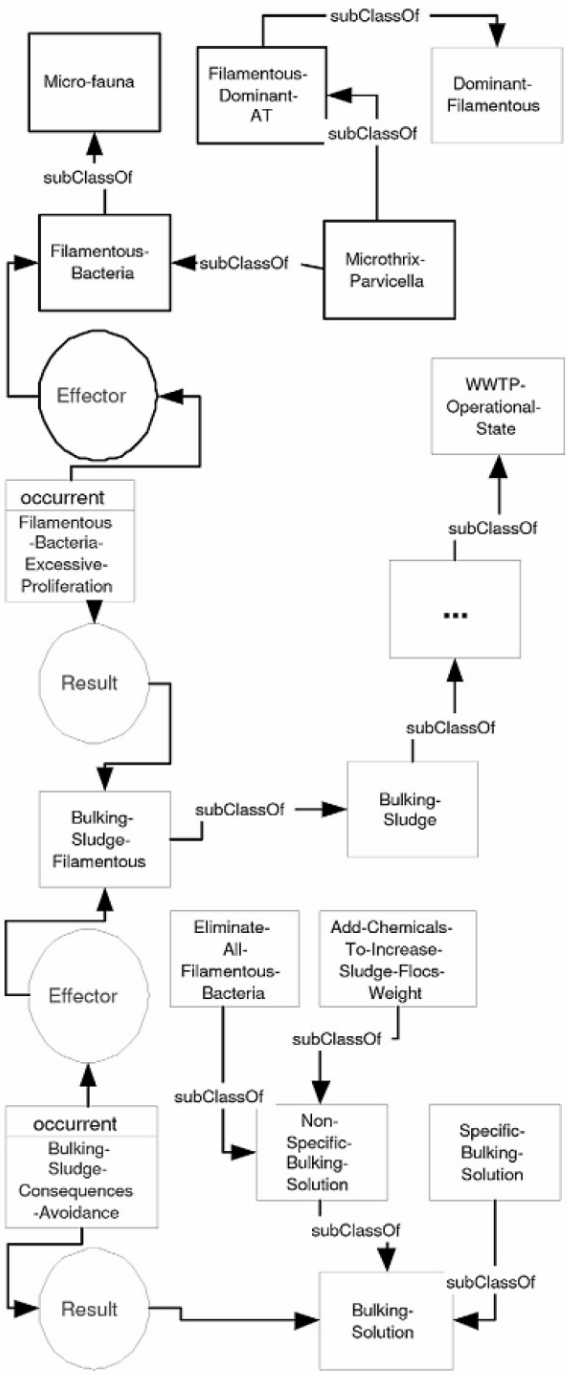


Рис. 7.5
Фрагмент онтологии WaWO

Система мониторинга и лечения больных с сердечной недостаточностью[1]

Рассмотрим использование онтологий и других средств Semantic Web в медицинской экспертной системе мониторинга и лечения больных с сердечной недостаточностью [Prcela, 2008], которая разрабатывалась в рамках европейского проекта HEARTFAID (http://lis.irb.hr/heartfaid/).

Основным компонентом этой системы является база знаний, которая включает дескриптивные, процедурные и фактуальные знания. Дескриптивные или концептуальные знания в системе представляются онтологией сердечной недостаточности (Heart Failure Ontology), описанной на языке OWL средствами редактора Protégé. Эта онтология включает около 200 классов, 100 свойств и более 2000 индивидов (экземпляров). Для стандартизации и согласования с мировым медицинским знанием там, где возможно, понятия снабжаются идентификатором CUI (Concept Unique Identifier) из системы UMLS® (Unified Medical Language System, http://www.nlm.nih.gov/research/umls/), которая на данный момент интегрирует около 100 тезаурусов по медицине и биологии и рассматривается в качестве главного справочника медицинских терминов.

Процедурные знания в системе представлены десятью группами правил вида ***IF*** <Условие X — истинно> ***THEN*** <Выполни действие У>, написанных на языке SWRL (http://www.w3.org/Submission/SWRL/).

При разработке правил использовались понятия, систематизированные в Heart Failure Ontology. Это означает, что в условиях и заключениях правил могут стоять именованные OWL-классы и свойства из онтологии сердечной недостаточности. Это является достоинством данного подхода, так как эксперты могут описывать правила непосредственно в терминах своей предметной области. В то же время в условиях правил разрешается использовать только простейшие операции над данными, например, выполнять простое сравнение значений двух атрибутов. Использование каких-то более сложных вычислительных функций не допускается, что значительно снижает их мощность.

К фактуальным знаниям, относятся данные о конкретных пациентах и ситуациях. Как правило, они представляются индивидами онтологии.

Для верификации онтологии, а также для вывода на онтологии и правилах используются стандартные машины вывода.

В проекте HEARTFAID онтология является центральным компонентом системы вывода в медицинских ЭС и СППР. Примерный сценарий использования онтологии показан на рисунке 7.6. Событие (1), возникающее в системе, через специальный интерфейс поступает в управляющий модуль (2). Этот модуль инициирует (3) извлечение фактуальных знаний из базы данных (4, 5). Затем релевантные данные о пациенте трансформируются в формат онтологии (6) и подготавливаются для процесса вывода (7) в виде множества фактов. Машина вывода отрабатывает, результаты ее работы снова загружаются в онтологию (8), которая затем анализируется интерпретатором онтологии (9). Информация, полученная от интерпретатора, поступает через интерфейс (10) обратно к пользователю системы (11).

[1] Фрагмент из [Загорулько Ю. А., Загорулько Г. Б., 2011].

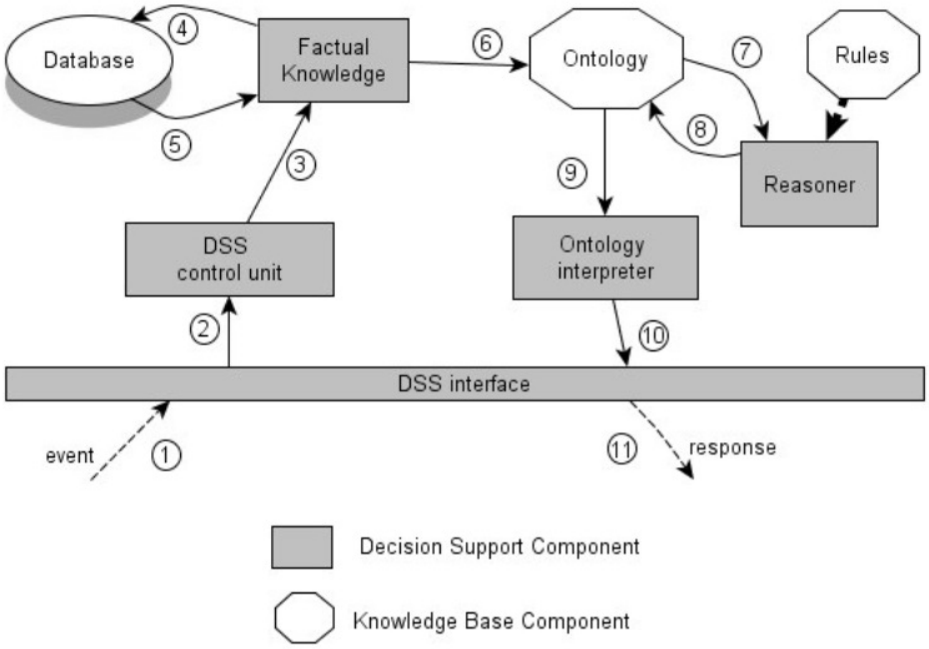


Рис. 7.6

Схема использования онтологии в проекте HEARTFAID

Система поддержки принятия решений по повышению энергоэффективности и экологической безопасности на нефтегазодобывающем предприятии[1]

Рассмотрим СППР [Загорулько и др., 2010], которая была разработана в рамках создания системы оперативного мониторинга технологической инфраструктуры (СОМТИ) нефтегазодобывающего предприятия (НГДП) и предназначена для решения следующих задач:

- Анализ состояния объектов технологической инфраструктуры предприятия с целью выработки рекомендаций по улучшению показателей их работы и предотвращению аварийных ситуаций.
- Анализ статической и динамической информации об объектах с целью выработки рекомендаций для ЛПР для проведения планового технического обслуживания и/или экстренного ремонта объектов, а также по списанию объектов и замене их новыми.
- Выработка рекомендаций для ЛПР об оптимизации потоков технологического транспорта и процессов технического обслуживания и ремонтов объектов.

СППР при выработке решений и рекомендаций для ЛПР использует информацию из центрального хранилища данных СОМТИ (ЦХД СОМТИ), в котором представлены данные о структуре и состоянии технологической инфраструктуры предприятия. В связи с этим СППР реализуется в виде двух взаимодействующих модулей — адаптера СППР, обеспечивающего получение задач от подсистемы

[1] Фрагмент из [Загорулько Ю. А., Загорулько Г. Б., 2011].

управления (ПУ) СОМТИ и обмен данных с ЦХД СОМТИ, и супервизора, организующего работу решателей, обеспечивающих в СППР решение определенных классов задач.

Включением в СППР онтологии в качестве полноправного компонента обеспечивается настройка системы на предметную область и типы решаемых задач. В связи с этим назначением онтология состоит из двух частей — онтологии предметной области и онтологии задач, для описания которых используется язык представления знаний системы Semp-ТАО [Загорулько, Попов, 1996].

Базовыми понятиями (классами) онтологии предметной области являются (рис. 7.7): Объект, Нормативно-справочный объект (НС-объект), Состояние и Результат.

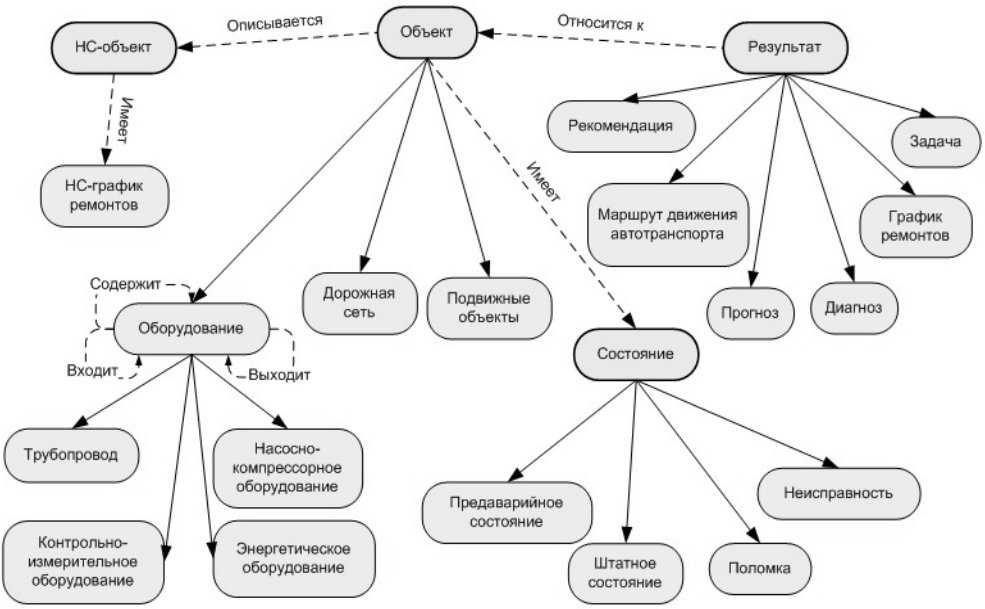


Рис. 7.7

Фрагмент онтологии ПО

В классе Объект выделяются подклассы Оборудование, Подвижные объекты и Дорожная сеть. Объекты могут находиться в том или ином состоянии, для описания которого вводится класс Состояние. В свою очередь класс Состояние имеет следующие подклассы: Неисправность, Поломка, Штатное состояние, Предаварийное состояние и др.

Для контроля соответствия параметров объектов анализа нормативным значениям используется нормативно-справочная информация: для каждого понятия, представляющего некоторый вид оборудования или контейнера оборудования, заводится специальное понятие — НС-объект.

Результатами работы СППР являются Диагнозы состояния оборудования и подвижных объектов, Рекомендации для ЛПР, Прогнозы изменения состояния объектов, решения транспортных задач (Маршруты движения автотранспорта), скорректированные Графики ремонтов и т. п. Отдельным результатом может

быть порожденная исполняемым модулем поддержки принятия решений Задача, в которой, например, более детально анализируется тот или иной объект.

Важное место в онтологии предметной области занимает описание топологии технологической инфраструктуры предприятия. Взаимное расположение и связность объектов технологической инфраструктуры задается отношениями «Содержит», «Входит», «Выходит» и др.

Онтология предметной области, с одной стороны, выступает в качестве высокоуровневого интерфейса к центральному хранилищу данных, обеспечивая доступ к данным ЦХД в виде объектов предметной области, с другой стороны, она определяет формат представления данных в самой СППР в виде тех же объектов предметной области (экземпляров понятий онтологии) и отношений между ними, что позволяет упростить и унифицировать обмен информацией между разнородными компонентами и модулями (решателями) СППР.

Онтология задач обеспечивает настройку системы на типы решаемых задач, тем самым определяя ее функциональность. Она включает описания решаемых системой задач и модулей поддержки принятия решений, реализующих их решения.

Для описания задачи вводится понятие (класс) Задача, имеющее атрибуты «Имя задачи» и «Параметры задачи».

Последний атрибут представляет собой множество пар вида *<Csd, Lo>*, где *Csd* — класс онтологии предметной области, *Lo* — список имен объектов класса *Csd*, для которых должна быть решена данная задача. Если *Lo* — не задано (пусто), то считается, что задача решается для всех объектов класса *Csd*.

На задачах определены отношения «Подзадача» и «Порождает». Первое отношение связывает некоторую задачу с другими задачами (ее подзадачами), решение которых требуется для решения данной задачи. Отношение «Порождает» определяет потенциальную возможность порождения одной задачи другой. В онтологии задач также вводится отношение «Реализует», связывающее модуль принятий решений с задачей, решение которой он обеспечивает.

Реализации модулей принятия решений хранятся в специальном репозитарии.

