Раздел I. ВВЕДЕНИЕ В ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ

Создание экспертных систем значительно расширило область приложения вычислительной техники, позволив использовать ее для решения неформализованных задач. Различие между формализованными и неформализованными задачами обусловливается характером знаний, используемых для решения этих задач. Знания, которыми владеет специалист в какой-либо области, можно разделить на точные, выраженные формально, и неточные (неформальные). Точные знания формулируются в книгах и руководствах в виде общих, универсальных и строгих суждений, законов, формул, моделей, алгоритмов и т.д. Неформальные знания обычно не попадают в книги и руководства вследствие их субъективности и приблизительности. Знания этого рода являются результатом обобщения многолетнего опыта работы специалиста и представляют собой, как правило, множество эвристических приемов и правил. Они образуют то, что называют опытом и интуицией специалистов.

Задачи, решаемые на основе формальных знаний, называют формальными, а задачи, решаемые с помощью неточных знаний, - неформализованными задачами. Последние, возможно, и допускают формализацию, но точные методы их решения либо еще не получены, либо слишком сложны. В процессе деятельности в описательных областях (т.е. в науках и дисциплинах, не использующих математическую формализацию) специалисты решают, как правило, неформализованные задачи.

Экспертные системы предназначены для решения неформализованных задач на основе неточных знаний, представляющих опыт эксперта по решению задач в области приложения.

В отличие от традиционных программ, ориентированных на решение формальных задач, экспертные системы обладают следующими особенностями:

- алгоритм получения решения не известен заранее, а строится самими экспертными системами с помощью символьных рассуждений на основе знаний экспертов;
- полученные решения обосновываются системой, т.е. экспертная система "осознает" в терминах пользователя, как было получено решение;
- экспертные системы способны анализировать и объяснять свои действия и знания;
- экспертные системы могут приобретать новые знания и менять в соответствии с ними свое поведение;
- эксперты, вводящие знания в экспертные системы, могут не обладать навыками программирования.

Типичная статическая ЭС состоит из следующих основных компонентов (рис. 1.1):

- решателя (интерпретатора);
- рабочей памяти (РП), называемой также базой данных (БД);
- базы знаний (БЗ);
- компонентов приобретения знаний;
- объяснительного компонента;
- диалогового компонента.

База данных (рабочая память) предназначена для хранения исходных и промежуточных данных решаемой в текущий момент задачи. Этот термин совпадает по названию, но не по смыслу с термином, используемым в информационно-поисковых системах (ИПС) и системах управления базами данных (СУБД) для обозначения всех данных (в первую очередь долгосрочных), хранимых в системе.

База знаний (БЗ) в ЭС предназначена для хранения долгосрочных данных, описывающих рассматриваемую область (а не текущих данных), и правил, описывающих целесообразные преобразования данных этой области.

Решатель, используя исходные данные из рабочей памяти и знания из БЗ, формирует такую последовательность правил, которые, будучи примененными к исходным данным, приводят к решению задачи.

Компонент приобретения знаний автоматизирует процесс наполнения ЭС знаниями, осуществляемый пользователем-экспертом.

Объяснительный компонент объясняет, как система получила решение задачи (или почему она не получила решение) и какие знания она при этом использовала, что облегчает эксперту тестирование системы и повышает доверие пользователя к полученному результату.

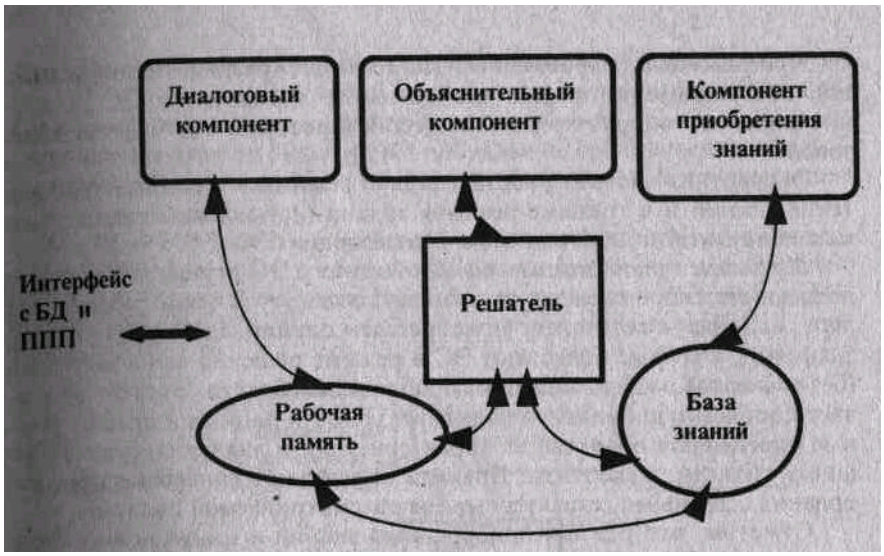


Рис. 1.1 Структура статической ЭС

Диалоговый компонент ориентирован на организацию дружественного общения с пользователем как в ходе решения задач, так и в процессе приобретения знаний и объяснения результатов работы.

Экспертная система работает в двух режимах: режиме приобретения знаний и в режиме решения задачи (называемом также режимом консультации или режимом использования ЭС).

В режиме приобретения знаний общение с ЭС осуществляет (через посредничество инженера по знаниям) эксперт. В этом режиме эксперт, используя компонент приобретения знаний, наполняет систему знаниями, которые позволяют ЭС в режиме решения самостоятельно (без эксперта)

решать задачи из проблемной области. Эксперт описывает проблемную область в виде совокупности данных и правил. Данные определяют объекты, их характеристики и значения, существующие в области экспертизы. Правила определяют способы манипулирования с данными, характерные для рассматриваемой области.

Отметим, что режиму приобретения знаний в традиционном подходе к разработке программ соответствуют этапы алгоритмизации, программирования и отладки, выполняемые программистом. Таким образом, в отличие от традиционного подхода в случае ЭС разработку программ осуществляет не программист, а эксперт (с помощью ЭС), не владеющий программированием.

В режиме консультации общение с ЭС осуществляет конечный пользователь, которого интересует результат и (или) способ его получения. Необходимо отметить, что в зависимости от назначения ЭС пользователь может не быть специалистом в данной проблемной области (в этом случае он обращается к ЭС за результатом, не умея получить его сам), или быть специалистом (в этом случае пользователь может сам получить результат, но он обращается к ЭС с целью либо ускорить процесс получения результата, либо возложить на ЭС рутинную работу). Следует подчеркнуть, что термин "пользователь" является многозначным, так как использовать ЭС кроме конечного пользователя может и эксперт, и инженер по знаниям, и программист. Поэтому когда хотят подчеркнуть, что речь идет о том, для кого делалась ЭС, используют термин "конечный пользователь".

В режиме консультации данные о задаче пользователя после обработки их диалоговым компонентом поступают в рабочую память. Решатель на основе входных данных из рабочей памяти, общих данных о проблемной области и правил из БЗ формирует решение задачи. Подчеркнем, что в отличие от традиционных программ ЭС при решении задачи не только исполняет предписанную последовательность операции, но и предварительно формирует ее. Если реакция системы не понятна пользователю, то он может

потребовать объяснения "Почему система задает тот или иной вопрос?", "как ответ, собираемый системой, получен?".

Структуру, приведенную на рис. 1.1, называют структурой статической ЭС. ЭС данного типа используются в тех приложениях, где можно не учитывать изменения окружающего мира, происходящие за время решения задачи. Первые ЭС, получившие практическое использование, были статическими. Они нашли применение в широком классе приложений.

Из общих соображений понятно, что существует огромный класс приложений, в которых требуется учитывать динамику, т. е. изменения, происходящие в окружающем мире за время исполнения приложения. На рис. 1.2 показано, что в архитектуру динамической ЭС по сравнению со статической ЭС вводятся два компонента: подсистема моделирования внешнего мира и подсистема связи с внешним окружением. Последняя осуществляет связи с внешним миром через систему датчиков и контроллеров.

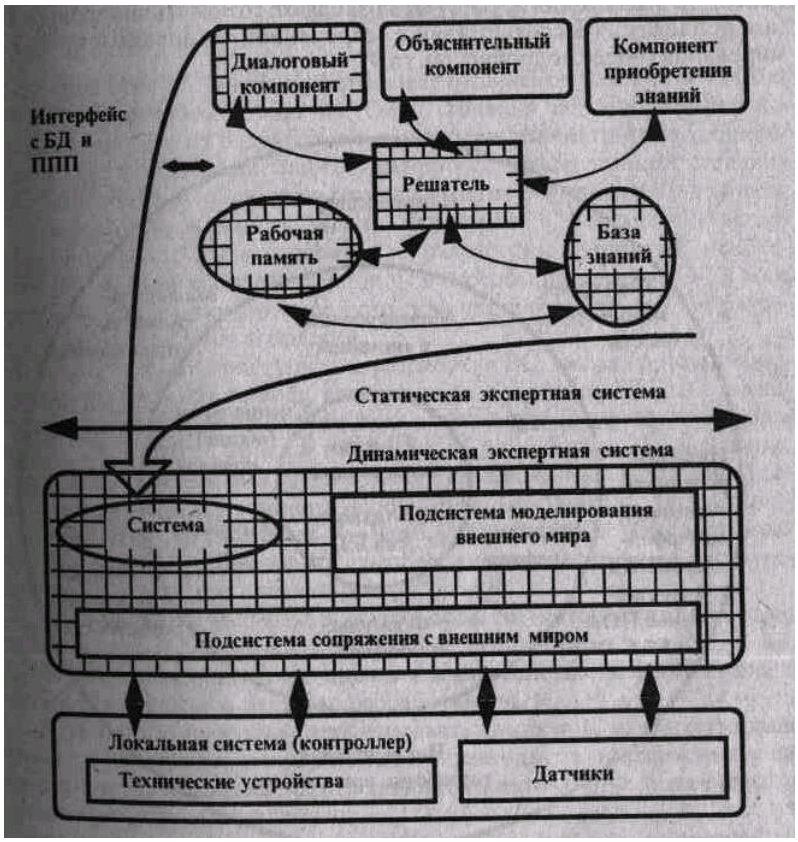


Рис. 1.2. Архитектура статических и динамических ЭС (компоненты, подвергающиеся изменениям, заштрихованы)

Подчеркнем, что структура ЭС, представленная на рис. 1.1 и 1.2 отражает только компоненты (функции), и многое остается "за кадром". На рис. 1.3 приведена обобщенная структура современной интеллектуальной системы (ИС) для создания динамических ЭС, содержащая кроме основных компонентов те возможности, которые позволяют создавать интегрированные приложения в соответствии с современной технологией программирования.

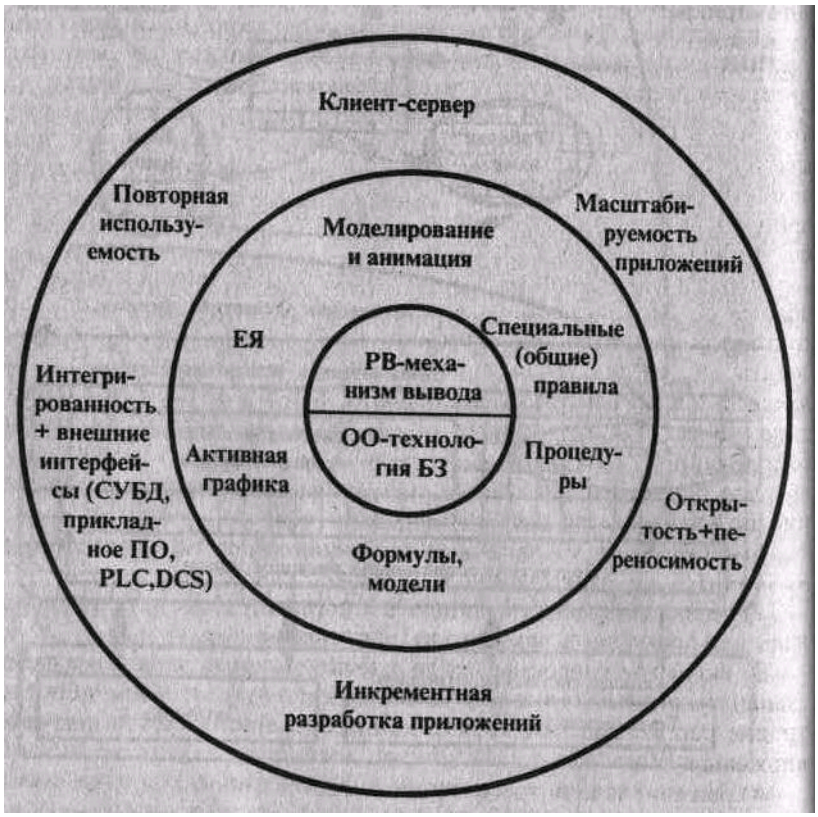


Рис. 1.3. Структура современных ИС для ЭС 18

В основе ИС лежат объектно-ориентированная база знаний и механизм вывода, способный оперировать с правилами в которых явным образом отражено время. Во внутреннем кольце расположены компоненты, обеспечивающие моделирование, анимацию, активную графику, механизм общих правил и т.д. Во внешнем кольце отражены технологии и требования, обязательные в современных ИС для создания ЭС.