Во время функционирования СППР, когда возникает потребность в решении какой-либо задачи, планировщик при построении плана ее решения обращается к онтологии, где представлены связи между задачами и модулями принятия решения. При этом он выбирает из репозитория необходимые модули принятия решений и последовательно передает их на исполнение соответствующим решателям. Результаты решения задачи сообщаются пользователю и выкладываются во внешнее хранилище данных (ЦХД) в формате, определенном онтологией предметной области.

В настоящий момент СППР решает 2 типа задач: оптимизационные и аналитические. Оптимизационные задачи реализуются средствами решателя задач целочисленного линейного программирования GLPK[1]. Для решения аналитических задач используются модули поддержки принятия решений, включающие продукционные правила, описанные в терминах онтологии системы. Для спецификации продукционных правил используется язык представления знаний системы Semp-TAO [Загорулько, Попов, 1996], а для их исполнения — интерпретатор этой системы.

---

[1] Glpk for Windows. Glpk: GNU Linear Programming Kit / Version 4.34. URL: http://gnuwin32.sourceforge.net/packages/glpk.htm (дата обращения: 30.04.2010).

7.4. СРЕДСТВА ПОСТРОЕНИЯ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ НА ОСНОВЕ ОНТОЛОГИЙ[1]

Средства создания систем, основанных на знаниях, можно разделить на языки и инструменты. К первым относятся языки представления онтологий, правил, а также языки логического программирования. Инструменты в свою очередь представляют собой среды для построения онтологий, а также системы обеспечивающие взаимодействие онтологий и правил (рис. 7.8).

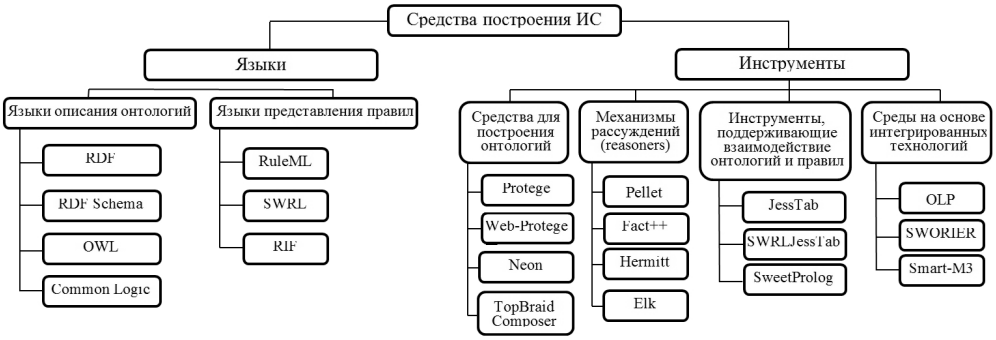


Рис. 7.8

Обзор средств для создания интеллектуальных систем на основе онтологий

Ниже перечислены наиболее яркие примеры подобных средств.

Языки

Существует множество языков описания онтологий и языков представления правил. К первым относятся такие как: RDF, RDF Schema, OWL, Common Logic и другие. Обзор языков представления онтологий см. в разделе 5.7 «Представление онтологий и языки семантического веба», поэтому в данном разделе основное внимание будет уделено языкам представления правил (подробнее про правила см. раздел 4.2 «Продукционные системы»), в особенности тем языкам, которые «согласованы» с языками представления онтологий.

С целью создания стандарта языка разметки для описания правил, позволяющего предусмотреть универсальный подход к формированию правил, пригодных для ввода в различные промышленные системы, был разработан *RuleML (Rule Markup Language)*. С помощью данного языка можно публиковать и обмениваться правилами, созданными в рамках разных систем и задач. RuleML поддерживает как прямой, так и обратный логический вывод, а также использует XML-синтаксис для представления знаний в виде правил. Базовая технология RuleML включает в себя представление и сериализацию синтаксиса, конфигурируемые подмножества языков, преобразования, теоретико-модельную семантику, и процессор правил(rule engine) [RuleML, 2015].

• *Semantic Web Rule Language (SWRL)* [Horrocks et al, 2004] — это язык, который основан на объединении OWL и RuleML, т. е. объединены онтологии (OWL-DL) и правила. Выражение в SWRL, т. е. SWRL-правило, состоит из головы и тела: голова — это консеквент (следствие) продукционного правила и может состоять только из одного атома (предиката), а тело — это антецедент (причина), который может состоять из нескольких атомов. Это означает, что всякий

[1] Раздел подготовлен Ульяной Дудко, редактирование — Дмитрий Кудрявцев.

раз, когда условие, указанное в антецеденте, соблюдается, то условия, указанные в консеквенте, также должны быть соблюдены. В целом, SWRL — это язык, который помогает описать абстрактный механизм оперирования объектами предметной области, а также закономерности предметной области. Основным достоинством SWRL является то, что он дает возможность выводить новые факты из существующих утверждений. Также плюсом языка SWRL является то, что его правила можно описывать с помощью таких языков, как CLIPS (что реализовано в Jess) или Prolog. Некоторые практические аспекты языка SWRL представлены в [Воскобойникова, 2009]. Преимущества SWRL: правила SWRL не содержат конкретных объектов, а только ссылаются на них, что дает возможность применять одно и то же правило к ряду групп объектов; правила SWRL могут быть добавлены к OWL-описанию, т. е. включены в онтологию; написание и «чтение» правил удобнее, если для этого существует специальный язык.

• *RIF* (*Rule Interchange Format*) [Kifer, 2008; Kifer, Boley, 2010] был разработан в 2005 как стандарт для объединения систем правил для обмена знаниями, в частности между сетевыми процессорами правил (rule engine). Цель RIF — это прежде всего обмен правилами, а не попытка разработать один универсальный язык. В рамках RIF было разработано семейство языков, так называемые «диалекты», со строго определенным синтаксисом и семантикой, которые являются единообразными и расширяемыми. Единообразие означает совместное использование существующего синтаксического и семантического аппарата. Расширяемость предоставляет возможность эксперту определять новый RIF-диалект как синтаксическое расширение существующего, с новыми элементами, соответствующими желаемому увеличению функционала. Основной замысел создания обмена знаниями с помощью RIF заключается в том, что различные системы имеют возможность осуществлять синтаксическое преобразование своего языка в RIF диалекты и обратно. Эти преобразования должны сохранять семантику. Это позволит множествам правил взаимодействовать друг с другом от одной системы к другой при условии, что эти системы должны общаться между собой на том диалекте, который они обе поддерживают.

Инструменты

Среды для построения онтологий

Существует большой набор сред для построения онтологий: Protégé, WebProtégé, NeOn, TopBraid Composer, OntoWiki, Knoodl. Рассмотрим некоторые из них.

• *Protégé (http://protege.stanford.edu/)* — это свободный, открытый редактор онтологий и фреймворк для построения баз знаний. Платформа Protégé поддерживает два основных способа моделирования онтологий посредством редакторов Protégé-Frames и Protégé-OWL. Онтологии, построенные в Protégé, могут быть экспортированы во множество форматов RDF (RDFS), OWL и XML Schema. Архитектура Protege за счет использования плагинов может быть адаптирована для создания как простых, так и сложных приложений на основе онтологий. Разработчики имеют возможность интегрировать результат Protégé с системами правил или другими системами решения задач для построения широкого спектра интеллектуальных систем. Преимуществом Protégé является его открытая архитектура, что делает возможным разработку большого количества сторонних компонентов, например, для решения задач визуализации, вывода и поиска.

• *Web-Protege* (http://protegewiki.stanford.edu/wiki/WebProtege) — это удобный и легкий редактор онтологий для Web с поддержкой одновременной совместной работы. Web-Protege предоставляет следующие возможности, как поддержка редактирования OWL2, OBO онтологий, поддержка различных форматов загрузки и выгрузки онтологий, простой настраиваемый интерфейс редактирования, который обеспечивает доступ к наиболее часто используемым OWL-конструкциям, полное отслеживание истории изменения, а также различные средства совместной разработки онтологий.

• *NeOn Toolkit* (http://neon-toolkit.org/wiki/Main_Page.html) — построенная на базе Eclipse кроссплатформенная модульная система, предназначенная для поддержки полного жизненного цикла разработки онтологии. Для системы существует значительное количество (более 45) модулей, расширяющих ее функциональность и направленных на поддержку различных этапов разработки онтологии: аннотации, взаимодействие онтологии и человека, приобретение знаний, оценка онтологии, рассуждение и вывод, повторное использование и т. д. [Suárez-Figueroa et al, 2012].

Механизм рассуждений / Машина логического вывода (Reasoner)

Механизм рассуждений (https://www.w3.org/2001/sw/wiki/Category:Reasoner) — это часть программного обеспечения, способная выводить логические следствия из набора заявленных фактов или аксиом. Примерами сервисов механизмов рассуждений являются: тестирования иерархии классов, проверка согласованности. Среди наиболее распространенных можно выделить Pellet, Fact++, Hermit, RacerPro, Elk-reasoner. Рассмотрим некоторые из них.

• *Pellet* (https://www.w3.org/2001/sw/wiki/Pellet) [Sirin et al, 2007] — это общедоступный механизм рассуждения на базе Java, который поддерживает вывод на уровне OWL-DL. Он представляет собой удобное средство для реализации полного OWL-DL вывода. Pellet обеспечивает такие функции вывода, как проверка на непротиворечивость, выполнимость, классификация и реализация понятий. Pellet включает также оптимизированный механизм запросов, способный отвечать на запросы Abox.

• *Hermit* (https://www.w3.org/2001/sw/wiki/Hermit) [Glimm et al, 2014] позволяет проверять правильность составления онтологий в OWL файлах и составлять иерархическую структуру классов. В Hermit вычисления при помощи гипертабличного исчисления ускоряют процесс классификации онтологий. Рассуждение и логический вывод могут быть также произведены с помощью оболочки для разработки экспертных систем Jess. Механизм рассуждения *Hermit* также является общедоступным. Он написан при помощи OWL API.

• *ELK* (https://code.google.com/p/elk-reasoner/) [Kazakov, Krötzsch, 2012; Kazakov, Klinov, 2015; Поспешный, Стиренко, 2012]. ELK является свободно распространяемым высоко производительным логическим анализатором с открытым исходным кодом для онтологий OWL 2 EL профиля. Высокая производительность логического анализа достигается за счет применения оптимизированного параллельного накопительного алгоритма для рассуждений, работа которого, в общих чертах, заключается в итеративном применении к множеству исходных аксиом трансформирующих правил, порождающих новые аксиомы-следствия. EKL обладает гибкой модульной архитектурой и может быть использован в разнообразных конфигурациях, в частности возможно использование ELK как утилиты, вызываемой из командной строки операционной системы, как библиотеки, интегрируемой в приложение посредством интерфейса OWL API или как подключаемого расширения к редактору онтологий и баз знаний Protege.

Средства, поддерживающие взаимодействие онтологий и правил

• *JessTab* (http://www.jessrules.com/jesswiki/view?JessTab) является мостом между Protégé и Jess. Jess является аббревиатурой названия Java Expert System Shell. Jess представляет собой механизм правил и среду сценариев, написанные целиком на языке Java. JessTab позволяет программам Jess использовать знания, создаваемые в Protégé. Это достигается за счет отображения баз знаний Protégé в Jess-утверждения [Sensoy et al, 2012].

• *SWRL JESS Tab* (http://protege.cim3.net/cgi-bin/wiki.pl?SWRLJessTab). Выполнение правил SWRL требует наличия машины обработки правил, которая осуществляет вывод, используя в качестве входной информации множество правил и фактов. Любые новые выводимые факты используются как входные данные для того, чтобы активировать по возможности больше правил. В большинстве реализаций используется машина обработки правил Jess для того, чтобы сделать возможным SWRL вывод на множестве данных Protégé. SWRL-правила хранятся как OWL-экземпляры вместе с ассоциируемой с ними базой знаний. Прежде всего, SWRL-правила переводятся в Jess-правила, используя SWRL JessBridge, который встроен в Protégé и добавлен в машину обработки правил Jess; затем онтологии и база знаний транслируются в Jess-факты, а также вводятся в машину вывода; в-третьих, машина обработки правил Jess делает возможным вывод и производит результаты в формате Jess и, наконец, эти результаты переводятся обратно в формат OWL. SWRL-правила можно легко моделировать, используя механизм SWRLJessTab, который включен в Protégé. Механизм SWRLJessTab позволяет вводить, удалять и редактировать SWRL-правила [Борисов, 2014].

• *SweetProlog* [Laera et al, 2004] — это система, основанная на Java, для перевода OWL-онтологий и правил, записанных в RuleML в множество фактов и правил языка Prolog, после чего весь механизм рассуждения об этих фактах и правилах производится в Prolog. Этот подход использует механизм JIProlog в качестве процессора правил (rule engine), он переводит подмножество OWL в простые предикаты языка Prolog, которые JIProlog способен обрабатывать [Sensoy et al, 2012].

Среды на основе интегрированных технологий

К средам, сочетающим в себе онтологии и логическое программирование, можно отнести новый инструмент *OLP (Ontological Logic Programming)*[1], который сочетает в себе преимущества рассуждения на основе онтологий (ontological reasoning) и логическое программирование без каких-либо ограничений в выразительности. Программа OLP может динамически импортировать онтологии и использовать термины (классы, свойства, индивиды) напрямую в программе OLP. Термины интерпретируются при помощи механизма рассуждений (ontology reasoner) во время интерпретации программы OLP. Объединение двух технологий обеспечивает не только расширение выразительных возможностей, но и удобство программирования [Sensoy et al, 2012].

Система *SWORIER*, используя механизм рассуждений на основе онтологий и правил, служит для того, чтобы отвечать на запросы пользователя. Преимуществом SWORIER является то, что она позволяет пользователю вносить изменения динамически, а также добавлять новые факты и удалять старые в оперативном режиме. Она переводит OWL-DL онтологии с SWRL-правилами в Prolog, а затем, на основе этого перевода производится ответ на запрос средствами Prolog [Samuel et al, 2008].

[1] Исходный код OLP доступен по адресу: http://olp-api.sourceforge.net

Платформа *Smart-M3* позволяет создать инфраструктуру, на основе которой распределенные интеллектуальные системы строят свои интеллектуальные пространства. Платформа делится на: 1) инфраструктуру и 2) инструменты разработки. Инфраструктура реализует семантических информационных брокеров, которые предоставляют доступ агентам к информационному содержимому интеллектуального пространства (реализованному в виде RDF-графа) посредством семантических запросов на основе языка SPARQL. Инструменты предназначены для разработчиков программного кода агентов на основе специальных библиотек KPI (англ. Knowledge Processor Interface) [Кашевник и др., 2013].