Раздел II. ТЕХНОЛОГИЯ ПРОЕКТИРОВАНИЯ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

Тема 2.1. Представление знаний и вывод на знаниях в интеллектуальных системах

При изучении интеллектуальных систем традиционно возникает вопрос — что же такое знания и чем они отличаются от обычных данных, десятилетиями обрабатываемых ЭВМ. Можно предложить несколько рабочих определений, в рамках которых это становится очевидным.

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

При обработке на ЭВМ данные трансформируются, условно проходя следующие этапы:

1. D1 — данные как результат измерений и наблюдений;

2. D2 — данные на материальных носителях информации (таблицы, протоколы, справочники);

3. D3 — модели (структуры) данных в виде диаграмм, графиков, функций;

4. D4 — данные в компьютере на языке описания данных;

5. D5 — базы данных на машинных носителях информации.

Знания основаны на данных, полученных эмпирическим путем. Они представляют собой результат мыслительной деятельности человека, направленной на обобщение его опыта, полученного в результате практической деятельности.

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.

При обработке на ЭВМ знания трансформируются аналогично данным.

1. Z1 — знания в памяти человека как результат мышления;

2. Z2 — материальные носители знаний (учебники, методические пособия);

3. Z3 — *поле знаний* — условное описание основных объектов предметной области, их атрибутов и закономерностей, их связывающих;

4. Z4 — знания, описанные на языках представления знаний (продукционные языки, семантические сети, фреймы — см. далее);

5. Z5 — *база знаний на машинных носителях информации.* Часто используется такое определение 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. *Интенсионол* понятия — это определение его через соотнесение с понятием более высокого уровня абстракции с указанием специфических свойств. Интенсионалы формулируют знания об объектах. Другой способ определяет понятие через соотнесение с понятиями более низкого уровня абстракции или перечисление фактов, относящихся к определяемому объекту. Это есть определение через данные, или *экстенсионол понятия.*

Пример. Понятие «персональный компьютер».

Его интенсионал: *«Персональный компьютер — это дружественная ЭВМ, которую можно поставить на стол и купить менее чем за $2000-3000»*

Экстенсионал этого понятия: *«Персональный компьютер — это Mac, IBM PC, Sinkler »*

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 следующим категориям:

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

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

Современные экспертные системы работают в основном с поверхностными знаниями. Это связано с тем, что на данный момент нет универсальных методик, позволяющих выявлять глубинные структуры знаний и работать с ними.

Кроме того, в учебниках по искусственному интеллекту знания традиционно делят на *процедурные* и *декларативные.* Исторически первичными были процедурные знания, то есть знания, «растворенные» в алгоритмах. Они управляли данными. Для их изменения требовалось изменять программы. Однако с развитием искусственного интеллекта приоритет данных постепенно изменялся, и все большая часть знаний сосредоточивалась в структурах данных (таблицы, списки, абстрактные типы данных), то есть увеличивалась роль декларативных знаний.

Сегодня знания приобрели чисто декларативную форму, то есть знаниями считаются предложения, записанные на языках представления знаний, приближенных к естественному и понятных неспециалистам.

Модели представления знаний

Существуют десятки моделей (или языков) представления знаний для различных предметных областей. Большинство из них может быть сведено к следующим классам:

• продукционные модели;

• семантические сети;

• фреймы;

• формальные логические модели.

Продукционная модель

Продукционная модель или модель, основанная на правилах, позволяет представить знания в виде предложений типа «Если (условие), то (действие)».

Под «условием» *(антецедентом)* понимается некоторое предложение-образец, по которому осуществляется поиск в базе знаний, а под «действием» *(консеквентом)* — действия, выполняемые при успешном исходе поиска (они могут быть промежуточными, выступающими далее как условия и терминальными или целевыми, завершающими работу системы).

Чаще всего вывод на такой базе знаний бывает *прямой* (от данных к поиску цели) или *обратный* (от цели для ее подтверждения — к данным). Данные — это исходные факты, хранящиеся в базе фактов, на основании которых запускается машина вывода или интерпретатор правил, перебирающий правила из продукционной базы знаний.

Продукционная модель чаще всего применяется в промышленных экспертных системах. Она привлекает разработчиков своей наглядностью, высокой модульностью, легкостью внесения дополнений и изменений и простотой механизма логического вывода.

Семантические сети

Термин *семантическая* означает «смысловая», а сама семантика — это наука, устанавливающая отношения между символами и объектами, которые они обозначают, то есть наука, определяющая смысл знаков.

Семантическая сеть — это ориентированный граф, вершины которого — понятия, а дуги — отношения между ними.

В качестве понятий обычно выступают абстрактные или конкретные объекты, а *отношения* — это связи типа: «это» («АКО — A-Kind-Of», «is»), «имеет частью» («has part»), «принадлежит», «любит». Характерной особенностью семантических сетей является обязательное наличие трех типов отношений:

• **класс** — элемент класса (цветок — роза);

• **свойство** — значение (цвет — желтый);

• **пример** элемента класса (роза — чайная).

Можно предложить несколько классификаций семантических сетей, связанных с типами отношений между понятиями.

По количеству типов отношений:

• Однородные (с единственным типом отношений).

• Неоднородные (с различными типами отношений).

По типам отношений:

• Бинарные (в которых отношения связывают два объекта).

• N-арные (в которых есть специальные отношения, связывающие более двух понятий).

Наиболее часто в семантических сетях используются следующие отношения:

• связи типа *часть — целое* («класс — *подкласса, 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);

• лингвистические связи и др.

Проблема поиска решения в базе знаний типа семантической сети сводится к задаче поиска фрагмента сети, соответствующего некоторой подсети, отражающей поставленный запрос к базе.

Фреймы

Термин *фрейм* (от английского frame, что означает «каркас» или «рамка») был предложен *Маренном Минским*], одним из пионеров ИИ, в 70-е годы для обозначения структуры знаний для восприятия пространственных сцен. Эта модель, как и семантическая сеть, имеет глубокое психологическое обоснование.

Фрейм — это абстрактный образ для представления некоего стереотипа восприятия.

В психологии и философии известно понятие абстрактного образа. Например, произнесение вслух слова «комната» порождает у слушающих образ комнаты:

«жилое помещение с четырьмя стенами, полом, потолком, окнами и дверью, площадью 6-20 $м^2$». Из этого описания ничего нельзя убрать (например, убрав окна, мы получим уже чулан, а не комнату), но в нем есть «дырки» или «слоты» — это незаполненные значения некоторых атрибутов — например, количество окон, цвет стен, высота потолка, покрытие пола и др.

В теории фреймов такой образ комнаты называется фреймом комнаты. Фреймом также называется и формализованная модель для отображения образа.

Различают *фреймы-образцы,* или *прототипы,* хранящиеся в базе знаний, и *фреймы-экземпляры,* которые создаются для отображения реальных фактических ситуаций на основе поступающих данных. Модель фрейма является достаточно универсальной, поскольку позволяет отобразить все многообразие знаний о мире через:

• *фреймы-структуры,* использующиеся для обозначения объектов и понятий (заем, залог, вексель);

• *фреймы-роли* (менеджер, кассир, клиент);

• *фреймы-сценарии* (банкротство, собрание акционеров, празднование именин);

• *фреймы-ситуации* (тревога, авария, рабочий режим устройства) и др.

Важнейшим свойством теории фреймов является заимствование из теории семантических сетей — так называемое *наследование свойств.* И во фреймах, и в семантических сетях наследование происходит по *АКО-связям (A-Kind-Of ~ это).* Слот АКО указывает на фрейм более высокого уровня иерархии, откуда неявно наследуются, то есть переносятся, значения аналогичных слотов.

Основным преимуществом фреймов как модели представления знаний является то, что она отражает концептуальную основу организации памяти человека, а также ее гибкость и наглядность.

Формальные логические модели

Традиционно в представлении знаний выделяют *формальные логические модели, основанные на классическом* исчислении предикатов 1-го порядка, когда предметная область или задача описывается в виде набора аксиом. Исчисление предикатов 1-го порядка в промышленных экспертных системах практически не используется. Эта логическая модель применима в основном в исследовательских «игрушечных» системах, так как предъявляет очень высокие требования и ограничения к предметной области.

В промышленных же экспертных системах используются различные ее модификации и расширения.

Вывод на знаниях

Несмотря на все недостатки, наибольшее распространение получила продукционная модель представления знаний. При использовании продукционной модели база знаний состоит из набора правил. Программа, управляющая перебором правил, называется *машиной вывода*.

Машина вывода

Машина вывода (интерпретатор правил) выполняет две функции: во-первых, просмотр существующих фактов из рабочей памяти (базы данных) и правил из базы знаний и добавление (по мере возможности) в рабочую память новых фактов и, во-вторых, определение порядка просмотра и применения правил. Этот механизм управляет процессом консультации, сохраняя для пользователя информацию о полученных заключениях, и запрашивает у него информацию, когда для срабатывания очередного правила в рабочей памяти оказывается недостаточно данных .

В подавляющем большинстве систем, основанных на знаниях, механизм вывода представляет собой небольшую по объему программу и включает два компонента — один реализует собственно вывод, другой управляет этим процессом. Действие *компонента вывода* основано на применении правила, называемого *modus ponens*.

Правило modus ponens. Если известно, что истинно утверждение А и существует правило вида «ЕСЛИ А, ТО В", тогда утверждение В также истинно.

Правила срабатывают, когда находятся факты, удовлетворяющие их левой части: если истинна посылка, то должно быть истинно и заключение.

Компонент вывода должен функционировать даже при недостатке информации. Полученное решение может и не быть точным, однако система не должна останавливаться из-за того, что отсутствует какая-либо часть входной информации.

Управляющий компонент определяет порядок применения правил и выполняет четыре функции:

1. *Сопоставление* — образец правила сопоставляется с имеющимися фактами.

2. *Выбор* — если в конкретной ситуации может быть применено сразу несколько правил, то из них выбирается одно, наиболее подходящее по заданному критерию (разрешение конфликта).

3. *Срабатывание* — если образец правила при сопоставлении совпал с какими-либо фактами из рабочей памяти, то правило срабатывает.

4. *Действие* — рабочая память подвергается изменению путем добавления в нее заключения сработавшего правила. Если в правой части правила содержится указание на какое-либо действием-то оно выполняется (как, например, в системах обеспечения безопасности информации).

Интерпретатор продукций работает циклически. В каждом цикле он просматривает все правила, чтобы выявить те, посылки которых совпадают с известными на данный момент фактами из рабочей памяти. После выбора правило срабатывает, его заключение заносится в рабочую память, и затем цикл повторяется сначала.

В одном цикле может сработать только одно правило. Если несколько правил успешно сопоставлены с фактами, то интерпретатор производит выбор по определенному критерию единственного правила, которое срабатывает в данном цикле.

Информация из рабочей памяти последовательно сопоставляется с посылками правил для выявления успешного сопоставления. Совокупность отобранных правил составляет так называемое *конфликтное множество.* Для разрешения конфликта интерпретатор имеет критерий, с помощью которого он выбирает единственное правило, после чего оно срабатывает. Это выражается в занесении фактов, образующих заключение правила, в рабочую память или в изменении критерия выбора конфликтующих правил. Если же в заключение правила входит название какого-нибудь действия, то

оно выполняется. Работа машины вывода зависит только от состояния рабочей памяти и от состава базы знаний. На практике обычно учитывается история работы, то есть поведение механизма вывода в предшествующих циклах. Информация о поведении механизма вывода запоминается в памяти состояний. Обычно память состояний содержит протокол системы.

Стратегии управления выводом

От выбранного метода поиска, то есть стратегии вывода, будет зависеть порядок применения и срабатывания правил. Процедура выбора сводится к определению направления поиска и способа его осуществления. Процедуры, реализующие поиск, обычно «зашиты» в механизм вывода, поэтому в большинстве систем инженеры знаний не имеют к ним доступа и, следовательно, не могут в них ничего изменять по своему желанию. При разработке стратегии управления выводом важно определить два вопроса:

1. Какую точку в пространстве состояний принять в качестве исходной? От выбора этой точки зависит и метод осуществления поиска — в прямом или обратном направлении.

2. Какими методами можно повысить эффективность поиска решения? Эти методы определяются выбранной стратегией перебора — глубину, в ширину, по подзадачам или иначе.

Прямой и обратный вывод

При обратном порядке вывода вначале выдвигается некоторая гипотеза, а затем механизм вывода как бы возвращается назад, переходя к фактам, пытаясь найти те**,** которые подтверждают гипотезу. Если она оказалась правильной, то выбирается следующая гипотеза, детализирующая первую и являющаяся по отношению к ней подцелью. Далее отыскиваются факты, подтверждающие истинность подчиненной гипотезы. Вывод такого типа называется управляемым целями, или управляемым консеквентами.

Обратный поиск применяется в тех случаях, когда цели известны и их сравнительно немного.

В системах с прямым выводом по известным фактам отыскивается заключение, которое из этих фактов следует. Если такое заключение удается найти, то оно заносится в рабочую память. Прямой вывод часто называют выводом, управляемым данными, или выводом, управляемым антецедентами.

Существуют системы, в которых вывод основывается на сочетании упомянутых выше методов — обратного и ограниченного прямого. Такой комбинированный метод получил название циклического.

Пример. Имеется фрагмент базы знаний из двух правил:

П1. Если «отдых — летом» и «человек — активный», то «ехать в горы».

П2. Если «любит солнце», то «отдых летом».

Предположим, в систему поступили факты — «человек активный» и «любит солнце».

ПРЯМОЙ ВЫВОД — исходя из фактических данных, получить рекомендацию.

1-й проход.

Шаг 1. Пробуем *П1,* не работает (не хватает данных «отдых — летом»).

Шаг 2. Пробуем *П2,* работает, в базу поступает факт «отдых — летом».

2-й проход.

Шаг 3. Пробуем *П1,* работает, активируется цель «ехать в горы», которая и выступает как совет, который дает ЭС.

ОБРАТНЫЙ ВЫВОД — подтвердить выбранную цель при помощи имеющихся правил и данных.

1-й проход.

Шаг 1. Цель — «ехать в горы»: пробуем *П1* — данных «отдых — летом» нет, они становятся новой целью и ищется правило, где цель в левой части.

Шаг 2. Цель «отдых — летом»: правило *П2* подтверждает цель и активирует ее.

2-й проход.

Шаг 3. Пробуем П 1, подтверждается искомая цель.

Методы поиска в глубину и ширину

В системах, база знаний которых насчитывает сотни правил, желательным является использование стратегии управления выводом, позволяющей минимизировать время поиска решения и тем самым повысить эффективность вывода. К числу таких стратегий относятся: поиск в глубину, поиск в ширину, разбиение на подзадачи и альфа-бета алгоритм.

При поиске в глубину в качестве очередной подцели выбирается та, которая соответствует следующему, более детальному уровню описания задачи. Например, диагностирующая система, сделав на основе известных симптомов предположение о наличии определенного заболевания, будет продолжать запрашивать уточняющие признаки и симптомы этой болезни до тех пор, пока полностью не опровергнет выдвинутую гипотезу.

При поиске в ширину, напротив, система вначале проанализирует все симптомы, находящиеся на одном уровне пространства состояний, даже если они относятся к разным заболеваниям, и лишь затем перейдет к симптомам следующего уровня детальности.

Разбиение на подзадачи — подразумевает выделение подзадач, решение которых рассматривается как достижение промежуточных целей на пути к конечной цели. Примером, подтверждающим эффективность разбиения на подзадачи, является поиск неисправностей в компьютере — сначала выявляется отказавшая подсистема (питание, память и т. д.), что значительно сужает пространство поиска. Если удается правильно понять сущность задачи и оптимально разбить ее на систему иерархически связанных целей-подцелей, то можно добиться того**,** что путь к ее решению в пространстве поиска будет минимален.

Альфа-бета алгоритм позволяет уменьшить пространство состояний путем удаления ветвей, неперспективных для успешного поиска. Поэтому

просматриваются только те вершины, в которые можно попасть в результате следующего шага, после чего неперспективные направления исключаются. Альфа-бета алгоритм нашел широкое применение в основном в системах, ориентированных на различные игры, например в шахматных программах.

Тема 2.2. Проблемы разработки промышленных интеллектуальных систем

Разработка программных комплексов экспертных систем как за рубежом, так и в нашей стране находится на уровне скорее искусства, чем науки. Это связано с тем, что долгое время системы искусственного интеллекта внедрялись в основном во время фазы проектирования, а чаще всего разрабатывалось несколько прототипных версий программ, и на их основе уже создавался конечный продукт. Такой подход действует хорошо в исследовательских условиях, однако в коммерческих условиях он является слишком дорогим, чтобы оправдать затраты на разработку.

Процесс разработки промышленной экспертной системы, опираясь на традиционные технологии, практически для любой предметной области можно разделить на шесть более или менее независимых этапов:

- Выбор проблемы
- Разработка прототипа ЭС
- Доработка до промышленной ЭС
- Оценка ЭС
- Стыковка ЭС
- Поддержка 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 может 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 расходы на разработку таких систем очень большие, время разработки слишком велико, а полученные в результате программы накладывают тяжелое бремя на вычислительные ресурсы. В целом за разработку экспертных систем целесообразно браться организации, где накоплен опыт по автоматизации рутинных процедур обработки информации, таких как:

• формирование корпоративных информационных систем;

• организация сложных расчетов;

• работа с компьютерной графикой;

• обработка текстов и автоматизированный документооборот.

Решение таких задач, во-первых, подготавливает высококвалифицированных специалистов по информатике, необходимых для создания интеллектуальных систем, во-вторых, позволяет отделить от экспертных систем неэкспертные задачи.

Выбор подходящей проблемы

Этот этап определяет деятельность, предшествующую решению начать разрабатывать конкретную ЭС. Он включает:

• определение проблемной области и задачи;

• нахождение эксперта, желающего сотрудничать при решении проблемы, и назначение коллектива разработчиков;

• определение предварительного подхода к решению проблемы;

• анализ расходов и прибылей от разработки;

• подготовку подробного плана разработки.

Правильный выбор проблемы представляет самую критическую часть разработки в целом. Если выбрать неподходящую проблему, можно очень быстро увязнуть в «болоте» проектирования задач, которые никто не знает, как решать. Неподходящая проблема может также привести к созданию

экспертной системы, которая стоит намного больше, чем экономит. Дело будет обстоять еще хуже, если разработать систему, которая работает, но неприемлема для пользователей. Даже если разработка выполняется самой организацией для собственных целей, эта фаза является подходящим моментом для получения рекомендаций извне, чтобы гарантировать удачно выбранный и осуществимый с технической точки зрения первоначальный проект.

При выборе области применения следует учитывать, что если знание, необходимое для решения задач, постоянное, четко формулируемое и связано с вычислительной обработкой, то обычные алгоритмические программы, по всей вероятности, будут самым целесообразным способом решения проблем в этой области.

Экспертная система ни в коем случае не устранит потребность в реляционных базах данных, статистическом программном обеспечении, электронных таблицах и системах текстовой обработки. Но если результативность задачи зависит от знания, которое является субъективным, изменяющимся, символьным или вытекающим частично из соображений здравого смысла, тогда область может обоснованно выступать претендентом на экспертную систему. Обычно 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. 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.

Во время первоначальных бесед они должны решить, будет ли их сотрудничество успешным. Это немаловажно, поскольку обе стороны будут работать совместно, по меньшей мере в течение одного года. Кроме них в коллектив разработчиков целесообразно включить потенциальных

пользователей и профессиональных программистов. Подробно функции каждого члена коллектива описаны в следующем параграфе.

Предварительный подход к программной реализации задачи определяется, исходя из характеристик задачи и ресурсов, выделенных на ее решение. Инженер по знаниям выдвигает обычно несколько вариантов, связанных с использованием имеющихся на рынке программных средств. Окончательный выбор возможен лишь на этапе разработки прототипа.

После того как задача определена, необходимо подсчитать расходы и прибыль от разработки экспертной системы. В расходы включаются затраты на оплату труда коллектива разработчиков. В дополнительные расходы будет включена стоимость приобретаемого программного инструментария, с помощью которого будет разработана экспертная система.

Прибыль может быть получена за счет снижения цены продукции, повышения производительности труда, расширения номенклатуры продукции или услуг или даже разработки новых видов продукции или услуг в области, в которой будет использоваться ЭС. Соответствующие расходы и прибыль от системы определяются относительно времени, в течение которого возвращаются средства, вложенные в разработку. На современном этапе большая часть фирм, развивающих крупные экспертные системы, предпочли разрабатывать дорогостоящие проекты, приносящие значительную прибыль.

Можно ожидать развития тенденции разработки менее дорогостоящих систем, хотя и с более длительным сроком окупаемости вложенных в них средств, так как программные средства разработки экспертных систем непрерывно совершенствуются. После того как инженер по знаниям убедился, что:

• 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,

он составляет план разработки. План определяет шаги процесса разработки и необходимые затраты, а также ожидаемые результаты.

Развитие прототипа до промышленной 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 необходимо изменить. Можно сделать оценку общего числа эвристических правил, необходимых для создания окончательного варианта экспертной системы. Иногда при разработке промышленной и/ или коммерческой системы выделяют дополнительные этапы для перехода .

демонстрационный прототип —> действующий прототип —> промышленная система —> коммерческая система

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 «промышленная ЭС» (в этой работе).

Основная работа на данном этапе заключается в существенном расширении базы знаний, то есть в добавлении большого числа дополнительных правил, фреймов, узлов семантической сети или других элементов знаний. Эти элементы знаний обычно увеличивают *глубину* системы, обеспечивая большее число правил для трудно уловимых аспектов

отдельных случаев. В то же время эксперт и инженер по знаниям могут увеличить базу знаний системы, включая правила, управляющие дополнительными подзадачами или дополнительными аспектами экспертной задачи (метазнания).

Переход от прототипа к промышленной экспертной системе приведен в следующей таблице:

Система	Описание
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, то есть хорошо документирована и снабжена сервисом .

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. Необходимо обратить особое внимание на языковые возможности интерфейсов, их простоту и удобство для управления работой ЭС. Система должна обеспечивать пользователю возможность легким и естественным образом

уточнять непонятные моменты, приостанавливать работу и т. д. В частности, могут оказаться полезными графические представления.

Оценка системы

После завершения этапа разработки промышленной экспертной системы необходимо провести ее тестирование в отношении критериев эффективности. К тестированию широко привлекаются другие эксперты с целью апробирования работоспособности системы на различных примерах. Экспертные системы оцениваются главным образом для того, чтобы проверить точность работы программы и ее полезность. Оценку можно проводить, исходя из различных критериев, которые сгруппируем 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.).

Стыковка системы

На этом этапе осуществляется стыковка экспертной системы с другими программными средствами в среде, в которой она будет работать, и обучение людей, которых она будет обслуживать. Иногда это означает внесение существенных изменений. Такие изменения требуют непременного

вмешательства инженера по знаниям или какого-либо другого специалиста, который сможет модифицировать систему. Под стыковкой подразумевается также разработка связей между экспертной системой и средой, в которой она действует.

Когда экспертная система уже готова, инженер по знаниям должен убедиться в том, что эксперты, пользователи и персонал знают, как эксплуатировать и обслуживать ее. После передачи им своего опыта в области информационной технологии инженер по знаниям может полностью предоставить ее в распоряжение пользователей.

Для подтверждения полезности системы важно предоставить каждому из пользователей возможность поставить перед ЭС реальные задачи, а затем проследить, как она выполняет эти задачи. Для того чтобы система была одобрена, необходимо представить ее как помощника, освобождающего пользователей от обременительных задач, а не как средство их замещения.

Стыковка включает обеспечение связи ЭС с существующими базами данных и Другими системами на предприятии, а также улучшение системных факторов, зависящих от времени, чтобы можно было обеспечить ее более эффективную работу и улучшить характеристики ее технических средств, если система работает в необычной среде (например, связь с измерительными устройствами).

Поддержка системы

При перекодировании системы на язык, подобный С, повышается ее быстродействие и увеличивается переносимость, однако гибкость при этом уменьшается. Это приемлемо лишь в том случае, если система сохраняет все знания проблемной области и это знание не будет изменяться в ближайшем будущем. Однако если экспертная система создана именно из-за того, что проблемная область изменяется, то необходимо поддерживать систему в ее инструментальной среде разработки.

Тема 2.3. Основы методологии разработки интеллектуальных систем

Разработка ЭС имеет существенные отличия от разработки обычного программного продукта. Опыт создания ЭС показал, что использование при их разработке методологии, принятой в традиционном программировании, либо чрезмерно затягивает процесс создания ЭС, либо вообще приводит к отрицательному результату. Дело в том, что неформализованность задач, решаемых ЭС, отсутствие завершенной теории ЭС и методологии их разработки приводят к необходимости модифицировать принципы и способы построения ЭС в ходе процесса разработки по мере того, как увеличивается знание, разработчиков о проблемной области.