Пример архитектуры финансовой системы, основанной на онтологиях

В работе [Shue et al, 2009] представлена система формирования финансового рейтинга, основанная на онтологии. Архитектура этой системы (рис. 7.9) иллюстрирует использование рассмотренных выше средств создания интеллектуальных систем на основе онтологий.

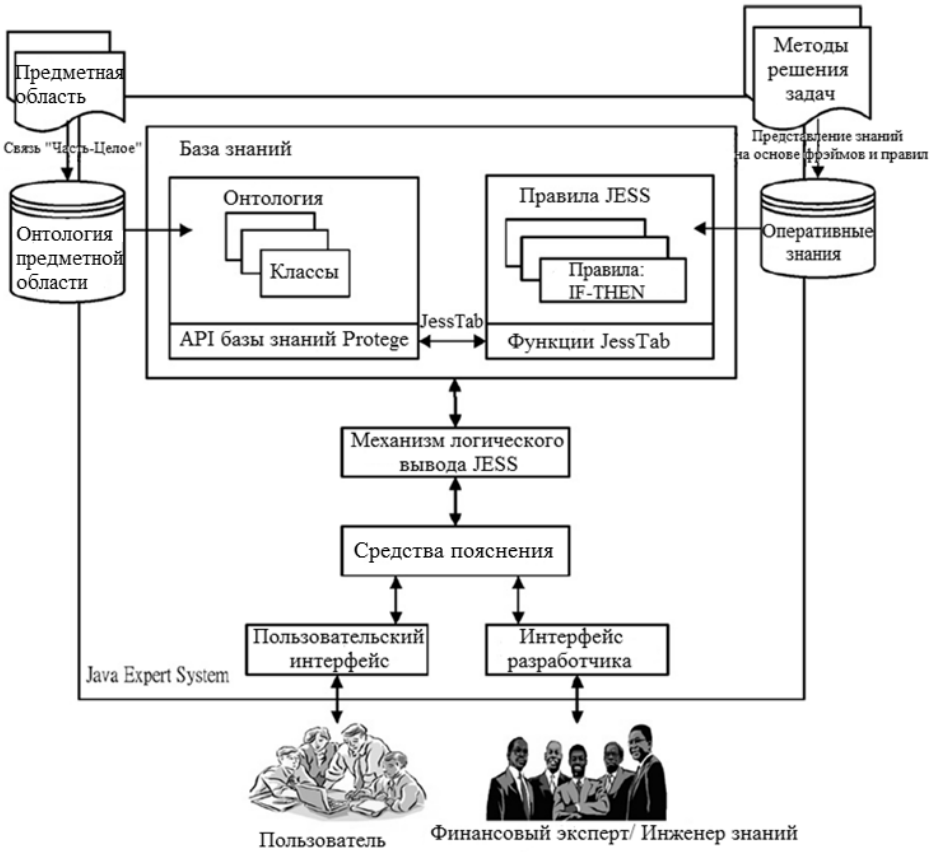


Рис. 7.9

Пример архитектуры интеллектуальной системы на основе Protégé и Jess [Shue et al., 2009]

Эта архитектура состоит из модуля базы знаний, механизма логического вывода, средства пояснения (explanation facility), пользовательского интерфейса и интерфейса разработчика. *Модуль базы знаний* — это ядро системы; оно состоит из предметных знаний (domain knowledge) о финансовой отчетности и оперативных знаний, содержащих правила принятия решений для финансовых аналитиков. *Пользовательский интерфейс* принимает реальные факты из финансовой отчетности и вызывает механизм логического вывода, чтобы активировать правила принятия решений в базе оперативных знаний. Правила принятия решения затем обеспечивают доступ к соответствующим данным домена знаний, которые возможно в перспективе активируют новые правила. Окончательный вывод будет также представлен через пользовательский интерфейс вместе с соответствующими объяснениями из *модуля обоснования*. *Интерфейс разработчика* позволяет разработчикам системы изменять ее содержимое. Вся база построена на языке Java, что позволяет обеспечить интеграцию между модулями во время выполнения.

В процессе разработки базы знаний была использована платформа Protégé для построения онтологии предметной области (domain knowledge), а также механизм правил Jess для определения оперативных знаний. Protégé основан на Java и имеет открытую архитектуру, что позволяет связываться с другими системами через подключаемые модули. В случае с Jess был использован плагин JessTab, который представляет собой своеобразный мост для связи Protégé и Jess. API из Protégé способен работать с функциями JessTab. Это дает возможность механизму логического вывода Jess функционировать как механизм комбинированной экспертной системы. Во время работы функция JessTab имеет право управлять базой знаний Protégé и позволяет правилам в Jess получить доступ к содержимому онтологии предметной области. Эта способность повышает уровень сложности рассуждений платформы Protege по сравнению с первоначальным.

ЗАКЛЮЧЕНИЕ

Знания, как известно, это сила. Эта фраза Фрэнсиса Бэкона давно стала крылатой, но только в XXI веке она стала осязаемой в условиях экономики знаний. Феномен превращения знаний в главную производительную силу был предсказан еще в 60-70-е годы XX века в работах П. Друкера, Д. Белла, О. Тоффлера. Именно поэтому сейчас так востребованы технологии работы со знаниями, описанные в данной книге. Системный взгляд на управление знаниями должен учитывать достижения всех наук, изучающих интеллектуальные ресурсы человека, предприятия или отрасли — от психологии и лингвистики до организационного поведения и информационных технологий. Это особенно важно сейчас, когда управление знаниями из модного лозунга превращается в практику современного просвещенного менеджмента [Мильнер, 2003; Катькало, 2008]. Феномен корпоративных знаний — реальность информационного мира, где бизнес держится на знаниях, ценность которых часто превосходит материальные активы компании.

И в России, и на Западе последние годы значительно усилился интерес пользователей к использованию интеллектуальных технологий в практических приложениях. Особенно заметен этот интерес при построении корпоративных информационных систем (КИС), в которых реализуются новейшие достижения технологий искусственного интеллекта. Интеллектуализацию информационных технологий можно рассматривать как общую тенденцию в их развитии. Невозможно управлять знаниями, не понимая их природы, особенностей, способов представления и обработки. В данной книге рассмотрен весь спектр моделей и методов работы со знаниями, из которых квалифицированный аналитик выбирает подходящие к конкретной задаче.

Особое внимание авторы уделили средствам представления знаний и современным методам онтологического моделирования и управления знаниями на предприятии. В современных КИС органически применяются описанные технологии извлечения, структурирования и формализации знаний. Также в бизнесе актуальны системы, поддерживающие многоаспектный анализ данных, принятие решений в сложных ситуациях, управление корпоративными знаниями, экспертные системы и системы интеллектуального поиска информации.

Между этими приложениями и средой КИС необходимо иметь программные средства промежуточного слоя (middleware), обеспечивающие функционирование базы знаний, логический вывод на базе знаний, естественно-языковый пользовательский интерфейс для формирования запросов к БЗ и взаимодействие с внешней средой. Эти вопросы вышли за рамки рассмотрения.

Авторы надеются, что эта книга будет полезна не только студентам и преподавателям, но также специалистам и менеджерам, и расширит представления об инженерии и управлении знаниями.

ЛИТЕРАТУРА И ИНТЕРНЕТ-РЕСУРСЫ

1. Абдикеев Н. М., Киселев А. Д. Управление знаниями корпорации и реинжиниринг бизнеса: Учебник. — М. : ИНФРА-М, 2011. — С. 371–380.
2. Агроскин В., Левенчук А., Головков В. Онтология для инженерных данных // Открытые системы (06). — 2013.
3. Адаменко А., Кучуков А. Логическое программирование и Visual Prolog. — СПб. : БХВ-Петербург, 2003.
4. Адельсон-Вельский Г. М., Арлазаров В. Л., Битман А. Р., Донской М. В. Машина играет в шахматы. — М. : Наука, 1983.
5. Алахвердов В. М. Когнитивные стили в контурах процесса познания. Когнитивные стили / Под ред. В. Колги. — Таллинн, 1986. — С. 17–23.
6. Александров Е. А. Основы теории эвристических решений. — М. : Советское радио, 1975.
7. Алексеевская М. А., Недоступ А. В. Диагностические игры в медицинских задачах. Вопросы кибернетики. Задачи медицинской диагностики и прогнозирования с точки зрения врача, (112). — 1988. С. 128–139.
8. Алпатов А. М. Циркодианные ритмы человека и режим труда-отдыха: гипотеза «сжатой пружины» // Известия РАН. Сер. биол. (6). — 1993. — С. 874–881.
9. Амамия М., Танака Ю. Архитектура ЭВМ и ИИ. — М. : Мир, 1993.
10. Андреас С., Герлинг К., Фолкнер Ч. и др. Миссия НЛП: новейшие американские психотехнологии. — М. : Институт общегуманитарных исследований, 2000.
11. Анохин П. К. Избранные труды. Системные механизмы высшей нервной деятельности. — М. : Наука, 1979.
12. Аншаков О. М., Скворцов Д. П., Финн Д. К. Логические средства экспертных систем типа ДСМ. Семиотика и информатика (28). — 1986. — С. 5–15.
13. Аткинсон Р. Человеческая память и процесс обучения. Пер. с англ. — М. : Прогресс, 1989.
14. Бабкин Э. А., Князькин В. П., Шиткова М. С. Сравнительный анализ языковых средств, применяемых в методологии бизнес моделирования. Бизнес-информатика (2). — 2011. — С. 31–42.
15. Бабкин Э. А., Князькин В. П., Шиткова М. С. Разработка метода проведения сравнительного анализа языков бизнес-моделирования. Бизнес-информатика (3). — 2010. — С. 41–46.
16. Бахтин М. М. Вопросы литературы и эстетики: Исследования разных лет. — М. : Худож. лит, 1975.
17. Бездушный А. А., Бездушный А. Н., Нестеренко А. К., Серебряков В. А., Сысоев Т. М. Предложения по наборам метаданных для научных информационных ресурсов ЕНИП РАН. Электронные библиотеки 7(5). — 2004. — С. 1–23.
18. Белнап Н., Стил Т. Логика вопросов и ответов. — М. : Прогресс, 1981.
19. Берн Э. Игры, в которые играют люди. Люди, которые играют в игры / Пер. с англ. — М. : Прогресс, 1992.
20. Борисов А. Н. Построение интеллектуальных систем, основанных на знаниях, с повторным использованием компонентов. — Латвия : Рижский технический университет, 2014. — С. 97–102.
21. Борисова Н. В., Соловьева А. А. и др. Деловая игра «Методика конструирования деловой игры». — М. : ИПКИР, 1988.
22. Ботвинник М. М. Шахматный метод решения переборных задач. — М. : Сов. радио, 1983.
23. Брагина Н. Н, Доброхотова Т. А. Функциональные асимметрии человека. — М. : Медицина, 1988.
24. Брейс А. Анкетирование . Баланс Бизнес Букс. — 2005.
25. Брукс Ф. Мифический человеко-месяц или как создаются программные системы. — СПб. : Символ-Плюс, 2000.
26. Брукс Ф. Проектирование процесса проектирования: записки компьютерного эксперта. — М. : Вильямс, 2012.
27. Бурков В.И. и др. Деловые игры в принятии управленческих решений. — М. : МИСИС, 1980.
28. Буч Г. Объектно-ориентированное проектирование с примерами применения. — М. : Конкорд, 1992.
29. Буч Г. и др. Объектно-ориентированный анализ и проектирование с примерами приложений на C++. — СПб. : Невский диалект, 1998.
30. Буч Г., Рамбо Дж., Якобсон А. UML. Классика CS. — СПб. : Питер, 2006.
31. Бьюзен Т., Бьюзен Б. Супермышление. — Минск : Белорусский дом печати, 2002.