Перед тем как приступить" к разработке ЭС, инженер по знаниям должен рассмотреть вопрос, следует ли разрабатывать ЭС для данного приложения. В обобщенном виде ответ может быть таким: использовать ЭС следует только тогда, когда разработка ЭС возможна, оправдана и методы инженерии знаний соответствуют решаемой задаче. Ниже будут уточнены использованные понятия "возможно", "оправдано", "соответствует". Чтобы разработка ЭС была возможной для данного приложения, необходимо одновременное выполнение по крайней мере следующих требований:

- существуют эксперты в данной области, которые решают задачу значительно лучше, чем начинающие специалисты;
- эксперты сходятся в оценке предлагаемого решения, иначе нельзя будет оценить качество разработанной ЭС;
- эксперты способны вербализовать (выразить на естественном языке) и объяснить используемые ими методы, в противном случае трудно рассчитывать на то, что знания экспертов будут "извлечены" и вложены в ЭС;
- решение задачи требует только рассуждений, а не действий;

- задача не должна быть слишком трудной (т.е. ее решение должно занимать у эксперта несколько часов или дней, а не недель);
- задача хотя и не должна быть выражена в формальном виде, но все же должна относиться к достаточно "понятной" и структурированной области, т.е. должны быть выделены основные понятия, отношения и известные (хотя бы эксперту) способы получения решении задачи;

Применение ЭС может быть оправдано одним из следующих факторов:

- решение задачи принесет значительный эффект, например экономический;
- использование человека-эксперта невозможно либо из-за недостаточного количества экспертов, либо из-за необходимости выполнять экспертизу одновременно в различных местах;
- использование ЭС целесообразно при необходимости решать задачу в окружении, враждебном для человека.

Приложение соответствует методам ЭС, если решаемая задача обладает совокупностью следующих характеристик:

- задача может быть естественным образом решена посредством манипуляции с символами (т.е. с помощью символических рассуждений), а не манипуляций с числами, как принято в математических методах и в традиционном программировании;
- задача должна иметь эвристическую, а не алгоритмическую природу, т.е. ее решение должно требовать применения эвристических правил. Задачи, которые могут быть гарантированно решены (с соблюдением заданных ограничений) с помощью некоторых формальных процедур, не подходят для применения ЭС;
- задача должна быть достаточно сложна, чтобы оправдать затраты на разработку ЭС. Однако она не должна быть чрезмерно сложной (решение занимает у эксперта часы, а не недели), чтобы ЭС могла ее решать;

- задача должна быть достаточно узкой, чтобы решаться методами ЭС, и практически значимой.

При разработке ЭС, как правило, используется концепция "быстрого прототипа". Суть этой концепции состоит в том, что разработчики не пытаются сразу построить конечный продукт. На начальном этапе они создают прототип (прототипы) ЭС. Прототипы должны удовлетворять двум противоречивым требованиям: с одной стороны они должны решать типичные задачи конкретного приложения, а с другой - время и трудоемкость их разработки должны быть весьма незначительны. Для удовлетворения указанным требованиям, как правило, при создании прототипа используются разнообразные средства, ускоряющие процесс проектирования.

Прототип должен продемонстрировать пригодность методов инженерии знаний для данного приложения. В случае успеха эксперт с помощью инженера по знаниям расширяет знания прототипа о проблемной области. При неудаче может потребоваться разработка нового прототипа или разработчики могут прийти к выводу о непригодности методов ЭС для данного приложения. По мере увеличения знаний прототип может достигнуть такого состояния, когда он успешно решает все задачи данного приложения. Преобразование прототипа ЭС в конечный продукт обычно приводит к перепрограммированию ЭС на языках низкого уровня, обеспечивающих как увеличение быстродействия ЭС, так и уменьшение требуемой памяти. Трудоемкость и время создания ЭС в значительной степени зависят' от типа используемого инструментария.

В ходе работ по созданию ЭС сложилась определенная технология их разработки, включающая шесть следующих этапов: идентификацию, концептуализацию, формализацию, выполнение, тестирование, опытную эксплуатацию. На этапе идентификации определяются задачи, которые подлежат решению, выявляются цели разработки, определяются эксперты и типы пользователей.

На этапе концептуализации проводится содержательный анализ проблемной области, выявляются используемые понятия и их взаимосвязи, определяются методы решения задач.

На этапе формализации выбираются ИС и определяются способы представления всех видов знаний, формализуются основные понятия, определяются способы интерпретации знаний, моделируется работа системы, оценивается адекватность целям системы зафиксированных понятий, методов решений, средств представления и манипулирования знаниями.

На этапе выполнения осуществляется наполнение экспертом базы знаний. В связи с тем, что основой ЭС являются знания, данный этап является наиболее важным и наиболее трудоемким этапом разработки. Процесс приобретения знаний разделяют на извлечение знаний из эксперта, организацию знаний, обеспечивающую эффективную работу системы, и представление знаний в виде, понятном ЭС. Процесс приобретения знаний осуществляется инженером по знаниям на основе анализа деятельности эксперта по решению реальных задач.

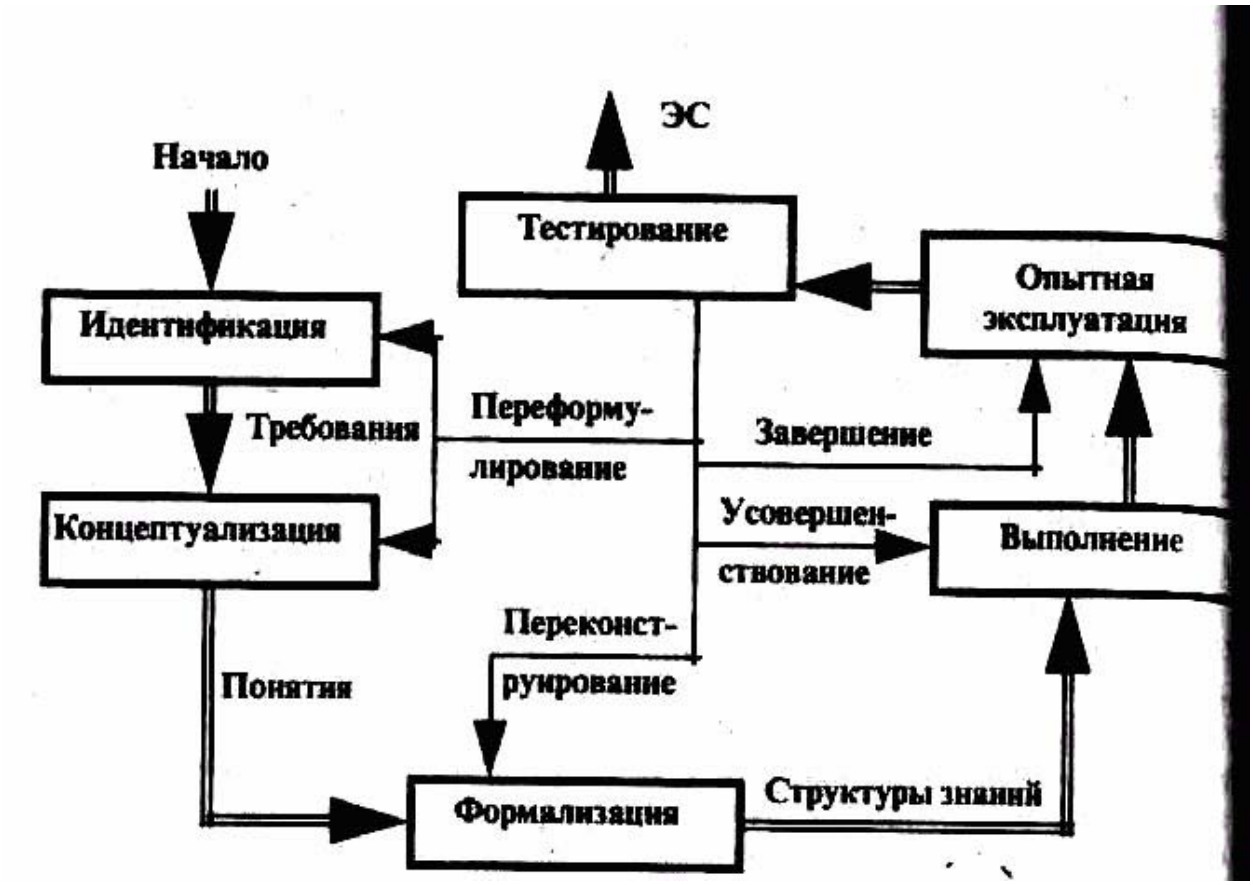


Рис. 2.1. Технология разработки ЭС

На этапе тестирования эксперт (и инженер по знаниям) в интерактивном режиме с использованием диалоговых и объяснительны! средств системы проверяет компетентность ЭС. Процесс тестировании продолжается до тех пор, пока эксперт не решит, что система достигла требуемого уровня компетентности.

На этапе опытной эксплуатации проверяется пригодность ЭС дли конечных пользователей. По результатам этого этапа может потребоваться существенная модификация ЭС. Процесс создания ЭС не сводится к строгой последовательности перечисленных выше этапов. В ходе разработки приходится неоднократно возвращаться на более ранние этапы и пересматривать принятые там решения.

На этапе идентификации определяются задачи, участники процесса разработки и их роли, ресурсы и цели. Определение участников и их ролей сводится к определению количества экспертов и инженеров по знаниям, а также формы их взаимоотношений. Обычно в основном цикле разработки ЭС участвуют не менее трех-четырех человек (один эксперт, один или два инженера по знаниям и один программист, привлекаемый для модификации и согласования инструментальных средств). К процессу разработки ЭС могут привлекаться и другие участники. Например, инженер по знаниям может привлекать других экспертов для того, чтобы убедиться в правильности своего понимания основного эксперта; представительности тестов, демонстрирующих особенности рассматриваемой задачи; совпадении взглядов различных экспертов на качество предлагаемых решений. Формы взаимоотношений экспертов и инженеров следующие: эксперт исполняет роль информирующего или эксперт выполняет роль учителя, а инженер - ученика. Несмотря на то, что основу знаний ЭС будут составлять знания эксперта, для достижения успеха инженер по знаниям должен использовать

дополнительные источники знаний в виде книг, инструкций, которые ему рекомендовал эксперт.

Идентификация задачи заключается в составлении неформального (вербального) описания решаемой задачи. В этом описании указываются общие характеристики задачи; подзадачи, выделяемые внутри данной задачи; ключевые понятия (объекты), характеристики и отношения; входные (выходные) данные; предположительный вид решения; знания, релевантные решаемой задаче; примеры (тесты) решения задачи.

Цель этапа идентификации задачи состоит в том, чтобы характеризовать задачу и структуру поддерживающих се знаний и приступить к работе по созданию базы знаний. Если исходная задача оказывается слишком сложной с учетом имеющихся ресурсов, то этап идентификации может потребовать нескольких итераций. В ходе идентификации задачи необходимо ответить на следующие вопросы:

«Какие задачи предлагается решать экспертной системе ?»

"Как эти задачи могут быть охарактеризованы и определены ?",

"На какие подзадачи разбивается каждая задача, какие данные они используют ?",

"Какие ситуации препятствуют решению ?",

"Как эти препятствия будут влиять на экспертную систему ?"

В процессе идентификации задачи инженер и эксперт работают в тесном контакте. Начальное содержательное описание задачи экспертом влечет за собой вопросы, инженера no-знаниям с целью уточнения терминов и ключевых понятий. Эксперт уточняет описание задачи, объясняет, как решать эту задачу и какие рассуждения лежат в основе решения. После нескольких циклов, уточняющих описание, эксперт и инженер по знаниям получают окончательное неформальное описание задачи.

При разработке экспертной системы типичными ресурсами являются: источники знаний, время разработки, вычислительные средства (возможности ЭВМ и программного инструментария) и объем

финансирования. Для достижения успеха эксперт и инженер должны использовать при построении ЭС все доступные им источники знаний. Для эксперта источниками знаний могут быть его предшествующий опыт по решению задачи, книги, конкретные примеры задач и использованных решений. Для инженера по знаниям источниками знаний могут быть опыт в решении аналогичных задач, методы решения и представления знаний, программный инструментарий.

При определении (назначении) временных ресурсов необходимо иметь в виду, что сроки разработки и внедрения экспертной системы составляют (за редким исключением) не менее шести месяцев (при трудоемкости от двух до пяти человеко-лет). Задача определения ресурсов является весьма важной, поскольку ограниченность какого-либо/ ресурса существенно влияет на процесс проектирования. Так, например, при недостаточном финансировании предпочтение может быть; отдано не разработке оригинальной новой системы, а адаптации существующей.