32. Валькман Ю. Р., Исмагилова Л. Р. О языке образного мышления // Труды Международного семинара «Диалог», «Компьютерная лингвистика и интеллектуальные технологии». — М., 2004.
33. Величковский Б. М., Капица М. С. Психологические проблемы изучения интеллекта. Интеллектуальные процессы и их моделирование. — М. : Наука, 1987. — С. 120–141.
34. Вертгеймер М. Продуктивное мышление / Пер. с нем. — М. : Прогресс, 1987.
35. Волков А. М., Ломнев В. С. Классификация способов извлечения опыта экспертов // Известия АН СССР, Техн. кибернетика (5). — 1989. — С. 34–45.
36. Воскобойникова А. SWRL — как, зачем? — [Электронный ресурс]. — URL: http://shcherbak.net/2009/04/swrl/ (Дата обращения: 24.08.2015).
37. Гаврилова Т. А. Представление знаний в экспертной диагностической системе АВТАНТЕСТ // Изв. АН СССР. Техн. Кибернетика. 5. — 1984. — С. 165–173.
38. Гаврилова Т. А. Подготовка коллектива разработчиков экспертной системы // Труды школы-семинара «Проблемы применения ЭС в народном хозяйстве». — 1989. — С. 59–62.
39. Гаврилова Т. А. Объектно-структурная методология концептуального анализа знаний и технология автоматизированного проектирования баз знаний // Труды Междунар. конф. «Знания — Диалог — Решение 95», (1). — 1995. — С. 1–9.
40. Гаврилова Т. А. Управление знаниями: ЧТО ДЕЛАТЬ? // Сб. докладов Седьмой научно-практической конференции «Реинжиниринг бизнес-процессов на основе современных информационных технологий. Системы управления знаниями» (РБП-СУЗ-2004). — М., 2004. — С. 61–67.
41. Гаврилова Т. А., Горовой В. А., Болотникова Е. С. Оценка когнитивной эргономичности онтологии на основе анализа графа. Искусственный интеллект и принятие решений (3). — 2009. — С. 33–41.
42. Гаврилова Т. А., Кудрявцев Д. В., Лещева И. А., Павлов Я. Ю. Об одном методе классификации визуальных моделей // Бизнес-информатика 26(4). — 2013. — С. 21–34.
43. Гаврилова Т. А., Хорошевский В. Ф. 2000. Базы знаний интеллектуальных систем. Учебник для вузов. — СПб. : Питер.
44. Гаврилова Т., Гулякина Н. 2008. Визуальные методы работы со знаниями: попытка обзора. Журнал «ИИ и принятие решений», (N1): 15–22.
45. Гаврилова Т. А. 2005. Об одном подходе к онтологическому инжинирингу. Новости искусственного интеллекта, (3): 25–31.
46. Гаврилова Т. А., Воинов А. В. Когнитивный подход к созданию онтологии. Научно-техническая информация. Серия 2: Информационные процессы и системы, (3). — 2007. — С. 19–23.
47. Гаврилова Т. А., Кудрявцев Д. В. Формирование корпоративной памяти на основе фасетной классификации // Сб. Научных трудов 9-й Российской научно-практической конференции «Реинжиниринг бизнес-процессов на основе современных информационных технологий. Системы управления знаниями» (РБП-СУЗ-2006). — М., 2006. — С. 364–377.
48. Гаврилова Т. А., Лещева И. А., Страхович Э. В. Об использовании визуальных концептуальных моделей в преподавании // Вестник СПбГУ, серия Менеджмент, (4). — 2011. — С. 125–151.
49. Гаврилова Т. А., Муромцев Д. И. Интеллектуальные технологии в менеджменте. — СПб. : Изд. СПбГУ, 2008.
50. Гаврилова Т. А., Червинская К. Р. Извлечение и структурирование знаний для экспертных систем. — М. : Радио и связь, 1992.
51. Гаек П., Гавранек Т. Автоматическое образование гипотез. — М. : Наука, 1984.
52. Галушкин А. Нейрокомпьютеры. — М. : ИПРЖР, 2000.
53. Гельфанд И. И., Розенфельд Б. И., Шифрин М. А. Структурная организация данных в задачах медицинской диагностики и прогнозирования // Вопросы кибернетики. Задачи медицинской диагностики и прогнозирования с точки зрения врача. — М. : АН СССР, 1988. — С. 5–64.
54. Гиляревский Р. С., Шапкин А. В., Белоозеров В. Н. Рубрикатор как инструмент информационной навигации. — СПб. : Профессия, 2008.
55. Глаголева Е. А. Трудности усвоения школьных навыков. Педагогика. — 1926.
56. Горбань А. Н., Россиев Д. А. Нейронные сети на персональном компьютере. — Новосибирск : Наука, 1996.
57. Горелов И. Н. Разговор с компьютером. — М. : Наука, 1987.
58. Горовой В. А. Модель классификации методов оценки онтологий // Материалы 2-й международной молодежной конференции «Искусственный интеллект: философия, методология, инновации». Санкт-Петербург. — 2007. — С. 307–310.

59. Городецкий В. И. Многоагентные системы: современное состояние исследований и перспективы применения // Новости искусственного интеллекта (1). — 1996. — С. 44–59.
60. Горшков С. Введение в онтологическое моделирование — ТриниДата, 2016.
61. Гращенко Л. А. и др. Концептуальная модель системы русско-таджикского машинного перевода // Доклады Академии наук Республики Таджикистан 54(4).— 2011. — С. 279–285.
62. Григорьев Л. Ю., Кудрявцев Д. В. Организационное проектирование на основе онтологий: методология и система ОРГ-Мастер // Научно-технические ведомости СПбГПУ. Серия «Информатика. Телекоммуникации. Управление», (1). — 2012. — С. 21–28.
63. Григорьев Л. Ю., Заблоцкий А. А., Кудрявцев Д. В. Технология наполнения баз знаний онтологического типа // Научно-технические ведомости СПбГПУ. Серия «Информатика. Телекоммуникации. Управление», 3(150).— 2012. — С. 27–36.
64. Григорьев Л. Ю., Кознов Д. В., Кудрявцев Д. В. Обзор языка моделирования ОРГ-Мастер. Системное программирование. — СПб. : Изд-во СПбГУ, 2013. — 8(1). — С. 5–34.
65. Григорьев Л. Ю. и др. Менеджмент по нотам: Технология построения эффективных компаний / Под ред. Л. Ю. Григорьева. — М. : Альпина Паблишер, 2010.
66. Гурьянова М. А., Ефименко И. В., Хорошевский В. Ф. Онтологическое моделирование экономики предприятий и отраслей современной России: Часть 2. Мировые исследования и разработки: аналитический обзор: препринт WP7/2011/08 (ч. 2). Нац. исслед. ун-т «Высшая школа экономики». — М. : Изд. дом Высшей школы экономики, 2011.
67. Гусакова С. М., Финн В. К. Сходства и правдоподобный вывод // Известия АН СССР. Техническая кибернетика (5). — 1987. — С. 42–63.
68. Данилин А., Слюсаренко А. Архитектура и стратегия. «Инь» и «янь» информационных технологий. Интернет-университет информационных технологий ИНТУИТ.ру. — 2005.
69. Демьянков В. З., Кубрякова Е. С., Панкрац Ю. Г., Лузина Л. Г. Краткий словарь когнитивных терминов / Под общ. ред. Е. С. Кубряковой. — М. : Филологический факультет МГУ им. М. В. Ломоносова, 1996. — С. 171–174. (Электронная версия на http://infolex.ru).
70. Детмер У. Теория ограничений Голдратта: Системный подход к непрерывному совершенствованию / Пер. с англ. У. В. Саламатова. — М. : Альпина Бизнес Букс, 2006.
71. Джарратано Д., Райли г. Экспертные системы. Принципы разработки и программирование. — М. ; СПб. ; Киев : Вильямс, 2006.
72. Джексон П. Введение в экспертные системы. — М. ; СПб. ; Киев : Вильямс, 2001.
73. Добров Б. В., Иванов В. В., Лукашевич Н. В., Соловьев В. Д. Онтологии и тезаурусы: учебно-методическое пособие. — Казань : Изд-во Казанского ГУ, 2006.
74. Епифанов М. Е. Индуктивное обобщение в ассоциативных сетях // Известия АН СССР. Техническая кибернетика (5). — 1984. — С. 132–146.
75. Ерохин В. А. Композитор и компьютер. Музыка как форма интеллектуальной деятельности. — 2011.
76. Жинкин Н. И. Речь как проводник информации. — М. : Педагогика, 1982.
77. Загоруйко Н. Г. Прикладные методы анализа данных и знаний. — Новосибирск, 1999.
78. Загорулько Ю. А. Методы и методологии разработки, сопровождения и реинжиниринга онтологий // Труды Первого симпозиума «Онтологическое моделирование», Звенигород, 19–20 мая 2008 г. — М. : ИПИ РАН, 2008. — С. 167–201.
79. Загорулько Ю. А. Построение порталов научных знаний на основе онтологий. Вычислительные технологии, т. 12, спецвыпуск 2. — 2007. — С. 169–177.
80. Загорулько Ю. А., Загорулько Г. Б. Использование онтологий в экспертных системах и системах поддержки принятия решений // Труды Второго симпозиума «Онтологическое моделирование», Казань, 11–12 октября 2010 г. — М. : ИПИ РАН, 2011.
81. Загорулько Ю. А., Загорулько Г. Б., Кравченко А. Ю., Сидорова Е. А. Разработка системы поддержки принятия решений для нефтегазодобывающего предприятия // Труды 12-й национальной конференции по искусственному интеллекту с международным участием — КИИ-2010. — М. : Физматлит, 2010. — (3). — С. 137–145.
82. Загорулько Ю. А., Попов И. Г. Представление знаний в интегрированной технологической среде Semp-TAO. Проблемы представления и обработки не полностью определенных знаний. — Москва ; Новосибирск, 1996. — С. 59–74.
83. Зарипов Р. Х. Машинный поиск вариантов при моделировании творческого процесса. — М. : Наука, 1983.
84. Зиндер Е. З. Архитектура предприятия в контексте бизнес-реинжиниринга. Intelligent Enterprise. — 2008. — 1(4). — С. 46; 2(7). — С. 183.
85. Иберла К. Факторный анализ / Пер. с нем. — М. : Статистика, 1980.

86. Иванов П. И. Влияние некоторых индивидуально-психологических особенностей на процесс общения / Дис. канд. психол. наук. — М., 1986.
87. Каган М. С. Мир общения: проблема межсубъектных отношений. — М. : Политиздат, 1988.
88. Каллан Р. Основные концепции нейронных сетей. — М. : Вильямс, 2001.
89. Калянов Г. Н. Архитектура предприятия и инструменты ее моделирования. Автоматизация в промышленности (7). — 2004. — С. 9–12.
90. Кашевник А., Корзун Д. Ж., Баландин С. И. Разработка интеллектуальных систем на базе платформы Smart-M3. — Петрозаводск : Издательство ПетрГУ, 2013.
91. Кобринский Б. А. Ретроспективный анализ медицинских экспертных систем // Новости искусственного интеллекта, (2). — 2005. — С. 6–17.
92. Кобринский Б. А. Системы искусственного интеллекта в медицине: Состояние, проблемы и перспективы // Новости искусств. интеллекта (2). — 1995. — С. 65–79.
93. Когаловский М. Р. Системы доступа к данным, основанные на онтологиях // Программирование (4). — 2012. — С. 55-77.
94. Кознов Д. В. Методика обучения программной инженерии на основе карт памяти // Системное программирование, 3(1). — 2008. — С. 121–140.
95. Комаров В. Ф. Управленческие имитационные игры. — Новосибирск : Наука, 1989.
96. Коршунов А. М., Манталов В. В. Диалектика социального познания. — М. : Политиздат, 1988.
97. Костырко А., Кудрявцев Д., Григорьев Л., Кислова В., Жулин А., Синятуллина Л., Ермаков Р. Моделирование комплексов городского хозяйства для системного развития ИКТ города // Сборник трудов конференции «Инженерия знаний и технологии семантического веба — 2012», 1–9 октября 2012. Санкт-Петербург, Россия. — СПб. : СПбГУ ИТМО, 2012.— С. 81–88.
98. Кудрявцев Д. В. Разработка моделей и методов обработки знаний в области организационного проектирования на основе онтологий / Дисс. канд. техн. наук: 05.13.01; 05.13.11. — СПб. : СПбГПУ, 2009.
99. Кудрявцев Д. В. Системы управления знаниями и применение онтологий: Учеб. пособие. — СПб. : Изд-во Политехн. ун-та, 2010.
100. Кудрявцев Д. В., Арзуманян М. Ю., Григорьев Л. Ю. Технологии бизнес-инжиниринга: учеб. пособие. — СПб. : Изд-во Политехнического университета, 2014.
101. Кудрявцев Д. В., Григорьев Л. Ю., Кислова В. В., Жулин А. Б. Административное моделирование на основе онтологий // Вопросы государственного и муниципального управления (1).— 2009. — С. 157–169.
102. Кулюткин Ю. И., Сухобская Г. С. Индивидуальные различия в мыслительной деятельности взрослых учащихся. — М. : Педагогика, 1971.
103. Леоненков А. Самоучитель UML. — СПб. : БХВ-Петербург, 2001.
104. Лимантов Ф. С. О природе вопроса. Вопрос. Мнение. Человек // Уч. записки ЛГПИ им. Герцена (497). — 1971. — С. 4–20.
105. Лукашевич Н. В. Тезаурусы в задачах информационного поиска. — М. : Издательство Московского университета, 2011.
106. Лунева О. В., Хорошилова Е. А. Психология делового общения. — М. : ВКШ при ЦК ВЛКСМ, 1987.
107. Любич Д. В. 1000 афоризмов. — М. : Буковский, 1998.
108. Люгер Д. Искусственный интеллект. Стратегии и методы решения сложных проблем. — М. : Вильямс, 2003.
109. Мальковский М. Г. Диалог с системой искусственного интеллекта. — М. : МГУ, 1985.
110. Мариничева М. К. Управление знаниями на 100%: Путеводитель для практиков. — М. : Альпина Бизнес Букс, 2008.
111. Маслов С. Ю. Теория дедуктивных систем и ее применения. — М. : Радио и связь, 1986.
112. Месарович М., Такахара Я. Общая теория систем: математические основы. — М. : Мир, 1978.
113. Мильнер Б., Румянцева З., Смирнова В., Блинникова А. Управление знаниями в корпорациях. — М. : Дело, 2006.
114. Мильнер Б. З. Управление знаниями. — М. : Инфра-М, 2003.
115. Михалко М. Рисовый штурм и еще 21 способ мыслить нестандартно. — Манн, Иванов и Фербер, 2015.
116. Мицич П. П. Как проводить деловые беседы. — М. : Экономика, 1987.
117. Моделирование деятельности органов власти, государственных и муниципальных организаций. 2006 // Отчет по научно-исследовательской работе, выполненной компанией