Задача идентификации целей заключается в формулировании в явном виде целей построения экспертной системы. При этом важно отличать цели, ради которых строится система, от задач, которые она должна решать. Примерами возможных целей являются: формализация неформальных знаний экспертов; улучшение качества решений, принимаемых экспертом; автоматизация рутинных аспектов работы эксперта (пользователя); тиражирование знаний эксперта.

На первом этапе инженер по знаниям должен ответить на основной вопрос: "Подходят ли методы инженерии знаний для решения предложенной задачи?" Для положительного ответа на данный вопрос необходимо, чтобы задача относилась к достаточно узкой, специальной области знаний и не требовала для своего решения использования того, что принято называть здравым смыслом, поскольку методы искусственного интеллекта не дают возможности формализовать это понятие. Кроме того, качество ЭС зависит в конечном счете от уровня сложности решаемой задачи и ясности ее

формулировки. Задача не должна быть ни слишком легкой, ни слишком трудной, Обычно число связанных понятий, релевантных проблеме, должно составлять несколько сотен. Говоря другими словами, назначение экспертной системы в том, чтобы решать некоторую задачу из данной области, а не в том, чтобы быть экспертом в этой области.

Тема 2.4. Проектирование интеллектуальных систем

В разработке ЭС участвуют представители следующих специальностей:

- эксперт в проблемной области, задачи которой будет решать ЭС;
- инженер по знаниям - специалист по разработке ЭС (используемые им технологию, методы называют технологией (методами) инженерии знаний);
- программист по разработке инструментальных средств (ИС), предназначенных для ускорения разработки ЭС.

Необходимо отметить, что отсутствие среди участников разработки инженеров по знаниям (т. е. их замена программистами) либо приводит к неудаче процесс создания ЭС, либо значительно удлиняет его.

Эксперт определяет знания (данные и правила), характеризующие проблемную область, обеспечивает полноту и правильность введенных в ЭС знаний.

Инженер по знаниям помогает эксперту выявить и структурировать знания, необходимые для работы ЭС; осуществляет выбор того ИС, которое наиболее подходит для данной проблемной области, и определяет способ представления знаний в этом ИС; выделяет и программирует (традиционными средствами) стандартные функции (типичные для данной проблемной области), которые будут использоваться в правилах, вводимых экспертом.

Программист разрабатывает ИС (если ИС разрабатывается заново), содержащее в пределе все основные компоненты ЭС, и осуществляет его сопряжение с той средой, в которой оно будет использовано.

Специалисты по искусственному интеллекту считают, что трудно описать в общем виде свойства, которые делают некую проблему подходящей для разработки соответствующей экспертной системы. Им намного проще взять описание конкретной проблемы и либо оценить ее пригодность, либо пересмотреть степень общности ее постановки или характер искомых решений, чтобы сделать ее приемлемой. Часто правильным вопросом оказывается не вопрос «Будет ли экспертная система пригодна для решения моих задач?», а вопрос «Какой аспект моей деятельности наиболее подходит для разработки экспертной системы?»

Однако организацию, обдумывающую разработку экспертной системы, интересуют совсем иные вопросы. По очевидным причинам первый вопрос, который она поставит, это «Будет ли экспертная система приложима к нашим конкретным задачам?» Хотя мы не можем дать простой общий ответ на этот важный вопрос, мы можем предложить некоторые рекомендации о том, как следует к нему подходить. Эти рекомендации можно подытожить так:

Прибегать к использованию экспертной системы следует только тогда, когда разработка экспертной системы возможна, оправдана и уместна.

Рассмотрим требования, которым должна удовлетворять предметная область, чтобы разработка экспертной системы стала возможна.

Одно из наиболее важных требований состоит в том, чтобы существовали подлинные эксперты. Это люди, которые по общему признанию имеют огромный опыт профессиональной работы в данной предметной области; они гораздо лучше новичков решают проблемы в этой области. Без такого источника глубоких знаний все усилия по разработке действительно совершенной программы будут обречены на неудачу.

Иметь высококвалифицированных экспертов еще недостаточно. Их оценки правильности выбора решения и его точности должны в основном совпадать. В противном случае подтверждение высокого качества экспертной системы станет почти невозможным. Эксперты должны также уметь четко формулировать и объяснять методы, которые они используют при решении задач в предметной области. Если они не умеют делать этого, то специалисты по построению знаний не смогут преуспеть в «извлечении» знаний из них и встроить эти знания в программы.

Другие требования при разработке экспертных систем связаны с характеристиками задачи, которую экспертная система должна решать, т. е. с работой, которую она должна выполнять. Эта работа должна требовать интеллектуальных, а не физических навыков. Если работа состоит в физических манипуляциях, которым можно научиться лишь на практике, то традиционный подход экспертных систем окажется неприменимым. Однако это не означает, что каждая проблема с (физическими компонентами должна быть отброшена. Если работа требует и интеллектуальных, и физических навыков, как, например, управление механическим манипулятором, обслуживающим конвейер, то интеллектуальную часть работы можно реализовать методами инженерии знаний, а физическую часть — более обычными техническими методами.

Следующее требование состоит в том, чтобы работа не была чрезвычайно трудной. Если эксперт не может обучить процессу работы новичка, так как соответствующая квалификация приобретается лишь на опыте практической деятельности, то может оказаться, что такой процесс слишком труден, чтобы его можно было вложить в экспертную систему. Или, если эксперту нужны дни или недели, а не часы, чтобы решить проблему, то весьма вероятно, что она слишком трудна или слишком сложна для подхода инженерии знаний. Однако если задача, требующая дней или недель напряженных усилий, может быть разбита на меньшие, более быстро

решаемые, относительно независимые подзадачи, то каждая такая подзадача может быть выбрана для разработки соответствующей экспертной системы.

В некоторых отношениях трудность задачи также связана с тем, насколько хорошо эксперт понимает предметную область, т. е. с тем, в какой степени знания и решения задачи точны и хорошо структурированы. Если задача столь нова или столь плохо понятна, что необходимы основательные исследования для нахождения решения, то подход инженерии знаний неприменим. Он также неприменим, если решение задачи требует существенного обращения к здравому смыслу.

Приведем требования, которым должна удовлетворять предметная область, чтобы разработка экспертной системы была оправдана.

Сама по себе возможность разработки экспертной системы для конкретной задачи еще не означает, что желательно ее сделать. Существует много способов обоснования оправданности разработки экспертной системы. Компания может оправдать разработку экспертной системы, когда полученное с ее применением решение приносит очень высокий доход. Например, экспертная система для разведки полезных ископаемых может открыть богатое рудное месторождение стоимостью в миллионы долларов. Если есть разумная вероятность получить большой доход, то разработка системы представляется хорошей идеей.

Разработка экспертной системы оправдана, когда эксперты-люди недоступны или не в состоянии выполнить работу. Часто экспертов-людей мало, а требуется их много, и поэтому их услуги дороги. Проблема усложняется, когда компании нужны сходные по выполняемым обязанностям специалисты во многих физически разных местах, например, нужно управлять процессом перегонки нефти на каждой нефтеперегонной колонне, принадлежащей нефтехимической компании. Это порождает необходимость иметь много версий эксперта, что легко сделать, и притом практически без дополнительных расходов, если в качестве эксперта

выступает компьютерная программа. В такой ситуации экспертная система является дешевым, эффективным способом решения проблемы. На самом деле она может оказаться единственным приемлемым по стоимости и эффективности вариантом.

Экспертные системы оправдывают себя, когда персональные перемещения приводят к значительной утере компетентности организациями. Выход в отставку, служебные перемещения, в военных ведомствах перевод по службе — все это часто вызывает развал работы или даже хаос, когда жизненно важный опыт, накопленный кадровыми сотрудниками, уходит вместе с ними. Поэтому институциональная память экспертной системы может свести к минимуму или даже снять эту проблему.

Наконец, разработка экспертной системы оправдана, когда принятие экспертных решений должно происходить в неблагоприятных или враждебных человеку условиях, например на ядерных реакторах, космических станциях или других планетах. Было бы слишком дорого или опасно пытаться использовать человека-эксперта в таких условиях. Конечно, решения можно было бы принимать дистанционно и посылать от человека-эксперта по каналам связи с помощью электронного оборудования. Но возможность задержек или помех, создаваемых противником, делают это решение менее привлекательным, чем автономный эксперт на месте событий.

Рассмотрим, когда разработка экспертной системы разумна.

Ключевые факторы в определении того, когда разработка экспертной системы разумна,— это характер, сложность и широта постановки задачи, которую нужно решить.

Характер

Чтобы экспертная система была применима к задаче, последняя должна обладать некоторыми специфическими свойствами. Это должна быть задача, которую можно было бы вполне естественным образом решить,

манипулируя символами и символьными структурами. Большинство задач реального мира требует символьных рассуждений, за исключением тех, которые имеют пригодные для использования математические решения. Поэтому задачи, ориентированные на математический аппарат, такие как решение дифференциальных уравнений численным методом, обычно не подходят для разработки экспертных систем. Однако некоторые математические задачи, такие как упрощение алгебраических выражений или преобразование тригонометрических выражений, вполне удобны для символьных рассуждений.

Большинство задач, с которыми успешно справляются экспертные системы, являются *эвристическими* по своей природе, т. е. они требуют использования эмпирических правил для получения приемлемого решения. Проблемы, которые могут быть решены посредством алгоритмов — формальных процедур, гарантирующих получение правильного решения всякий раз, когда они применяются — не очень подходящие для разработки экспертных систем. Например, существует много разных алгоритмов сортировки, и дешевле решать такую задачу с помощью обычной программы, чем с помощью экспертной системы. В некотором смысле, использование экспертных систем— это последнее средство, когда все остальные оказались неприменимыми. Если задача может быть решена математически или посредством хитроумных алгоритмов, то именно эти методы и должны быть использованы. Если она слишком трудна для этих обычных методов, то экспертные системы могут оказаться подходящими.

Сложность

Задача должна быть *не слишком легкой.* Это должна быть трудная, серьезная задача из такой области, что человеку нужно затратить годы учения и практики, чтобы стать в этой специалистом.

Широта постановки

Наконец, постановка задачи должна обладать надлежащей широтой. Она должна быть достаточно узкой, чтобы с ней можно было справиться, и

достаточно широкой, чтобы представлять практический интерес. К сожалению, определения этих критериев зависят от конкретной предметной области. Что еще хуже, выбор правильной постановки решающим образом определяет успех проекта экспертной системы. Действительно, одна из самых опасных ловушек при разработке экспертной системы — это выбрать задачу, которая является слишком широкой или общей, чтобы ее можно было решить адекватно.

Тема 2.5. Разработка интеллектуальных систем

Прототипная система является усеченной версией экспертной системы, спроектированной для проверки правильности кодирования фактов, связей и стратегий рассуждения эксперта. Она также дает возможность инженеру по знаниям привлечь эксперта к активному участию в процессе разработки экспертной системы» и, следовательно, к принятию им обязательства приложить все усилия к созданию системы в полном объеме.

Объем прототипа — несколько десятков правил, фреймов или примеров. На рис. 2.4 изображено шесть стадий разработки прототипа и минимальный коллектив разработчиков, занятых на каждой из стадий. Приведем краткую характеристику каждой из стадий, хотя эта схема представляет собой грубое приближение к сложному, итеративному процессу.