АНО КМЦ «Бизнес-инжиниринг» по заказу Министерства экономического развития и торговли в рамках целевой программы «Административная реформа», шифр темы: 22.5.1.
118. Моргоев В. К. Метод структуризации и извлечения экспертных знаний: имитация консультаций. Человеко-машинные процедуры принятия решений. — М. : ВНИИСИ. — 1988. — С. 44–57.
119. Муромцев Д. И. Оболочка экспертных систем Exsys Corvid. Методическое пособие. — СПб. : СПб ГУ ИТМО, 2005.
120. Наст Дж. Эффект визуализации. — М. : Эксмо, 2008.
121. Научно-методическое обеспечение эффективной интеграции процессов управления комплексами городского хозяйства с использованием средств информационно-коммуникационных технологий: [Текст] Отчет по НИР / рук. А.Б. Жулин. — ИГМУ НИУ ВШЭ, 2011.
122. Нелюбин Л. Компьютерная лингвистика и машинный перевод: методическое пособие. Всес. центр переводов. — 1991.
123. Николов С. А. и др. Анализ состояния и тенденции развития информатики. Проблемы создания экспертных систем. Исследовательский отчет / Под ред. С. А. Николова. — София : Интерпрограмма, 1990.
124. Новак С. Бизнес-инструменты для производственного предприятия: от основ до высшего пилотажа / Пер. с англ. Л. Ю. Смоляк ; науч. ред. О. Ч. Шмигельский. — Минск : Гревцов Паблишер, 2008.
125. Норвиг П., Рассел С. Искусственный интеллект: современный подход (AIMA-2). — М. ; СПб. ; Киев : Вильямс, 2015.
126. Ноэль Э. Массовые опросы. — М. : Прогресс, 1978.
127. Обозов И. И. Психологическая культура взаимных отношений. — М. : Знание, 1986.
128. Орехов А. И. Формирование приемов эффективного решения творческих задач / Дис. канд. психол. наук. — М., 1985.
129. Осипов Г. С. Метод формирования и структурирования модели знаний для одного типа предметных областей // Известия АН СССР. Техническая кибернетика (2). — 1988. — С. 3–12.
130. Осипов Г. С. Приобретение знаний интеллектуальными системами. — М. : Наука ; Физматлит, 1997.
131. Основы теории коммуникации: Учебник / Под ред. проф. М. А. Василика. — М. : Гардарики, 2007.
132. Пажитнов Л. А. Логическая структура компьютерной игры // Микропроцессорные средства и системы (3).— 1987. — С. 11–13.
133. Панфилова А. П. Инновационные педагогические технологии. — М. : Издательский центр «Академия», 2009.
134. Пашкина Е. С., Зарубина Т. В. SNOMED CT и проблемы терминологических систем // Врач и информационные технологии (1). — 2013. — С. 54–62.
135. Пивоварова Л. Онтологии верхнего уровня (обзор). Компьютерная лингвистика и развитие семантического поиска в Интернете — IMS 2010. — 2010.
136. Пиз А., Гарднер А. Язык разговора. — М. : ЭКСМО-Пресс, 2000.
137. Погосян Г. А. Метод интервью и достоверность социологической информации. — Ереван : АН Арм. ССР, 1985.
138. Попов Э. В. Общение с ЭВМ на естественном языке. — М. : УРСС, 2004.
139. Попов Э. В., Фоминых И. Б., Кисель Е. Б., Шапот М. Д. Статические и динамические экспертные системы. — М. : Финансы и статистика, 1996.
140. Попов Э. В. Динамические интеллектуальные системы в управлении и моделировании. — М. : МИФИ, 1996.
141. Поспелов Д. А. Искусственный интеллект: фантазия или наука? — М. : Радио и связь, 1986.
142. Поспелов Д. А. Ситуационное управление: теория и практика. — М. : Наука, 1986.
143. Поспелов Д. А., Осипов Г. С. Прикладная семиотика // Новости искусственного интеллекта. — М., 1999. — (1). — С. 9–35.
144. Поспешный А. С., Стиренко С. Г. Эффективный алгоритм рассуждений для OWL 2 EL онтологий с типизированными выражениями на базе логического процессора ELK. Адаптивні системи автоматичного управління (20). — 2012. — С. 106–115.
145. Продеус А. Н., Захрабова Е. Н. Экспертные системы в медицине. — М. : Век +, 2004.
146. Райзберг Б. А., Лозовский Л. Ш., Стародубцева Е. Б. Современный экономический словарь. — М. : Инфра-М, 2007.

147. Рапуто А. Г. Дескриптивное моделирование образного мышления при репрезентации дидактических объектов // Сб. «Ученые записки» (34). — М. : ИИО РАО, 2011. — С. 114–116.
148. Распопов И. П. Актуальное членение предложения. — Уфа, 1961.
149. Ребельский И. В. Азбука умственного труда. ЭКО (7). — 1989. — С. 143–150.
150. Романовский Д. Ю., Кознов Д. В. Язык DRL для проектирования и разработки документации семейства программных продуктов // Вестник СПбГУ, Серия «Информатика» (4).— 2007. — С. 110–122.
151. Роэм Д. Визуальное мышление. Решение проблем и продажа идей при помощи картинок на салфетке / Пер. с англ. О. И. Медведь. — М. : Эксмо, 2009.
152. Рубашкин В. Ш. Онтологическая семантика. Знания. Онтологии. Онтологически ориентированные методы информационного анализа текстов. — М. : ФИЗМАТЛИТ, 2001.
153. Рыбина Г. В. Инструментальные средства построения динамических интегрированных экспертных систем: развитие комплекса АТТЕХНОЛОГИЯ. Искусственный интеллект и принятие решений (1). — 2010. — С. 41–48.
154. Соколов А. Н. Проблемы научной дискуссии. — Л. : Наука, 1980.
155. Соловьев С. Ю., Соловьева Г. М. Архитектура экспертной системы ФИАКР // Математические исследования, вып. 107. — Кишинев : Штиинца, 1989.
156. Солсо Р. Когнитивная психология. — СПб. : Питер, 2006.
157. Справочник по искусственному интеллекту в 3-х т. / Под ред. Э. В. Попова и Д. А. Поспелова. — М. : Радио и связь, 1990.
158. Спрингер С., Дейч Г. Левый мозг, правый мозг. — М., 1983.
159. Судаков Б, Ярмак М. Формальное представление семантики естественно-языковых текстов в экспертных системах. — 2012.
160. Сукиасян Э. Р. Каталогизация и классификация. Электронные каталоги и автоматизированные библиотечные системы: избр. ст. — СПб. : Профессия, 2009.
161. Сукиасян Э. Р. Логика развития информационно-поисковых языков // Научные и технические библиотеки (4). — 2004.
162. Сукиасян Э. Р. Школа индексирования : Практ. пособие. — М. : Либерея, 2006.
163. Тавокин Е. П. Основы методики социального исследования. — М. : Инфра-М, 2009.
164. Тадеусевич Р. и др. Элементарное введение в технологию нейронных сетей с примерами программ. Горячая линия. — 2011.
165. Тарасов В. Б. От многоагентных систем к интеллектуальным организациям: философия, психология, информатика. — М. : УРСС, 2002.
166. Тельнов Ю. Ф. Инжиниринг предприятий на основе интеллектуальных технологий // Информационно-измерительные и управляющие системы (6). — 2013. — С. 55–59.
167. Тельнов Ю. Ф. Реинжиниринг бизнес-процессов: компонентная методология. — М. : Финансы и статистика, 2004.
168. Тузовский А. Ф., Чириков С. В., Ямпольский В. З. Системы управления знаниями (методы и технологии). — Томск : Изд-во НТЛ, 2005.
169. Тузовский А. Ф. Онтолого-семантические модели в корпоративных системах управления знаниями. — 2007.
170. Тузовский А. Ф., Ямпольский В. З. Интеграция информации с использованием технологий Semantic Web // Проблемы информатики (2). — 2011. — С. 51–58.
171. Уотермен Д. Руководство по экспертным системам. — М. : Мир, 1989.
172. Финн В. К. Логические проблемы информационного поиска. — М. : Наука, 1976.
173. Финн В. К. Интеллектуальные системы и общество // Сб. статей. Серия «Науки об искусственном». Изд. 2. — УРСС, 2006.
174. Фомин А. Н. Методики прогнозирования динамики биржевого индекса [Текст] : аналит. доклад (Александр Фомин). — М., 2012.
175. Фопель К. Психологические принципы обучения взрослых. — М. : Генезис, 2010.
176. Хабаров С. Экспертные системы. — М., 2001.
177. Хант Д. Искусственный интеллект. — М. : Мир, 1986.
178. Хейес-Рот и др. Построение экспертных систем / Под ред. Ф. Хейес-Рота, Д. Уотермена, Д. Лената. — М. : Мир, 1987.
179. Холодная М. А. Когнитивные стили. — М. : ПЕР СЭ, 2002.
180. Хорошевский В. Ф. Управление проектами в интеллектуальной системе PIES Workbench // Изв. РАН, серия «Техническая кибернетика» (5). — 1993. — С. 71–98.
181. Хорошевский В. Ф. Пространства знаний в сети Интернет и Semantic Web (Часть 3) // Искусственный интеллект и принятие решений (1). — 2012. — С. 3–38.
182. Хювянен Р., Сеппянен М. Мир ЛИСПА в 2-х т. — М. : Мир, 1991.

183. Цейтин Г. С. Программирование на ассоциативных сетях. ЭВМ в проектировании и производстве, вып. 2 // Под ред. Г. В. Орловского. — 1985. — С. 16–48.
184. Частиков А. П., Гаврилова Т. А., Белов Д. Л. Разработка экспертных систем. Среда CLIPS. — СПб. : БХВ-Петербург, 2003.
185. Черниговская Т. В. Симметричные мужчины vs. асимметричные женщины: каким глазом подмигивать, или зеленая тоска // В кн. Языки страха: женские и мужские стратегии поведения. Материалы международной научно-практической конференции 3–5 октября 2003 г., Пропповский центр, СПбГУ. — 2004.
186. Черниговская Т. В., Деглин В. Л. Метафорическое и силлогистическое мышление как проявление функциональной асимметрии мозга. — Тарту, 1986.
187. Черняк Л. Интеграция данных: синтаксис и семантика. Открытые системы (10). — 2009.
188. Шаров С. А. Использование объектно-ориентированного программирования для лингвистического моделирования // Труды Международного семинара «ДИАЛОГ'95». — 1995. — С. 332–339.
189. Шатуновский И. Б. Основные коммуникативные типы полных (общих) вопросов в русском языке. Русский язык: пересекая границы. — Дубна, 2001. — С. 246–265.
190. Шенк Р., Бирнбаум Л., Мей Дж. К интеграции семантики и прагматики. Новое в зарубежной лингвистике. XXIV. Компьютерная лингвистика. — М. : Прогресс, 1989.
191. Шенк Р., Хантер Л. Познать механизмы мышления. Реальность и прогнозы искусственного интеллекта. — М. : Мир, 1987.
192. Шепотов Е. Г., Шмаков Б. В., Крикун П. Д. Методы активизации мышления. — Челябинск : ЧПИ, 1985.
193. Шумилина Т. В. Интервью в журналистике. — М. : МГУ, 1973.
194. Шэннон К., Уивер У. Математическая теория связи. — М., 1963.
195. Эндрю А. Искусственный интеллект. — М. : Мир, 1985.
196. Эталонные модели организации деятельности в государственном секторе: [Текст] Отчет по НИР / Рук. А. В. Клименко. ИПГМУ ВШЭ, Бизнес Инжиниринг Групп. — 2006.
197. Abelson R. P. Psychological status of the script concept // American Psychologist (36). — 1981. — P. 715–729.
198. Abelson H., Sussman G. J. Structure and Interpretation of Computer Programs. — 1985.
199. Adeli H. Knowledge Engineering. — New York : McGraw-Hill Publishing Company, 1994.
200. Adeli H. (Ed.). Expert systems in construction and structural engineering. — CRC Press, 2003.
201. Alexander C. The timeless way of building (Vol. 1). — New York : Oxford University Press, 1979.
202. Allen J. F. Maintaining knowledge about temporal intervals. Communications of the ACM 26 (11). — 1983. — P. 832–843.
203. Anderson J. R., Bower Gordon H. Human Associative Memory. — Hillsdale : New Jersey, 1979.
204. Anderson J. R. The Architecture of Cognition. — Cambridge : Harvard University Press, MA, 1983.
205. Andersson B., Bergholtz M., Edirisuriya A., Ilayperuma T., Johannesson P., Gordijn J., Gregoire B., Schmitt M., Dubois E., Abels S., Hahn A., Wangler B., Weigand H. Towards a reference ontology for business models. Proceedings of 25th International Conference on Conceptual Modeling Conceptual Modeling (ER 2006). — Berlin ; Heidelberg : Springer, 2006. 482–496.
206. Ashburner M. et al. Gene Ontology: tool for the unification of biology. Nature genetics 25 (1). — 2000. — P. 25–29.
207. Aussenac-Gilles N. Making the Method Solving Explicit with MACAO: SIZYPHUS, Case-Study. In: Linster, M. (Ed.): Sisyphus'92: Models of problem solving, GMD-Arbeitspapiere. — 1993. — P. 1–21.
208. Ausubel D. P. Educational psychology: A cognitive view. — New York : Holt, Rinehart and Winston, 1968.
209. Bahrammirzaee A. A comparative survey of artificial intelligence applications in finance: artificial neural networks, expert system and hybrid intelligent systems. Neural Computing and Applications, 19 (8).— 2010. — P. 1165–1195.
210. Barlett F. Remembering: An experimental and social study. — London : Cambridge University Press, 1932.
211. Beimel D., Peleg M. Using OWL and SWRL to represent and reason with situation-based access control policies. Data & Knowledge Engineering 70 (6).— 2011. — P. 596–615.

212. Bellezza F. S., Bower G. H. Remembering script-based text. Poetics 11 (1). — 1982. — P. 1–23.
213. Berners-Lee T., Hendler J., Lassila O. The Semantic Web. Scientific American, May 17, 2001 / Пер. Е. Золина. — http://ezolin.pisem.net/logic/semantic_web_rus.html.
214. Berry M. J., Linoff G. S. Mastering data mining: the art and science of customer relationship management. Amazon. Bishop, Christopher M. Pattern Recognition and Machine Learning. — Springer, 2006.
215. Beimel D., Peleg M. Using OWL and SWRL to represent and reason with situation-based access control policies. Data & Knowledge Engineering, 70(6). — 2011. — P. 596–615.
216. Bodenreider O. The unified medical language system (UMLS): integrating biomedical terminology. Nucleic acids research 32 (1). — 2004. — P. 267–270.
217. Boose J. H. A research framework for knowledge acquisition techniques and tools. Proceedings of 3-rd European Knowledge Acquisition for Knowledge-based Systems Workshop-Ekat 89.— Paris, 1989.
218. Brachman R. J. On the epistemological status of semantic networks. In Associative networks: Representation and use of knowledge by computers. Acad. — New York : Press, 1979. — P. 3–50.
219. Brank J., Grobelnik M., Mladenic D. A survey of ontology evaluation techniques. In In Proceedings of the Conference on Data Mining and Data Warehouses (SiKDD 2005). — Ljubljana ; Slovenia, 2005. — http://kt.ijs.si/dunja/sikdd2005/Papers/BrankEvaluationSiKDD2005.pdf.
220. Brank J., Grobelnik M. & Mladenić D. Automatic evaluation of ontologies.— London : Springer, 2007. — P. 193–219.
221. Brownston L., Farrel R., Kant E., Martin N. Programming Expert Systems in OPS5. An Introduction to Rule-Based Programming. Reading, Addison-Wesley: MA. — 1985.
222. Brusilovsky P., Sosnovsky S., Lee D. H., Yudelson M., Zadorozhny V., Zhou X. An open integrated exploratorium for database courses. ACM SIGCSE Bulletin 40 (3). — 2008. — P. 22–26.
223. Buchanan B. G. & Feigenbaum E. A. DENDRAL and Meta-DENDRAL: Their applications dimension. Artificial intelligence, 11(1). — 1978. — P. 5–24.
224. Burkhard R., Meier M., Smis M., Allemang J. & Honisch L. Beyond Excel and Powerpoint: knowledge maps for the transfer and creation of knowledge in organizations in information visualization. Proceedings of the Ninth International Conference on IEEE. — 2005. — P. 76–81.
225. Cacas A. J., Hill G. & Lott J. Support for constructing knowledge models in CmapTools Technical Report IHMC. Pensacola, FL. Institute for Human and Machine Cognition 2. — 2003.
226. Ceccaroni L., Cortés U., Sanchez-Marre M. OntoWEDSS: augmenting environmental decision-support systems with ontologies //Environmental Modelling & Software 19 (9). — 2004. — P. 785–797.
227. Ceccaroni L., Cortés U and Sánchez-Marré M. WaWO – An ontology embedded into an environmental decision-support system for wastewater treatment plant management. In Proceedings of ECAI2000 — Wo9: Applications of ontologies and problem-solving methods. — Berlin, Germany, 2000. — P. 2.1–2.9
228. Ceccato S. Linguistic analysis and programming for mechanical translation. — New York : Gordon & Breach, 1961.
229. Chairniak E. Toward a model of children story comprehension. Report No. TR-266, AI Lab. MIT. — 1972.
230. Chang S. K. (Editor). Handbook of Software Engineering and Knowledge Engineering. Vol 1: Fundamentals Vol 2 // Emerging Technologies. — World Scientific Publishing Co. Inc., 2002.
231. Chiou Chei-Chang. The effect of concept mapping on students' learning achievements and interests. Innovations in Education and Teaching International 45 (4). — 2008. — P. 375–387.
232. Chorafas D. N. Knowledge Engineering. — Wiley, John & Sons, Inc., 1990.
233. Christensen R. Plane answers to complex questions: the theory of linear models. — Springer Science & Business Media, 2011.
234. Christiansen P. E., Gotze J. International Enterprise Architecture survey. Trends in governmental Enterprise Architecture on a national level. — 2006.
235. Christianson M. K., Farkas M. T., Sutcliffe K. M., Weick K. E. Learning Through Rare Events: Significant Interruptions at the Baltimore & Ohio Railroad Museum. Organization Science 20 (5).— 2009. — P. 846–860.

236. Churchman C. W., Ackoff L. E. Introduction to Operations Research. — New York : Wiley, 1957.
237. Cornet R., de Keizer N. Forty years of SNOMED: a literature review. BMC Medical Informatics and Decision Making. — 2008.
238. Cruse D. A. Lexical semantics. — Cambridge University Press, 1986.
239. Cuena J. (Editor). Knowledge Engineering and Agent Technology. IOS. — 2000.
240. Davenport T. H., Prusak L. Working Knowledge. — Harvard Business School Press, 2000.
241. David J. M., Krivine J. P. & Simmons R. (Eds.). Second generation expert systems. — Springer Science & Business Media, 2012.
242. Davis R. TEIRESIAS: Applications of meta-level knowledge. In: Knowledge-based Systems in Artificial Intelligence. — New York : McGraw-Hill, 1982.
243. De Leenheer P., de Moor A., Meersman R. Context dependency management in ontology engineering: A formal approach, J. Data Semantics (8). — 2007. — P. 26–56.
244. De Leenheer P., Debruyne C. DOGMA-MESS: A tool for fact-oriented collaborative ontology evolution, in: On the Move to Meaningful Internet Systems: OTM2008Workshops of LNCS, Springer. (5333). — 2008. — P. 797–806.
245. De Moor A., De Leenheer P., Meersman R. DOGMA-MESS: A meaning evolution support system for interorganizational ontology engineering, in: 14th International Conference on Conceptual Structures, ICCS of Lecture Notes in Computer Science, Springer (4068). — 2006. — P. 189–202.
246. Debenham J. Knowledge Engineering: Unifying Knowledge Base and Database Design. Springer Verlag. Document Engineering: Analyzing and Designing Documents for Business Informatics and Web Services. — The MIT Press, 1998.
247. Dellschaft K., Engelbrecht H., Barreto J. M., Rutenbeck S., Staab S., Cicero. Tracking design rationale in collaborative ontology engineering, in: The Semantic Web: Research and Applications, 5th European Semantic Web Conference (ESWC), LNCS, Springer. — 2008. — P. 782–786.
248. Domingue J., Fensel D., Hendler J. A. (ed.). Handbook of semantic web technologies. — Springer Science & Business Media, 2011.
249. Dopico J. R. R., de la Calle J. D., Sierra A. P. Encyclopedia of artificial intelligence. — New York : Information Science Reference (1677), 2009.
250. Dubois M., Annavaram M., Stenström P. Parallel computer organization and design. — Cambridge University Press, 2012.
251. Durkin J. Expert Systems: Design and Development. Prentice Hall PTR, Upper Saddle River. — N. Y., 1998.
252. Dzbor M., Domingue J., Motta E. Magpie. Towards a Semantic Web Browser, In Proc. of ISWC2003.— P. 690–705.
253. Egan G. The Skilled Helper. 5th ed. Brooks / Cole: Belmont, CA. — 1994.
254. Eisenstadt M., Domingue J., Rajan T., Motta E. Visual Knowledge Engineering, IEEE Transactions on Software Engineering 16 (10). — 1986. — P. 164–1177.
255. Ellis G., Levinson R. The Birth of PEIRCE: A conceptual graphs workbench. Proc. 1st International Workshop on PEIRCE. — New Mexico State University : Las Cruces, 1992. — P. 149–156.
256. Euzenat J. State of the art on ontology alignment. KnowledgeWeb Report D2.2.3. — 2004.
257. Euzenat J., Shvaiko P. Ontology matching. — Heidelberg : Springer, 2013.
258. Fadel F. G., Fox M. S., Gruninger M. A Generic Enterprise Resource Ontology. Proceedings of 3rd Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, Morgantown, WV. — 1994.
259. Fensel D. Ontologies: a silver bullet for Knowledge Management and Electronic Commerce. — Berlin : Springer Verlag, 2001.
260. Fettke P. (ed.). Reference modeling for business systems analysis. IGI Global. — 2006.
261. Fikes R. E., Nilsson N. J. STRIPS: A new approach to the application of theorem proving to artificial intelligence. Artificial Intelligence, 1(2). — 1971.
262. Fillmore C. The case for case. In Bach E. and Harms R., editors, Universals in Linguistic TheoryHolt. — New York : Rhinehart and Winston, 1968. — P. 1–90.
263. Forgy C. L. RETE: a fast algorithm for the many pattern/many object pattern match problem // Artificial Intelligence 19 (1). — 1982. — P. 17–37.
264. Fox M., Barbuceanu M., Gruninger M., Lin J. An Organization Ontology for Enterprise Modelling. Simulating Organizations: Computational Models of Institutions and Groups, Menlo Park CA: AAAI/MIT Press. — 1997. — P. 131–152.

265. Fox P. et al. Semantically-enabled large-scale science data repositories. The Semantic Web-ISWC 2006. — Berlin ; Heidelberg : Springer , 2006. — P. 792–805.
266. Fox M. The TOVE Project: A Common-sense Model of the Enterprise, Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Belli, F. and Radermacher, F. J. (Eds.), Lecture Notes in Artificial Intelligence № 604. — Berlin : Springer-Verlag, 1992. — P. 25–34.
267. Gaines. Second Generation Knowledge Acquisition Systems. Proceedings of the Second European Knowledge Acquisition Workshop: Bonn 17. — 1989. — P. 1–14.
268. Gammack J. G., Young R. M. Psychological techniques for eliciting expert knowledge. In Research and Development in Expert Systems. — University Press: Cambridge, 1985.
269. Gangemi A., Guarino N., Masolo C., Oltramari A. Understanding Top-Level Ontological Distinctions 11 Proc. of IJCAI-2001 Workshop on Ontologies and Information Sharing. — 2001. — P. 26–33.
270. Gavrilova T., Andreeva T. Knowledge elicitation techniques in a knowledge management context // Journal of Knowledge Management 16 (4).— 2012. — P. 523–537.
271. Gerstl P., Pribbenow S. A conceptual theory of part-whole relations and its applications // Data & Knowledge Engineering 20 (3).— 1996. — P. 305–322.
272. Ghoniem M., Fekete J.D., Castagliola P. On the readability of graphs using node-link and matrix-based representations: a controlled experiment and statistical analysis // Information Visualization 4 (2). — 2005. — P. 114–135.
273. Giarratano J., Riley G. Expert Systems, Principles and Programming. — Boston : MA. PWS Publishing, 2005.
274. Glimm B., Horrocks I., Motik B., Stoilos G., Wang Z. HermiT: an OWL 2 reasoner // Journal of Automated Reasoning 53 (3). — 2014. — P. 245–269.
275. Goldsmith T. E, Johnson P. J. & Acton W. H. Assessing structural knowledge // Journal of Educational Psychology 83. — 1991. — P. 88–96.
276. Gómez-Pérez A., Fernández-López M., Corcho O. Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web. — Springer Science & Business Media, 2004.
277. Graeser A. C., Gordon S. E., Sawyer J. D. Recognition memory for typical and atypical actions in scripted activities: Tests of a script pointer + tag hypothesis // Journal of Verbal Learning and Verbal Behavior 18. — 1979. — P. 319–332.
278. Grigoriev L., Kudryavtsev D. ORG-Master: Combining Classifications, Matrices and Diagrams in the Enterprise Architecture Modeling Tool // Proceedings of the 4th Conference on Knowledge Engineering and Semantic Web, October 7–9, 2013. Communications in Computer and Information Science (CCIS) Series — Springer, 2013. — P. 250–258.
279. Gruber T. R. A translation approach to portable ontology specifications // Knowledge acquisition 5 (2). — 1993. — P. 199–220.
280. Gruninger M., Atefi K., Fox M.Ontologies to support process integration in enterprise engineering// Computational and Mathematical Organization Theory, 6.— 2000. — P. 381–394.
281. Grüninger M., Fox M. Methodology for the design and evaluation of ontologies. Workshop on Basic Ontological Issues in Knowledge Sharing. — Montreal, Canada, 1995.
282. Gruninger M., Fox M. S. An activity ontology for enterprise modelling. Department of Industrial Engineering.— University of Toronto, 1994.
283. Guarino N. Some ontological principles for designing upper level lexical resources. Proc. Of First International Conference on Language Resources and Evaluation.— Granada, Spain, 1995.
284. Guarino N., Welty C. Evaluating ontological decisions with OntoClean. Comm. of the ACM, 45(2).— 2002. — P. 61–65.
285. Guarino N. Formal ontology in information systems. Proceedings of the first international Formal Ontology in Information Systems conference FOIS'98, June 6–8. — Trento, Italy : IOS press, 1998. — P. 3–15.
286. Guijarro L. Interoperability frameworks and enterprise architectures in e-government initiatives in Europe and the United States. Government Information Quarterly 24 (1).— 2007. — P. 89–101.
287. Gullen J., Bryman A. The Knowledge Acquisition Bottleneck: Time For Reassessment. Expert Systems 5 (3). — 1988.
288. Hartmann J. et al. Methods for ontology evaluation. Knowledge Web Deliverable D1.2.3. Available at http://www.starlab.vub.ac.be/research/projects/knowledgeweb/KWeb-Del-1.2.3-Revised-v1.3.1.pdf. — 2005.

289. Heidari Farideh, et al. A Meta-Meta-Model For Seven Business Process Modeling Languages. Business Informatics (CBI), 2013 IEEE 15th Conference on. IEEE.— 2013.
290. Henderson-Sellers B., Ralyt J. Situational Method Engineering: State-of-the-Art Review // Journal of Universal Computer Science 16 (3). — 2010. — P. 424–478.
291. Hennessy J. L., Patterson D. A. Computer architecture: a quantitative approach.— Elsevier, 2011.
292. Hepp M., Roman D. An ontology framework for semantic business process management. Wirtschaftinformatik Proceedings 2007. — 2007. — P. 423–440.
293. Horridge M. et. al. A Practical Guide To Building OWL Ontologies Using Protégé 4 and CO-ODE Tools Edition 1.3. — The University Of Manchester, 2011.
294. Horrocks I., Patel-Schneider P. F., Boley H., Tabet, S., Grosof B. & Dean M. SWRL: A semantic web rule language combining OWL and RuleML. W3C Member submission 21. — 2004. — P. 79.
295. Hudson L. Contrary Imaginations; a psychological study of the English Schoolboy. — Harmondsworth : Penguin, 1967.
296. Hutchins W. J. Machine Translation: Past, Present, Future, Ellis Horwood: Chichester, England. Available at http://ourworld.compuserve.com/homepages/WJHutchins/PPF-TOC.htm. — 1986.
297. Hyerle D. Visual Tools for Transforming Information Into Knowledge. Thousand Oaks, CA. — Corwin Press, 2009.
298. Iacob M., Jonkers H., Lankhorst M., Proper E. ArchiMate 2.0 Specification / D. A. C. Quartel // The Open Group Publications Catalog, 2009–2012. URL: http://pubs.opengroup.org/architecture/archimate2-doc/ (дата обращения: 10.02.2013). — 2012.
299. Ishikawa K. Guide to Quality Control. Tokyo: Asian Productivity Organization. — 1976.
300. Jeffery A. B., Maes J. D., Bratton-Jeffery M. F. Improving team decision-making performance with collaborative modeling. Team Performance Management // An International Journal 11 (1/2). — 2005. — P. 40–50.
301. Jensen K., Heidorn G. E., Richardson S. D. (Eds.). Natural language processing: the PLNLP approach (Vol. 196). — Springer Science & Business Media, 2012.
302. Johansson I. On the transitivity of the parthood relations. Relations and predicates. — Frankfurt : Ontos Verlag, 2004. — P. 161–181.
303. Katifori A., Halatsis C., Lepouras G., Vassilakis C., Giannopoulou E. Ontology visualization methods — a survey. ACM Comput. Surv. 39(4). — 2007.
304. Kazakov Y., Klinov P. Advancing ELK: Not Only Performance Matters. In Diego Calvanese, Boris Konev, eds. : Proceedings of the 28th International Workshop on Description Logics (DL-15). CEUR Workshop Proceedings. — 2015.
305. Kazakov Y., Krötzsch M., Simančík F. ELK: A Reasoner for OWL EL Ontologies. System Description. — 2012.
306. Keller R., Eckert C. M., Clarkson P. J. Matrices or node-link diagrams: which visual representation is better for visualising connectivity models? Information Visualization 5. — 2006. — P. 62–76.
307. Kelly R. Practical Knowledge Engineering. Digital press. — 1999.
308. Kendal S., Creen M. An Introduction to Knowledge Engineering. Springer. — 2006.
309. Kifer M. Rule interchange format: The framework. Web reasoning and rule systems. — Springer Berlin Heidelberg, 2008. — P. 1–11.
310. Kifer M., Boley H. RIF overview. W3C Working Group Note. — 2010.
311. Kim H. M., Fox M. S., Grüninger M. An ontology for quality management—enabling quality problem identification and tracing // BT Technology Journal 17 (4). — 1999. — P. 131–140.
312. Kinchin I. M., Hay D. B., Adams A. How a qualitative approach to concept map analysis can be used to aid learning by illustrating patterns of conceptual development // Educational research 42 (1). — 2000. — P. 43–57.
313. Kocura P., Ho K. K., Moorehouse D., Sharpe G. Aspects of conceptual graphs processor design. Proc. 6th Annual Workshop on Conceptual Graphs. Binghamton. — New York, 1991. — P. 317–329.
314. Kontopoulos E., Vrakas D., Kokkoras F., Bassiliades N., Vlahavas I. An Ontology-based Planning System for e-Course Generation. Expert Systems with Applications 35 (1). — 2008. — P. 398–406.
315. Kotis K. On supporting HCOME-3O ontology argumentation using semantic wiki technology, in: On the Move to Meaningful Internet Systems: OTM 2008 Workshops. — 2008. — P. 193–199.