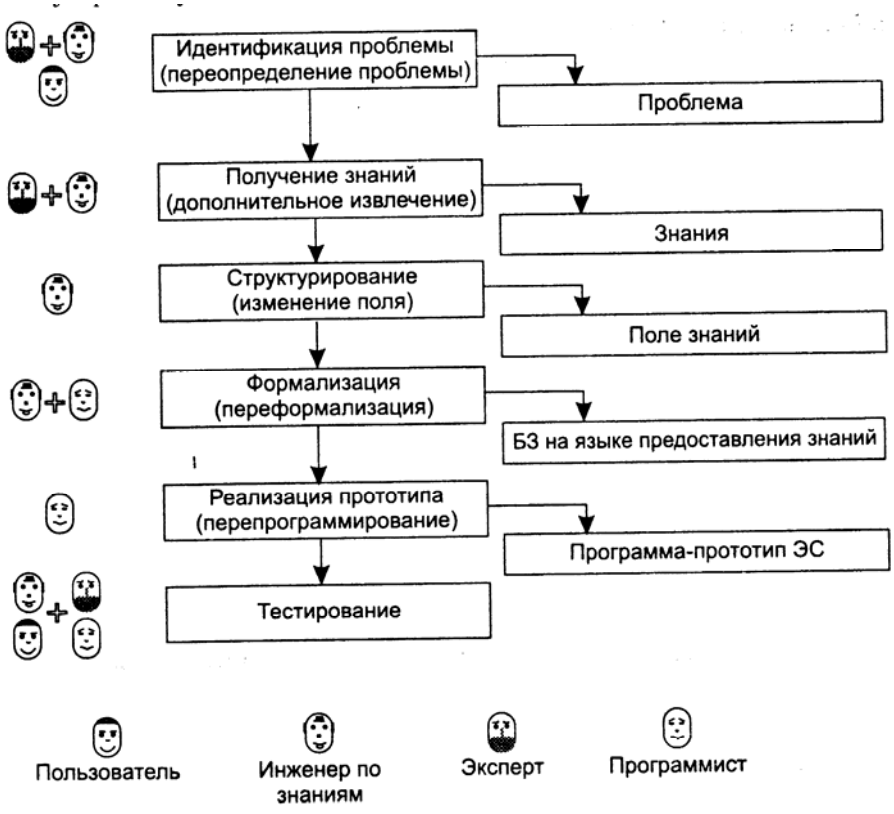


Рис. 2.4. Стадии разработки прототипа ЭС

Хотя любое теоретическое разделение бывает часто условным, осознание коллективом разработчиков четких задач каждой стадии представляется целесообразным. Роли разработчиков (эксперт, программист, пользователь и аналитик) являются постоянными на протяжении всей разработки. Совмещение ролей нежелательно. Сроки приведены условно, так как зависят от квалификации специалистов и особенностей задачи.

Идентификация проблемы

Уточняется задача, планируется ход разработки прототипа экспертной системы, определяются:

• необходимые ресурсы (время, люди, ЭВМ 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.

Идентификация проблемы — знакомство и обучение членов коллектива разработчиков**,** а также создание неформальной формулировки проблемы. Средняя продолжительность 1-2 недели.

Извлечение Знаний

На этой стадии происходит перенос компетентности от эксперта к инженеру по знаниям, с использованием различных методов:

• анализ текстов;

• диалоги;

• экспертные игры;

• лекции;

• дискуссии;

• интервью;

• наблюдение и другие.

Извлечение знаний — получение инженером по знаниям наиболее полного из возможных представлений о предметной области и способах принятия решения в ней. Средняя продолжительность 1-3 месяца.

Структурирование или концептуализация знаний

Выявляется структура полученных знаний о предметной области, то есть определяются:

• терминология;

• список основных понятий и их атрибутов;

• отношения между понятиями;

• структура входной и выходной информации;

• стратегия принятия решений;

• ограничения стратегий и т. д.

Структурирование (или концептуализация) знаний — 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.

Формализация

Строится формализованное представление концепций предметной области на основе выбранного языка представления знаний (ЯПЗ). Традиционно на этом этапе используются:

• логические методы (исчисления предикатов 1-го порядка и др.);

• продукционные модели (с прямым и обратным выводом);

• семантические сети;

• фреймы;

• объектно-ориентированные языки, основанные на иерархии классов, объектов.

Формализация знаний — 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.

Все чаще на этой стадии используется симбиоз языков представления знаний. Средняя продолжительность 1-2 месяца.

Реализация

Создается прототип экспертной системы, включающий базу знаний и остальные блоки, при помощи одного из следующих способов:

• программирование на традиционных языках типа Pascal, C++ и др.;

• программирование на специализированных языках, применяемых в задачах искусственного интеллекта: LISP, FRL, SMALLTALK и др.;

• использование инструментальных средств разработки ЭС;

• использование «пустых» ЭС или «оболочек» типа ЭКСПЕРТ

Реализация — разработка программного комплекса, демонстрирующего жизнеспособность подхода в целом. Чаще всего первый прототип отбрасывается на этапе реализации действующей ЭС. Средняя продолжительность 1-2 месяца.

Тестирование

Оценивается и проверяется работа программ прототипа с целью приведения в соответствие с реальными запросами пользователей. Прототип проверяется на:

• 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.);

• качество проверочных примеров;

• корректность базы знаний (полнота и непротиворечивость правил).

Тестирование — выявление ошибок в подходе и реализации прототипа и выработка рекомендаций по доводке системы до-промышленного варианта. Средняя продолжительность 1-2 недели.

Методы тестирования экспертных систем

Название метода	Описание
А. Тестирование на основе концепции "черного ящика"	Набор тестируемых ситуаций генерируется без учета используемых в системе методов решения задачи
Случайное тестирование	Тестируемые ситуации выбираются случайным образом из пространства входных наборов данных

Выборочное тестирование входов	Пространство входных наборов данных разбивается на выборки, для которых определяются ситуации для тестирования
Выборочное тестирование выходов	Тестируемые ситуации определяются на основе выборок, сформированных для выходных наборов данных
Б.Тестирование на основе концепции "белого ящика"	Тестируемые ситуации учитывают внутреннюю структуру системы в дополнение к входным и ожидаемым выходным наборам данных
Тестирование потоков данных	Анализ системы для выявления аномальных ситуаций, связанных с описанием, использованием и уничтожением переменных
Тестирование динамических потоков	Тестируемые ситуации генерируются для прохождения различных ветвей исполнения программы
Тестирование причин и следствий	Причины и следствия определяются на основе анализа решений, и тестовые ситуации формируются путем комбинации причин
В. Тестирование полноты базы знаний	Тестирование правил на внутреннюю неполноту
Поиск конфликтных правил	Поиск правил, возбуждающихся в сходных ситуациях, но приводящих к различным результатам
Поиск избыточных правил	Поиск правил, возбуждающихся в сходных ситуациях и приводящих к сходным результатам
Поиск пересекающихся правил	Анализ системы на наличие правил, являющихся подмножеством других правил
Г. Тестирование целостности базы знаний	Тестирование правил на внутреннюю целостность
Поиск пропущенных правил	Анализ системы для нахождения пропущенных правил, приводящих к требуемым результатам
Поиск атрибутов без ссылок	Анализ системы для нахождения атрибутов, на которые не существует ссылок ни в одном правиле
Поиск атрибутов с некорректными значениями	Нахождение некорректных значений атрибутов, на которые ссылаются правила системы

Проведенные по инициативе американского фонда National Science Foundation исследования позволяют сделать выводы об эффективности методов тестирования:

• случайное и выборочное тестирование наиболее эффективно на этапе идентификации;

• выборочное тестирование и анализ динамических потоков - на этапе концептуализации;

• поиск атрибутов без ссылок и выборочное тестирование - на стадии формализации;

• анализ потоков данных, поиск атрибутов с некорректными значениями и тестирование на основе концепции "черный ящик" - на этапе выполнения;

• выборочное тестирование и тестирование динамических потоков - при сквозном тестировании на всех фазах жизненного цикла экспертной системы.

Раздел III. ТЕХНОЛОГИИ ИНЖЕНЕРИИ ЗНАНИЙ

Тема 3.1. Приобретение знаний

Инженерия знания - достаточно молодое направление искусственного интеллекта, появившееся тогда, когда практические разработчики столкнулись с весьма нетривиальными проблемами трудности "добычи" и формализации знаний. В первых книгах по искусственному интеллекту эти факты обычно только постулировались, в дальнейшем начались серьезные исследования по выявлению оптимальных стратегий выявления знаний.

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.

Поле знаний формируется на третьей стадии разработки ЭС - стадии структурирования. Поле знаний, как первый шаг к формализации, представляет модель знаний о предметной области, в том виде, в каком ее

сумел выразить аналитик на некотором "своем" языке. Известно, что словарь языка конкретной науки формируется путем пополнения общеупотребительного языка специальными терминами и знаками, которые либо заимствуются из повседневного языка, либо изобретаются. Назовем этот язык L и рассмотрим его желаемые свойства, учитывая, что стандарта этого языка пока не существует, а каждый инженер по знаниям вынужден сам его изобретать. Во-первых, как и в языке любой науки, в нем должно быть как можно меньше неточностей, присущих обыденным языкам. Частично точность достигается более строгим определением понятий. Идеалом точности, конечно, является язык математики. Язык L, видимо, занимает промежуточное положение между естественным языком и языком математики.

Во-вторых, желательно не использовать в нем терминов иных наук. Это вызывает недоразумения. В-третьих, язык L, видимо, будет либо символьным языком, либо языком графическим (схемы, рисунки, пиктограммы).

При выборе языка описания поля знаний не следует забывать, что на стадии формализации необходимо его заменить на машинно-реализуемый язык представления знаний (ЯПЗ), выбор которого зависит от структуры поля знаний.

В некотором смысле создание языка L очень близко к идеям разработки универсальных языков науки. К XVII веку сложились два подхода в разработке универсальных языков: создание языков-классификаций и логико-конструктивных языков. К первому примыкают проекты, восходящие к идее Ф. Бэкона, - это языки Вилкинса и Далгарно. Второй подход связан с исследованиями в рамках поиска универсального метода познания, наиболее четко высказанного Р. Декартом, а затем в проекте универсальной характеристики Г. Лейбница. Именно Лейбниц наметил основные контуры учения о символах, которые в соответствии с его замыслами в XVIII веке развивал Г. Ламберт, который дал имя науке

"семиотика". Семиотика в основном нашла своих адептов в сфере гуманитарных наук. В последнее время сложилась также новая ветвь семиотики - прикладная семиотика. Представители естественных наук еще не до конца осознали достоинства семиотики только из-за того, что имеют дело с достаточно простыми и "жесткими" предметными областями. Им хватает аппарата традиционной математики. В инженерии знаний, однако, мы имеем дело с "мягкими" предметными областями, где явно не хватает выразительной адекватности классического математического аппарата и где большое значение имеет эффективность нотации (ее компактность, простота модификации, ясность интерпретации, наглядность и т. д.).

Языки семиотического моделирования как естественное развитие языков ситуационного управления являются первым приближением к языку инженерии знаний. Именно изменчивость и условность знаков делают семиотическую модель применимой к сложным сферам реальной человеческой деятельности. Поэтому главное на стадии концептуализации - сохранение естественной структуры поля знаний, а не выразительные возможности языка. Традиционно семиотика включает:

- синтаксис (совокупность правил построения языка или отношения между знаками);
- семантику (связь между элементами языка и их значениями или отношения между знаками и реальностью);
- прагматику (отношения между знаками и их пользователями).

Поле знании является некоторой семиотической моделью, которая может быть представлена как граф, рисунок, таблица, диаграмма, формула или текст в зависимости от вкуса инженера по знаниям и особенностей предметной области. Особенности предметной области могут оказать существенное влияние на форму и содержание компонентов структуры поля знаний. Рассмотрим соответствующие компоненты поля знаний.

Синтаксис

Обобщенно синтаксическую структуру поля знаний можно представить как P=(I,O,M),

где I - структура исходных данных, подлежащих обработке и интерпретации в экспертной системе;

O - структура выходных данных, то есть результата работы системы;

M - операциональная модель предметной области, на основании которой происходит модификация I в O.

Включение компонентов I и O в P обусловлено тем, что составляющие и структура этих интерфейсных компонентов имплицитно (то есть неявно) присутствуют в модели репрезентации в памяти эксперта. Операциональная модель M может быть представлена как совокупность концептуальной структуры системы, отражающей понятийную структуру предметной области, и функциональной структуры системы, моделирующей схему рассуждений эксперта:

M=(Sk, Sf).

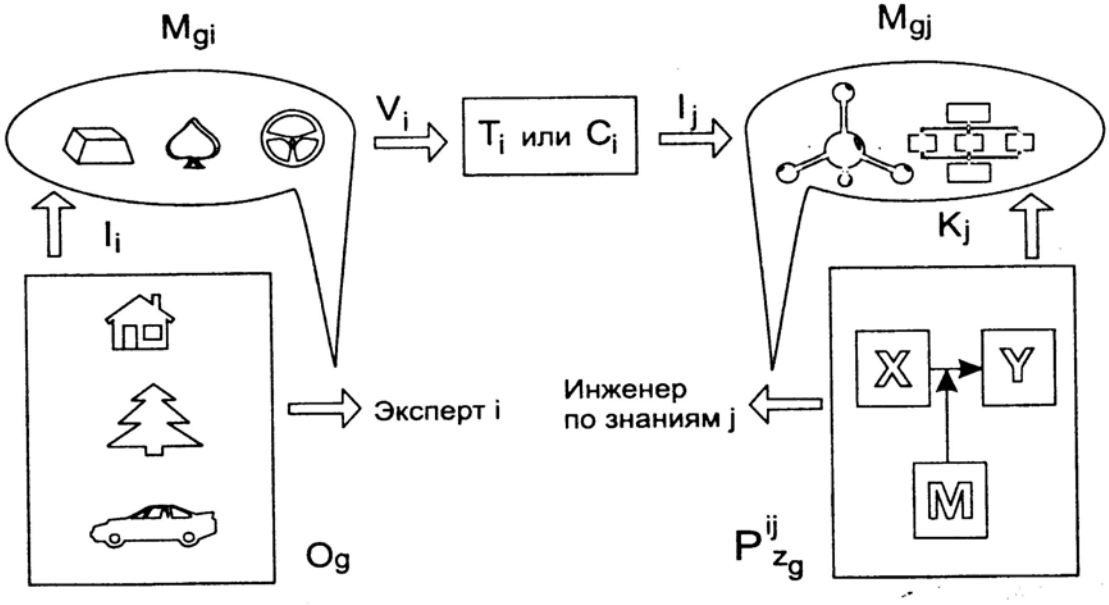
Sk выступает как статическая, неизменная составляющая, в то время как Sf, представляет динамическую, изменяемую составляющую.