316. Kozaki K., Hayashi Y., Sasajima M., Tarumi S., Mizoguchi R. Understanding Semantic Web Applications. Proc. of the 3rd Asian Semantic Web Conference (ASWC 2008), February 2–5. — Bangkok, Thailand, 2008.— P. 524–539.
317. Kozma R. B. Constructing knowledge with learning tool. In P. Kommers, D. Jonassen, A. T. Mayes (Eds.), Cognitive tools for learning Berlin. — Springer-Verlag, 1992. — P. 23–32.
318. Kruse L. Drehbücher für Verhaltensschauplätze oder: Skripts für Settings. Ordnung und Variabilität im Alltagsgeschehen. Göttingen etc. Hogrefe.— 1986. — P. 135–153.
319. Kudryavtsev D., Grigoriev L. Ontology-based business architecture engineering technology. The 10th International Conference on Intelligent Software Methodologies, Tools and Techniques, September 28–30, 2011. — 2011. — P. 233–252.
320. Kudryavtsev D., Grigoriev L. Systemic approach towards enterprise functional decomposition. The workshop «Convergence of Business Architecture, Business Process Architecture, Enterprise Architecture and Service Oriented Architecture» within the 13th IEEE Conference on Commerce and Enterprise Computing, Luxembourg, September 5–7, 2011. — 2011. — P. 310–317.
321. Kudryavtsev D., Gavrilova T. From Anarchy to System: a Novel Classification of Visual Knowledge Codification Techniques. Knowledge and Process Management // The Journal of Corporate Transformation. In press. — 2016.
322. Kuipers B. Commonsense reasoning about causality: deriving behavior from structure // Artificial intelligence 24 (1). — 1984. — P. 169–203.
323. Kuipers B. Qualitative reasoning: modeling and simulation with incomplete knowledge. MIT press. — 1994.
324. Kunz W., Rittel H. W. J. Issues as elements of information systems. Berkeley, California : Institute of Urban and Regional Development. — University of California, 1970. — P. 131.
325. Laera L., Tamma V., Bench-Capon T. & Semeraro G. SweetProlog: A System to Integrate Ontologies and Rules. Rules and Rule Markup Languages for the Semantic Web. — 2004. — P. 188–193.
326. Lakoff G., Johnson M. Metaphors we live by. — Chicago : U of Chicago P., 2003.
327. Lambe P. Organising knowledge: taxonomies, knowledge and organisational effectiveness. — Chandos Publishing, 2007.
328. Latfi F., Lefebvre B., Descheneaux C. Ontology-Based Management of the Telehealth Smart Home, Dedicated to Elderly in Loss of Cognitive Autonomy. Proceeding of the OWLED. — 2007.
329. Legat C., Neidig J. Roshchin M. Model-based Knowledge Extraction for Automated Monitoring and Control. Proceedings of the 18th IFAC World Congress. — Milan, 2011.
330. Lehman F. W. Semantic networks // Computers and Mathematics with Applications. 23 (2–5). — 1992. — P. 1–50.
331. Lehnert W. G. The role of scripts in understanding Frame conceptions and text understanding. B. — N. Y. : Gruyter, 1980. — P. 79–95.
332. Lengler R., Eppler M. Towards A Periodic Table of Visualization Methods for Management. IASTED Proceedings of the Conference on Graphics and Visualization in Engineering (GVE 2007). — Clearwater, Florida, USA, 2007.
333. Levine I. Robert, Diane E. Drang, Edelson B. A Comprehensive guide to AI and expert system. Byte Books. — 1988.
334. Lim B. C., Klein K. J. Team mental models and team performance: A field study of the effects of team mental model similarity and accuracy // Journal of Organizational Behavior 27 (4).— 2006. — P. 403–418.
335. Liu Y., Xu C., Zhang Q., Pan Y. The Smart Architect: Scalable Ontology-Based Modeling of Ancient Chinese Architectures. IEEE Intelligent Systems 1. — 2008. — P. 49–56.
336. Makela E., Hyvonen E., Saarela S. Ontogator — A Semantic View-Based Search Engine Service for Web Applications, In Proc. of ISWC 2006. — P. 847–860.
337. Mann G. BEELINE — A Situated, Bounded Conceptual Knowledge System // International Journal of Systems Research and Information Science 7. — 1995. — P. 37–53.
338. Margulies N., Valenza C. Visual Thinking Tools for Mapping Your Ideas. Bethel, CT. — USA : Crown House, 2005.
339. Martin A., Manjula M., Venkatesan V. P. A Business Intelligence Model to Predict Bankruptcy using Financial Domain Ontology with Association Rule Mining Algorithm. IJCSI. — 2011.
340. Martínez-Romero M., Vázquez-Naya J., Pereira J., Pereira M., Pazos A., Baños G. The iOSC3 System: Using Ontologies and SWRL Rules for Intelligent Supervision and Care of Patients

with Acute Cardiac Disorders. Computational and Mathematical Methods in Medicine. — 2013.
341. Masterman M. Semantic message detection for machine translation, using an interlingua. Proc. of Int. // Conference on machine translation. — 1961. — P. 438–475.
342. Matarozzo J. D., Wettman M., Weins A. N. Interviewer influence on duration of interview speech // J. Verbal Learning and Verbal Behavior 1. — 1963.
343. Matthes F. et al. Enterprise architecture management tool survey. — München : Techn. Univ., 2008.
344. Mayall A., Carter J. The Essential Project: Harnessing Conceptual Structures to Expose Organizational Dynamics // International Journal of Conceptual Structures and Smart Applications (IJCSSA). 3(2).— 2015. — P. 1–11.
345. Masolo C., Borgo S., Gangemi A., Guarino N., Oltramari A. Wonderweb Deliverable D18: Ontology Library (final). Technical report, Laboratory for Applied Ontology — Trento : ISTC-CNR, 2003.
346. McGuinness D. Ontologies Come of Age. In Dieter Fensel, Jim Hendler, Henry Lieberman, and Wolfgang Wahlster, editors. Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential. — MIT Press, 2003.
347. Melioli G., et al. Allergenius, an expert system for the interpretation of allergen microarray results // World Allergy Organ J 7.1.— 2014. — P. 15.
348. Meyer B. Object oriented software construction. — 1988.
349. Mikulescky L. Development of Interactive computer programs to help students transfer basic skills to college level science and behavioural sciences courses. Bloominglon IN. — Indiana University, 1988.
350. Minsky M. A Framework for Representing Knowledge. MIT-AI Laboratory: Memo 306. — 1974. (Русский перевод: Минский М.Л. Фреймы для представления знаний. — М. : Энергия, 1979).
351. Minsky M. Theory of memory. Perspectives in Cognitive Science. — Norwood, N.S. : Ablex, 1981.
352. Minsky M., ed. Semantic Information Processing. — Cambridge : MIT Press, MA, 1968.
353. Moore R. L., Rosenhof H. & Stanley G. Process control using the G2 real-time expert system // In Industry Applications Society Annual Meeting, 1989., Conference Record of the 1989 IEEE. — 1989. — P. 1452–1456.
354. Motschnig-Pitrik R., Kaasboll J. Part-whole relationship categories and their application in object-oriented analysis // Knowledge and Data Engineering, IEEE Transactions 11 (5) — 1999. — P. 779–797.
355. Munday C., Sobora F., Lukose D. UNE-CG-KEE: next generation knowledge engineering environment. Proc. 1st Australian Conceptual Structures Workshop. — Armidale, Australia, 1994. — P. 103–117
356. Nan C., Khan F., Iqbal M. T. Real-time fault diagnosis using knowledge-based expert system. process safety and environmental protection, 86(1). — 2008. — P. 55–71.
357. Negnevitsky M. Artificial Intelligence — A Guide to Intelligent Systems. Addison-Wesley. — 2002.
358. Newell A., Shaw J. C., Simon H. A. Report on a general problem-solving program. Proceedings of the International Conference on Information Processing. — 1959. — P. 256–264.
359. Novak J., Cacas A. The Theory Underlying Concept Maps and How to Construct Them. Technical Report IHMC Cmap. 1, Florida Institute for Human and Machine Cognition. Available at: http://cmap.ihmc.us/Publications/ResearchPapers/TheoryUnderlyingConceptMaps.pdf. — 2006.
360. Novak J. Concept maps and vee diagrams: Two metacognitive tools for science and mathematics education // Instructional Science 19. — 1990. — P. 29–52.
361. Noy N., McGuinness D. Ontology Development 101: A Guide to Creating Your First Ontology. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, March 2001. Available at: http://protege.stanford.edu/publications/ontology_development/ontology101.html. — 2001.
362. Op't Land M., Proper E., Waage M., Steghuis C. Enterprise Architecture: Creating Value by Informed Governance. — , Berlin : Springer-Verlag, 2009.
363. Chen P. P. The entity-relationship model: toward a unified view of data, ACM TODSl… Proceedings of 1995 Conference on Applications of Databases, ADB 95, San Jose.— 1995. — P. 9–36.

364. Palma R., Corcho O., Gomez-Perez A., Haase P. A. Holistic Approach to Collaborative Ontology Development Based on Change Management. Web Semantics: Science, Services and Agents on the World Wide Web, Volume 9, Issue 3, September 2011. — P. 299–314.
365. Paraiso E., Barthès J. An Ontology-Based Utterance Interpretation in the Context of Intelligent Assistance. V workshop em Tecnologia da Informacao e da Linguagem Humana, XXVII Congresso da SBC. — 2007.
366. Parker-Rhodes A. F. Inferential Semantics. — Sussex : Harvester Press/Humanities Press/, 1978.
367. Pearl J. Probabilistic Reasoning in Intelligent Systems: Network of Plausible Inference. Los Altos, CA: Morgan Kaufmann. — 1988.
368. Pearl J. Causality. — New York : Cambridge University Press, 2000.
369. Pollard E., Morignot P., Nashashibi F. An ontology-based model to determine the automation level of an automated vehicle for co-driving. Information Fusion (FUSION), 16th International Conference on. IEEE. — 2013. — P. 596–603.
370. Pople H. E. Heuristic Methods for Imposing Structure on Ill-Structured Problems: The Structuring of Medical Diagnostics. Chapter 5 in Szolovits, P. (Ed.) Artificial Intelligence in Medicine. — Boulder, Colorado : Westview Press, 1982.
371. Post E. Formal reduction of the general combination problem. American journal of Mathematics 65. — 1943. — P. 197–268.
372. Prcela M., Gamberger D., Jovic A. Semantic web ontology utilization for heart failure expert system design. Studies in health technology and informatics 136. — 2008. — P. 851–856.
373. Quillan M. R. Word concepts: A theory and simulation of some basic capabilities // Behavioral Science 12 (5). — 1967. — P. 410–430.
374. Quillian M. R. Semantic memory. In Semantic Information Processing (Minsky M. eds.). — Cambridge, MA : MIT Press, 1968. — P. 227–270.
375. Rieger C. An organization of knowledge for problem solving and language comprehension // Artificial Intelligence 7 (2). — 1976. — P. 89–127.
376. Roberts, D. The Existential Graphs of Charles S. Peirce. Mouton, The Hague. — 1973.
377. RuleML. The Rule Markup Initiative. Электронный ресурс. URL: http://www.ruleml.org (Дата обращения: 24.08.2015).
378. Sabah G., Vilnat A. Hierarchy of Relational Types in Conceptual Graphs to Handle Natural Language Parsing. Proc. International Conference on Conceptual Structures. — Quebec : Springer-Verlag, 1993. — P. 198–215.
379. Sakamura K. 21st-century microprocessors. IEEE Micro, (4). — 2000.— P. 10–11.
380. Samuel K., Obrst L., Stoutenberg S., Fox K., Franklin P., Johnson A., Peterson J. Translating OWL and semantic web rules into PROLOG: Moving toward description logic programs // Theory and Practice of Logic Programming 8 (3).— 2008. — P. 301–322.
381. Schank R. C., Abelson R. P. Scripts, plans, goals, and understanding: An inquiry into human knowledge structures. — Psychology Press, 1977.
382. Schank R. C. Reminding and memory organization: An introduction to MOPs. Strategies for natural language processing. — N.J. : Hillsdale, 1982. — P. 455–493.
383. Schank R. C. Tell Me A Story: A New Look At Real And Artificial Memory. — New York : Charles Scribner's Sons, 1990.
384. Schank R. C. and Rieger C. Inference and computer understanding of natural language // Artificial Intelligence 5 (4). — 1974. — P. 373–412.
385. Schank R. C. and Riesbeck C. Inside computer understanding: Five programs plus miniatures. — Hillsdale, N. J. : Erlbaum, 1981.
386. Schank R. C., Tesler G. A conceptual parser for natural language. Proc. of the IJCAI-69.— 1969. — P. 569–578.
387. Schlenker D. & Abegg G. Scoring student-generated concept maps in introductory college chemistry. Paper presented at the annual meeting of National Association for Research in Science Teaching. Lake Geneva. I. — 1991.
388. Schönpflug U. Begriffsentwicklung zweisprachiger Kinder. Die Rolle der Psychologie in der Sprachlehrforschung. T. Narr. — 1987. — P. 55–85.
389. Schouksmith G. Assessment through interviewers.— Oxford, 1978.
390. Scott A. C., Clayton J. E., Gibson E. L. A practical guide to knowledge acquisition. Addison-Wesley, Reading, MA. — 1991.
391. Seidewitz E., Stark M. General object-oriented software development. Document No. — SEL-86-002, NASA Goddard Space Flight Center, Greenbelt. — Maryland, 1986.
392. Selz O. Über die Gesetze des geordneten Denkverlaufs. — Stuttgart : Spemann, 1913.