Формирование Sk основано на выявлении понятийной структуры предметной области.

Семантику поля знаний можно рассматривать на двух уровнях.

- семантическая модель знаний эксперта о некоторой предметной области;
- любое поле знаний является моделью некоторых знаний.

Схему, отображающую отношения между реальной действительностью и полем знаний, можно представить следующим образом:



«Испорченный телефон» при формировании поля знаний

Прагматика

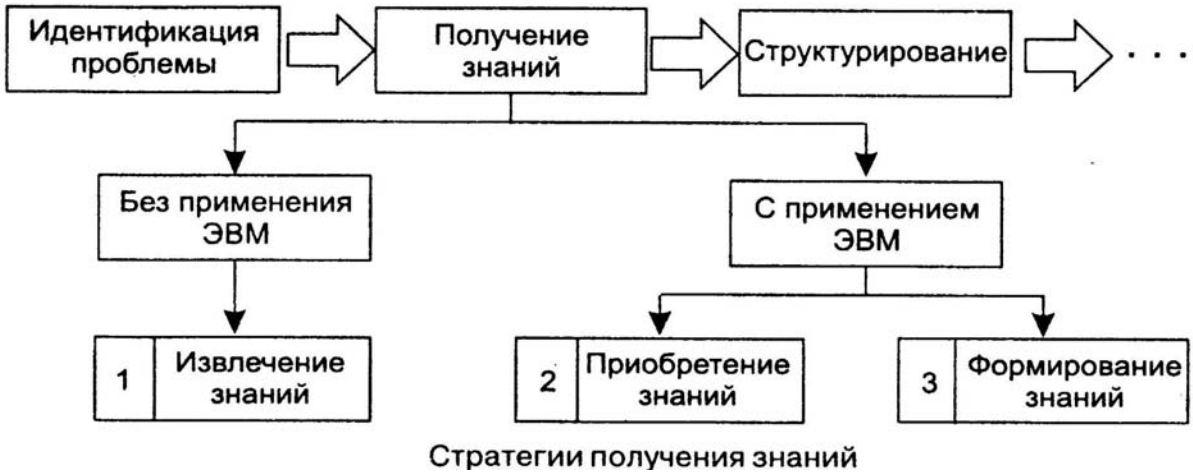
В качестве прагматической составляющей семиотической модели следует рассматривать технологии проведения структурного анализа предметной области, пользуясь которым инженер по знаниям может сформировать поле знаний по результатам стадии извлечения знаний.

Таким образом, под прагматикой будем понимать практические аспекты разработки и использования поля, то есть как от хаоса черновиков и стенограмм сеансов извлечения знаний перейти к стройной или хотя бы ясной модели.

Стратегии получения знаний

При формировании поля знаний ключевым вопросом является сам процесс получения знаний, когда происходит перенос компетентности экспертов на инженеров по знаниям. Для названия этого процесса в литературе по ЭС получило распространение несколько терминов: приобретение, добыча, извлечение, получение, выявление, формирование знаний. В англоязычной специальной литературе в основном используются два: acquisition (приобретение) и elicitation (выявление, извлечение, установление).

Выделяют три основные стратегии проведения стадии получения знаний



Стратегии получения знаний

Термин приобретение трактуется либо очень широко - тогда он включает весь процесс передачи знаний от эксперта к базе знаний ЭС, либо уже как способ автоматизированного построения базы знаний посредством диалога эксперта и специальной программы (при этом структура поля знаний заранее закладывается в программу). В обоих случаях термин "приобретение" не касается самого таинства экстрагирования структуры знаний из потока информации о предметной области. Этот процесс описывается понятием "извлечение".

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

В настоящее время большинство разработчиков ЭС отмечает, 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;

- 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 ЭС, желая упростить эту процедуру, пытаются подменить инженера по знаниям самим экспертом. По многим причинам это нежелательно.

Поскольку основной проблемой инженерии знаний является процесс извлечения знаний, инженеру по знаниям необходимо четко понимать природу и особенности этих процессов. Для того чтобы разобраться в природе извлечения знаний, выделим три основных аспекта этой процедуры :

A - {Al, A2, A3} - {психологический, лингвистический, гносеологический}.

Психологический аспект

Из трех аспектов извлечения знаний психологический - Al - 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 в общении психология является доминантной.

Общение, или коммуникация (от лат. communicatio - связь), - это междисциплинарное понятие, обозначающее все формы непосредственных контактов между людьми - от дружеских до деловых. Оно широко исследуется в психологии, философии, социологии, этологии, лингвистике,

семиотике и других науках. 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 w двухтактный обмен порциями сведений, а нерасчлененный процесс циркуляции информации, то есть совместный поиск истины.

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.

Можно выделить такие структурные компоненты модели общения при извлечении знаний:

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

В соответствии с этой структурой выделим три "слоя" психологических проблем, возникающих при извлечении знаний:

Al = {Sll, S12, S13 } = {контактный, процедурный, когнитивный}.

Лингвистический аспект

Лингвистический аспект касается исследований языковых проблем, так как язык - это основное средство общения в процессе извлечения знаний. В инженерии знаний можно выделить три слоя лингвистических проблем:

A2 = {S21, S22, S23} - {"общий код", понятийная структура, словарь пользователя}.

Гносеологический аспект извлечения знаний

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 базы знаний эксперт часто впервые формулирует некоторые закономерности, до того момента составлявшие его личный опыт.

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, то есть строить 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.

Основными методологическими критериями научности, позволяющими считать научным и само новое знание и способ его получения, являются:

А3 = {S31, S32, S33} = {внутренняя согласованность, системность, объективность, историзм}.

Классификация методов практического извлечения 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 инженер по знаниям комбинирует различные методы, например сначала изучает литературу, затем беседует с экспертами, или наоборот.

В свою очередь, коммуникативные методы можно также разделить на две группы: активные и пассивные. Пассивные методы подразумевают, что ведущая роль в процедуре извлечения как бы передается эксперту, а инженер по знаниям только протоколирует рассуждения эксперта во время его реальной работы по принятию решений или записывает то, что эксперт считает нужным самостоятельно рассказать в форме лекции. В активных

методах, напротив, инициатива полностью в руках инженера по знаниям, который активно контактирует с экспертом различными способами - в играх, диалогах, беседах за круглым столом и т. д.

Следует еще раз подчеркнуть, что и активные и пассивные методы могут чередоваться даже в рамках одного сеанса извлечения знаний.

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. 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.

Методы извлечения знаний, рассмотренные выше, являются непосредственной подготовкой к структурированию знаний.

Тема 3.2. Структурирование знаний

Простейший алгоритм структурирования знаний

В качестве простейшего прагматического подхода к формированию поля знаний начинающему инженеру по знаниям можно предложить следующий алгоритм:

1. Определение входных {X} и выходных {Y} данных. Этот шаг совершенно необходим, 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 знании и выписываются все значимые слова, обозначающие понятия, явления, процессы, предметы, действия, признаки и т. н. 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 между понятиями. Все в мире связано. Но определить, как направлены связи, что ближе, а что дальше, необходимо на этом этапе. Таким образом, строится сеть ассоциаций, где связи только намечены, но пока не поименованы.

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

6. Построение 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. 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.

8. Определение стратегий принятия решений. Определение стратегий принятия решения, то есть выявление цепочек рассуждений, связывает все сформированные ранее понятия и отношения в динамическую систему поля знаний.

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

Инженер по знаниям может испытывать необходимость в применении специальных методов структурирования на разных шагах алгоритма. При этом, естественно, для таких простых и очевидных шагов, как определение входных и выходных понятии или составление словаря, никаких искусственных методов предлагаться не будет.

Методы выявления объектов, понятий и их атрибутов Понятие или концепт - это обобщение предметов некоторого класса по их специфическим признакам. Обобщенность является сквозной характеристикой всех когнитивных психических структур, начиная с простейших сенсорных образов. Так, понятие "автомобиль" объединяет множество различных предметов, но все они имеют четыре колеса, двигатель и массу других деталей, позволяющих перевозить на них грузы и людей. Существует ряд методов выявления понятий предметной области в общем словаре терминов, который составлен на основании сеансов извлечения знаний. При этом важно выявление не только самих понятий, но и их признаков.

Сложность заключается в том, что для многих понятий практически невозможно однозначно определить их признаки, это связано с различными формами репрезентации понятий в памяти человека.

Стратификация знаний

Основы объектно-структурного анализа (ОСА) были предложены в работах Т.А.Гавриловой. ОСА подразумевает дезагрегацию предметной области, как правило, на восемь страт или слоев:

S_1 ЗАЧЕМ-знания: Стратегический анализ: назначение и функции системы

S_2 КТО-знания: Организационный анализ: коллектив разработчиков системы

S_3 ЧТО-знания: Концептуальный анализ: основные концепты, понятийная структура

S_4 КАК-знания: Функциональный анализ: гипотезы и модели принятия решения

S_5 ГДЕ-знания: Пространственный анализ: окружение, оборудование, коммуникации

S_6 КОГДА-знания: Временной анализ: временные параметры и ограничения

S_7 ПОЧЕМУ-знания: Каузальный или причинно-следственный анализ: формирование подсистемы объяснений

S_8 СКОЛЬКО-знания: Экономический анализ: ресурсы, затраты, прибыль, окупаемость

Разделение стадий извлечения и структурирования знаний является весьма условным, поскольку хороший инженер по знаниям, уже извлекая знания, начинает работу по структурированию и формированию поля знаний, описанному выше.

Однако в настоящее время прослеживается тенденция опережения технологических средств разработки интеллектуальных систем по отношению к их теоретическому обоснованию. Практически сейчас существует пропасть между блестящими, но несколько "постаревшими" математическими основами кибернетики (труды Винера, Эшби, Шеннона, Джорджа, Клира, Йордона, Ляпунова, Глушкова и др.) и современным

поколением интеллектуальных систем, которые основаны 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 слабо формализуемых смысловых задач, в которых не применим 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.

Тема 3.3. Новые тенденции и прикладные аспекты инженерии знаний

Методы многомерного шкалирования

Многомерное шкалирование (МШ) сегодня - это математический инструментарий, предназначенный для обработки данных о попарных сходствах, связях или отношениях между анализируемыми объектами с целью представления этих объектов в виде точек некоторого координатного пространства. МШ представляет собой один из разделов прикладной статистики, научной дисциплины, разрабатывающей и систематизирующей понятия, приемы, математические методы и модели, предназначенные для сбора, стандартной записи, систематизации и обработки статистических данных с целью их лаконичного представления, интерпретации и получения научных и практических выводов. Традиционно МШ используется для решения трех типов задач:

1. Поиск и интерпретация латентных (то есть скрытых, непосредственно не наблюдаемых) переменных, объясняющих заданную структуру попарных расстояний (связей, близостей).

2. Верификация геометрической конфигурации системы анализируемых объектов в координатном пространстве латентных переменных.
3. Сжатие исходного массива данных с минимальными потерями в их информативности.

Независимо от задачи МШ всегда используется как инструмент наглядного представления (визуализации) исходных данных. МШ широко применяется и исследованиях по антропологии, педагогике, психологии, экономике, социологии.

В основе данного подхода лежит интерактивная процедура субъективного шкалирования, когда испытуемому (то есть эксперту) предлагается оценить сходство между различными элементами с помощью некоторой градуированной шкалы (например, от 0 до 9, или от -2 до +2). После такой процедуры аналитик располагает численно представленными стандартизованными данными, поддающимися обработке существующими пакетами прикладных программ, реализующими различные алгоритмы формирования концептов более высокого уровня абстракции и строящими геометрическую интерпретацию семантического пространства в евклидовой системе координат.

Основной тип данных в МШ — меры близости между двумя объектами (i, j) – d_{ij}. Если мера близости такова, что самые большие значения d_{ij} соответствуют парам наиболее похожих объектов, то d_{ij} — мера сходства, если, наоборот, наименее похожим, то d_{ij} — мера различия.