393. Selz O. Zur Psychologie des produktiven Denkens und des Irrtums. — Bonn : Friedrich Cohen, 1922.
394. Sensoy M., Vasconcelos W. W., Norman T. J. Combining semantic web and logic programming for agent reasoning. Advanced Agent Technology. — Springer Berlin Heidelberg, 2012. — P. 428–441.
395. Shapiro S., Rappaport W. The SNePS family, in Lehmann.— 1992. — P. 243–275.
396. Shapiro S. A net structure for semantic information storage, deduction and retrieval, Proc. IJCAI-71. — 1971. — P. 512–523.
397. Shapiro S. The SNePS semantic network processing system. Findler. — 1979. — 263–315.
398. Shortliffe E. H. & Buchanan B. G. A model of inexact reasoning in medicine. Mathematical biosciences, 23(3). — 1975. — P. 351–37.
399. Shue L. Y., Chen C. W., Shiue W. The development of an ontology-based expert system for corporate financial rating // Expert Systems with Applications 36 (2). — 2009. — P. 2130–2142.
400. Shvaiko P., Euzenat J. Ontology matching: state of the art and future challenges // Knowledge and Data Engineering, IEEE Transactions on 25 (1). — 2013. — P. 158–176.
401. Simon H. Otto Selz and information-processing psychology. In N. H. Frijda A. D. de Groot, eds., Otto Selz: His Contribution to Psychology. The Hague: Mouton. — 1981.
402. Singh S. Why Visualization Matters in Elearning Courses? CommLab India blog. — URL: http://blog.commlabindia.com/elearning/visualization-in-elearning. — 2011. (Дата обращения: 19.03.13).
403. Sirin E., Parsia B., Grau B. C., Kalyanpur A., Katz Y. P. A practical OWL-DL reasoner // Web Semantics: science, services and agents on the World Wide Web, 5(2). — 2007. — P. 51–53.
404. Smith B. et al. Relations in biomedical ontologies // Genome biology 6 (5). — 2005. — R. 46.
405. Sosnovsky S., Mitrovic A., Lee D., Brusilovsky P., Yudelson M. Ontology-based integration of adaptive educational systems // Taipei, Taiwan: 16th International Conference on Computers in Education (ICCE 2008) — 2008. — P. 11–18.
406. Sowa J. Towards the expressive power of natural language. Principles of Semantic Networks. Sowa J. (eds.), Ch. 5, Morgan Kaufman. — 1991. — P. 157–189.
407. Sowa J. Conceptual Graphs as a universal knowledge representation // Computers and Mathematics with Applications 23 (2–5). — 1992. — 75–93.
408. Sowa J. Knowledge Representation: Logical, Philosophical, and Computational Foundations. — New York : Brooks/Cole, 2000.
409. Sowa, J. Conceptual Graphs for a Data Base Interface // IBM Journal of Research and Development 20 (4). — 1976. — P. 336–357.
410. Sowa, J. Conceptual Structures: information processing in mind and machine.— Cambridge, MA : Addison Wesley, 1984.
411. Sperry R. Mind-brain interaction: mentalism, yes; dualism, no. In: A.D. Smith, R. Llanas and P.G. Kostyuk (eds.). Commentaries in the Neurosciences. — Oxford : Pergamon Press, — 1980. — P. 651–662.
412. Stader J. Results of the Enterprise Project. Proceedings of Expert Systems '96, the 16th Annual Conference of the British Computer Society Specialist Group on Expert Systems, Cambridge, UK, December. — 1996.
413. Stephanopoulos G., Han C. Intelligent systems in process engineering: A review // Computers & Chemical Engineering, 20(6). — 1996. — P. 743–791.
414. Stewart D. Building enterprise taxonomies. — Mokita Press, 2011.
415. Stottler. History of Artificial Intelligence. — 2002.
416. Studer R., Benjamins R., Fensel D. Knowledge Engineering: Principles and Methods // Data and Knowledge Engineering 25(1–2). — 1998. — P. 161–197.
417. Studer R. Knowledge management course. Karlsruhe Institute of Technology. — 2008.
418. Suárez-Figueroa M. C., Gómez-Pérez A., Motta E., Gangemi A. (Eds.). Ontology engineering in a networked world. — Springer Science & Business Media, 2012.
419. Sure Y., Bloehdorn S., Haase P., Hartmann J. & Oberle D. The SWRC ontology-semantic web for research communities. Progress in Artificial Intelligence. — Springer Berlin Heidelberg, 2005. — P. 218–231.
420. Tempich C., E. Simperl M. Luczak R. Studer H. S. Pinto. Argumentation-based ontology engineering, IEEE Intelligent Systems 22 (6). — 2007. — P. 52–59.
421. Tempich C., Sofia Pinto H., Sure Y., Staab S. An Argumentation Ontology for DIstributed, Loosely-controlled and evolvInG Engineering processes of oNTologies (DILIGENT), in: The

Semantic Web: Research and Applications, Second European Semantic Web Conference (ESWC). — 2005. — P. 241–256.
422. Tham K. D., Fox M. S., Gruninger M. A cost ontology for enterprise modelling. Enabling Technologies: Infrastructure for Collaborative Enterprises. Proceedings of the Third Workshop on IEEE. — 1994. — P. 197–210.
423. Turban E., Aronson J. Decision support systems & intelligent systems, (7th Ed.) Editorial Prentice Hall. — 2005.
424. Tuthill G. S. Knowledge Engineering: Concepts and Practices for Knowledge Base Systems. Tab Books.T. — 1990.
425. Tzou H. & Anderson G. L. (Eds.). Intelligent structural systems (Vol. 13). — Springer Science & Business Media, 2013.
426. Uschold M., Jasper R. Framework for Understanding and Classifying Ontology Applications. Proceedings of the IJCAI-99 workshop on Ontologies and Problem-SolvingMethods (KRR5) Stockholm, Sweden, August 2. — 1999.
427. Uschold M. et al. The enterprise ontology // The knowledge engineering review 13 (1). — 1998. — P. 31–89.
428. Varzi A. C. A note on the transitivity of parthood // Applied Ontology 1 (2). — 2006. — P. 141–146.
429. Varzi A. C. Basic Problems of Mereotopology. Formal Ontology in Information Systems, IOS. — 1998. — P. 29–38.
430. Varzi A. C. Parts, wholes, and part-whole relations: The prospects of mereotopology // Data & Knowledge Engineering 20 (3). — 1996. — P. 259–286.
431. Vernadat F. UEML: towards a unified enterprise modelling language // International Journal of Production Research 40 (17).— 2002. — P. 4309–4321.
432. Villano M., Kennedy M., et al. Relative efficacy of various strategies for visual feedback in standing balance activities // Experimental brain research 230.1. — 2013. — P. 117–125.
433. Walker T., Miller R. Expert System Handbook, Fairmont Press. — 1990.
434. Waterman D. A guide to Expert Systems. Pearson Education. — 2004.
435. Weizenbaum J. ELIZA — A Computer Program for the Study of Natural Language Communication between Man and Machine // Communications of the Association for Computing Machinery 9.— 1966. — P. 36–45.
436. Whelan E. It's who you know not what you know: a social network analysis approach to talent management // European Journal of International Management 5 (5). — 2011. — P. 484–500.
437. Wielinga B. et al. A Knowledge Acquisition Perspective on Knowledge-Level Models // Proceedings of the Fourth Knowledge Acquisition for Knowledge-Based Systems Workshop. Banff 36. — 1989. — P. 1–22.
438. Wilks Y. A. Grammar, meaning and the machine analysis of language. — London : Routledge & Kegan Paul, 1972.
439. Winston M. E., Chaffin R., Herrmann D. A taxonomy of part-whole relations // Cognitive science 11 (4). — 1987. — P. 417–444.
440. Woods W. What's in a link: foundations for semantic networks (D. G. Bobrow & A. Collins, eds). Representation and Understanding. — New York : Academic Press, 1975. — P. 35–82.
441. Wooldridge M. Introduction to MultiAgent Systems. Wiley. — 2002.

ОГЛАВЛЕНИЕ

Предисловие **3**
ЧАСТЬ I **5**
Глава 1. Интеллектуальные системы **6**
1.1. Искусственный интеллект: краткая история 6
1.2. Интеллектуальные системы: направления разработок 12
1.3. Системы, основанные на знаниях, и инженерия знаний 18
1.4. Об экспертных системах 20
1.5. Разработка интеллектуальных систем 27
1.6. Прикладные интеллектуальные системы 32
Глава 2. Получение знаний **35**
2.1. Знания и данные 35
2.2. Источники и способы получения знаний 37
2.3. Определение и структура инженерии знаний 43
2.4. Основные аспекты инженерии знаний 45
2.5. Классификация практических методов извлечения знаний 55
2.6. Интеллектуальный анализ данных 71
Глава 3. Структурирование знаний **75**
3.1. Иерархическое структурирование 75
3.2. Визуальные ментальные модели и их классификации 77
3.3. Типы знаний и виды диаграмм 82
3.4. Табличные методы структурирования знаний 97
3.5. Стратегии структурирования 100
3.6. О языках представления знаний 104
3.7. Метаданные 106
3.8. Виды информационно-поисковых языков 108
Глава 4. Представление знаний **116**
4.1. Таблицы решений и таблицы операторов 116
4.2. Продукционные системы и способы организации рассуждений 126
4.3. Семантические сети 140
4.4. Примеры моделирования выражений естественного языка 152
4.5. Фреймы и представление стереотипов 162
4.6. Объектно-ориентированная модель 177
ЧАСТЬ II **185**
Глава 5. Онтологии **186**
5.1. Потребность в онтологиях 186
5.2. Определение онтологии 190
5.3. Классификация и примеры онтологий 193
5.4. Типы отношений в онтологиях 204
5.5. Методы разработки онтологий 209
5.6. Прочие методы онтологического инжиниринга (отображение, визуализация, оценка) 220

5.7. Представление онтологий и языки семантического веба 228
Глава 6. Применение онтологий 237
6.1. Обзор способов/сценариев применения онтологий 237
6.2. Моделирование предприятий на основе онтологий 245
6.3. Применение онтологий для моделирования в государственном и городском управлении 263
Глава 7. Системы на основе онтологий 282
7.1. Использование онтологий в экспертных системах и системах поддержки принятия решений 282
7.2. Обзор интеллектуальных систем на основе знаний 283
7.3. Примеры интеллектуальных систем на основе онтологий 286
7.4. Средства построения интеллектуальных систем на основе онтологий 299
Заключение 305
Литература и интернет-ресурсы 306

Татьяна Альбертовна ГАВРИЛОВА,
Дмитрий Вячеславович КУДРЯВЦЕВ,
Дмитрий Ильич МУРОМЦЕВ

ИНЖЕНЕРИЯ ЗНАНИЙ.
Модели и методы
Учебник

Редакция физико-математической литературы
Выпускающие *Н. А. Крылова,*
Т. С. Симонова, О. В. Шилкова

ЛР № 065466 от 21.10.97
Гигиенический сертификат 78.01.10.953.П.1028
от 14.04.2016 г., выдан ЦГСЭН в СПб

Издательство «ЛАНЬ»
lan@lanbook.ru; www.lanbook.com
196105, Санкт-Петербург, пр. Ю. Гагарина, д. 1, лит. А.
Тел./факс: (812) 336-25-09, 412-92-72.
Бесплатный звонок по России: 8-800-700-40-71

ГДЕ КУПИТЬ

ДЛЯ ОРГАНИЗАЦИЙ:

Для того, чтобы заказать необходимые Вам книги, достаточно обратиться в любую из торговых компаний Издательского Дома «ЛАНЬ»:

по России и зарубежью
«ЛАНЬ-ТРЕЙД». 192029, Санкт-Петербург, ул. Крупской, 13
тел.: (812) 412-85-78, 412-14-45, 412-85-82; тел./факс: (812) 412-54-93
e-mail: trade@lanbook.ru; ICQ: 446-869-967
www.lanpbl.spb.ru/price.htm

в Москве и в Московской области
«ЛАНЬ-ПРЕСС». 109263, Москва, 7-я ул. Текстильщиков, д. 6/19
тел.: (499) 178-65-85; e-mail: lanpress@lanbook.ru

в Краснодаре и в Краснодарском крае
«ЛАНЬ-ЮГ». 350901, Краснодар, ул. Жлобы, д. 1/1
тел.: (861) 274-10-35; e-mail: lankrd98@mail.ru

ДЛЯ РОЗНИЧНЫХ ПОКУПАТЕЛЕЙ:

интернет-магазин
Издательство «Лань»: http://www.lanbook.com

магазин электронных книг
Global F5: http://globalf5.com/

Подписано в печать 21.06.16.
Бумага офсетная. Гарнитура Школьная. Формат 70×100 $^{1}/_{16}$.
Печать офсетная. Усл. п. л. 26,33. Тираж 200 экз.

Заказ № 117-16.

Отпечатано в полном соответствии
с качеством предоставленного оригинал-макета
в ПАО «Т8 Издательские Технологии».
109316, г. Москва, Волгоградский пр., д. 42, к. 5.