МШ использует дистанционную модель различия, используя понятие расстояния в геометрии как аналогию сходства и различия понятий. Для того чтобы функция d, определенная на парах объектов (a, b), была евклидовым расстоянием, она должна удовлетворять следующим четырем аксиомам:

$d(a,b) >= 0$,

$d(a,a) = 0$,

$d(a,b) = d(b,a)$,

d(a,b)+d(b,c) >= d(a,c).

Тогда, согласно обычной формуле евклидова расстояния, мера различия двух объектов i и j со значениями признака k у объектов i и j соответственно X_{ik} и X_{jk}

$$\delta_{ij} = d_{ij} = \left[\sum_{k=1}^{K} \left(x_{ik} - x_{jk} \right)^2 \right]^{1/2}$$

Дистанционная модель была многократно проверена в социологии и психологии, что дает возможность оценить ее пригодность для использования.

В большинстве работ по МШ используется матричная алгебра. Геометрическая интерпретация позволяет представить абстрактные понятия матричной алгебры в конкретной графической форме. Для облегчения интерпретации решения задачи МШ к первоначально оцененной матрице координат стимулов X применяется вращение.

Среди множества алгоритмов МШ широко используются различные модификации метрических методов Торгерсона, а также неметрические модели, например Крускала.

При сравнении методов МШ с другими методами анализа, теоретически применимыми в инженерии знаний (иерархический кластерный анализ или факторный анализ), МШ выигрывает за счет возможности дать наглядное количественное координатное представление, зачастую более простое и поэтому легче интерпретируемое экспертами.

Метод репертуарных решеток

Среди методов когнитивной психологии — науки, изучающей то, как человек познает и воспринимает мир, других людей и самого себя, как формируется целостная система представлении и отношений конкретного

человека, особое место занимает такой метод личностной психодиагностики, как *метод репертуарных решеток* («repertory grid»).

Впервые метод был сформулирован автором теории личностных конструктов Джорджем Келли в 1955 г. Чем шире набор личностных конструктов у субъекта, тем более многомерным, дифференцированным является образ мира, человека, других явлений и предметов, то есть тем выше его когнитивная сложность.

Репертуарная решетка представляет собой матрицу, которая заполняется либо самим испытуемым, либо экспериментатором в процессе обследования или беседы. Столбцу матрицы соответствует определенная группа объектов, или, иначе, *элементов.* В качестве объектов могут выступать люди, предметы, понятия, отношения, звуки, цвета — все, что интересует психодиагноста. Строки матрицы представляют собой *конструкты* — биполярные признаки, параметры, шкалы, альтернативные противоположные отношения или способы поведения. Конструкты либо задаются исследователем, либо выявляются у испытуемого с помощью специальных приемов и процедур выявления. Вводя понятие конструкта, Келли объединяет две функции: функцию обобщения и функцию противопоставления. Он предлагает несколько определений понятия «конструкт». Одно из них:

Конструкт — это некоторый признак или свойство, по которому два или несколько объектов сходны между собой и, следовательно, отличны от третьего объекта или нескольких других объектов.

Например, выделение из трех предметов «диван, кресло, стул» двух «диван и кресло» выявляет конструкт «мягкость мебели». Келли в своих работах подчеркивает биполярность конструктов. Он считает, что, утверждая что-нибудь, мы всегда одновременно что-то отрицаем. Именно биполярность конструктов делает возможным построение репертуарной решетки. Например, север — юг — это *референтная* ось: элементы, которые в одном контексте являются «севером», в другом становятся «югом».

Возможности конструкта ограничены. Они могут быть применены только к некоторым объектам. Это нашло свое отражение в понятии *«диапазона пригодности»* конструкта.

Конструкты — не изолированные образования. Они взаимодействуют друг с другом, причем характер этого взаимодействия не случаен, а носит целостный системный характер.

В процессе заполнения репертуарной решетки испытуемый должен оценить каждый объект по каждому конструкту или каким-то другим образом поставить в соответствие элементы конструктам.

Определение *репертуарная* означает, что элементы выбираются по определенным правилам так, чтобы они соответствовали какой-либо одной области и все вместе были связаны осмысленным образом (контекстом) аналогично репертуару ролей в пьесе. Предполагается, что, изменяя репертуар элементов, можно «настраивать» методики на выявление конструктов разных уровней общности и относящихся к разным системам.

При переводе с английского языка термин матрица не используется, поскольку репертуарная решетка не всегда является матрицей в строгом смысле этого термина: в ней на пересечении строк и столбцов не обязательно стоят числа, не всегда выдерживается прямоугольный формат, строки могут быть разной длины. Второй смысл этого определения заключается в том, что в технике репертуарных решеток часто элементы задаются в виде обобщенных инструкций, репертуара ролей, на место которых каждый конкретный человек мысленно подставляет своих знакомых людей или конкретные предметы, если в качестве элементов заданы названия предметов.

По всей видимости, репертуарные решетки лучше считать специфической разновидностью структурированного интервью. Обычно мы исследуем систему конструктов другого человека в ходе разговора с ним. В процессе беседы мы постепенно начинаем понимать, как он видит мир, что с

чем связано, что из чего следует, что для чего важно, а что нет, как он оценивает других людей, события и ситуации.

Решетка формализует этот процесс и дает математическое обоснование связен между конструктами данного человека, позволяет более детально изучить отдельные подсистемы конструктов, подметить индивидуальное, специфическое в структуре и содержании мировоззрения человека.

Важное положение техники репертуарных решеток: ориентация на выявление собственных конструктов испытуемого, а не навязывание их ему извне.

Гибкость и эффективность репертуарных решеток, качество и количество получаемой информации делают их пригодными для решения широкого круга задач. Методики этого типа используются в различных областях практической деятельности: в педагогике и социологии, в медицине, рекламе и дизайне. Репертуарные решетки оказались методом, идеально приспособленным для реализации в виде диалоговых программ на компьютере, что также способствовало их широкому распространению. Достоинства и преимущества данного метода полностью раскрываются тогда, когда есть возможность, проведя исследование, быстро обработать результаты и проанализировать их с тем, чтобы уже при следующей встрече с испытуемым можно было уточнить и проверить возникшие предположения, составить и провести репертуарную решетку другого типа, а если это необходимо, и дополнить прежнюю, изменив репертуар элементов или выборку конструктов.

Раздел IV. **Методика проектирования экспертных систем на базе инструментальных средств семейства ЭКСПЕРТ-ЭКО**

4.1. Назначение комплекса

Комплекс представляет собой совокупность программных средств, предназначенных для создания и использования экспертных систем, решающих задачи в статических проблемных областях.

Комплекс может быть использован для создания экспертных систем диагностики технических, биологических объектов, вывода эвристических оценок риска или надежности (в строительстве, медицине и т.д.), качественного прогнозирования, обучения.

Средствами комплекса экспертные системы могут создаваться непосредственно высококвалифицированными специалистами в области приложения (экспертами), не обладающими навыками программирования. В сложных случаях они могут привлекать специалистов в области создания экспертных систем (инженеров по знаниям).

Экспертные системы, создаваемые средствами комплекса, позволяют решать конкретные прикладные задачи, а также объяснять, ЗАЧЕМ во время решения пользователю задается тот или иной вопрос и КАК получен результат.

Комплекс работает в диалоговом режиме. Использование подсказок и управление диалогом с помощью меню обеспечивает доступность средств комплекса для пользователя-непрограммиста.

Комплекс имеет средства для организации взаимодействия с внешним программным окружением.

1.2. Использование комплекса для разработки экспертных систем

Программы комплекса представляет собой настраиваемую "оболочку" экспертной системы.

"Оболочками" называются такие инструментальные средства, в которых реализован некоторый стандартный способ представления знаний (т.е. реализован некоторый язык представления знаний) и стандартизован механизм вывода решений. Подобная стандартизация позволяет значительно снизить трудоемкость разработки экспертных систем, так как "оболочка" содержит все программные средства, необходимые для создания и использования этих систем. "Оболочки" включают в себя следующие подсистемы: средства приобретения знаний, средства решения задач

(ведения консультации), средства объяснения и ведения диалога с пользователем, а также базу знаний (возможно, пустую). Разработка экспертной системы с помощью "оболочки" сводится к введению в базу средствами приобретения знаний описания знаний эксперта о проблемной области и порядке решения задач, выраженного на языке представления знаний "оболочки". Решение конкретных задач осуществляется средствами ведения консультации путем интерпретации содержимого базы знаний с учетом данных о конкретной ситуации в проблемной области (исходных данных консультации).

Средствами комплекса реализованы два режима работы: режим приобретения знаний и режим консультации (режим решения задач). Оба режима обеспечивают диалоговое взаимодействие с пользователем. Режим консультации обеспечивает объяснение получаемых решений.

Функционирование комплекса в различных фазах существования экспертных систем

Выделяют следующие фазы существования экспертных систем: создание, использование и модификацию. Программы комплекса используются во всех фазах. В фазе создания (модификации) с помощью средств комплекса разработчик экспертной системы может вводить в базу знаний (модифицировать) описание знаний эксперта, выраженное на языке представления знаний. Кроме того, он может отлаживать базу знаний, решая с ее помощью тестовые задачи и устраняя найденные ошибки. При традиционном подходе к разработке программ фазе создания экспертной системы соответствуют этапы алгоритмизации, программирования и отладки. В фазе использования экспертной системы программы комплекса применяются для решения задач из проблемной области.

В фазе создания (модификации) экспертной системы пользователями комплекса являются:

- эксперты, т.е. специалисты в той области знаний, задачи которой должна решать экспертная система;

- инженеры по знаниям, т.е. специалисты по проектированию экспертных систем.

Эксперт должен построить адекватную модель проблемной области, т.е. описать саму область и указать порядок решения задач данного типа.

Инженер по знаниям должен: помочь эксперту выявить знания, необходимые для функционирования экспертной системы; выразить выявленные знания на языке представления знаний; при необходимости выделить и запрограммировать традиционными средствами некоторые нестандартные функции, не предусмотренные в комплексе.

Разработанное описание модели проблемной области, выраженное на языке представления знаний, вводится в базу знаний в режиме приобретения знаний и преобразуется во внутреннее представление модели (в дальнейшем по тексту - модель). Ввод описания может осуществляться по частям, при этом проверяется синтаксическая корректность вводимых фрагментов.

После того как описание модели введено полностью, проверяется целостность модели. Если в ходе проверки ошибки не обнаружены, полученная модель помечается как корректная (т.е. готовая к работе), после чего она может быть использована в режиме консультации. При обнаружении ошибок сообщения о них заносятся в протокол проверки, а модель помечается как неготовая к работе. Ошибки могут быть исправлены путем коррекции описания модели в режиме приобретения знаний.

Экспертная система считается созданной, если в базе знаний получена соответствующая корректная модель.

Допускается создание иерархических экспертных систем, включающих несколько взаимосвязанных моделей. Каждая модель предназначена для решения некоторой частной подзадачи в проблемной области, причем одна из них - управляющая, или главная. Все модели оформляются в виде отдельных файлов данных. Единственным ограничением на количество моделей в базе знаний является объем доступной памяти на внешнем носителе информации.

Решение задачи в проблемной области будет осуществляться программами комплекса в режиме консультации на основе:

- одной или нескольких взаимосвязанных моделей, находящихся в базе знаний и представляющих знания эксперта о методах решения задач данного типа (например, модель для диагностики сердечно-сосудистых заболеваний);
- исходных данных, описывающих конкретную ситуацию в проблемной области, для которой пользователь хочет получить результат.

Чтобы получить решение задачи, необходимо в режиме консультации выбрать в базе знаний соответствующую модель (или главную модель), а затем в ответ на вопросы программ комплекса ввести те исходные данные, в которых возникает необходимость в ходе вывода решения.

Для проверки адекватности модели необходимо провести ее тестирование в режиме консультации. В ходе тестирования экспертная система должна решить достаточное количество контрольных задач. Результаты ее работы следует сравнить с решениями, предложенными экспертом. Кроме того, с помощью средств объяснения эксперт должен проверить правильность рассуждений системы в ходе вывода решений. Если эксперт не удовлетворен решениями или рассуждениями системы, он может с помощью средств объяснения локализовать ошибки, а затем исправить их в режиме приобретения знаний.

В фазе использования экспертной системы с программами комплекса взаимодействуют конечные пользователи, т.е. те специалисты, для которых создавалась система. В этой фазе комплекс работает только в режиме консультации.

Создаваемые средствами комплекса экспертные системы могут использоваться как неспециалистами, так и специалистами в областях приложения этих систем. В первом случае система дает пользователю совет, который тот не может получить сам. Во втором случае использование

системы ускоряет процесс получения результата и (или) освобождает специалиста от рутинной работы.

В данной фазе средства объяснения обеспечивают доверие пользователя к полученным экспертной системой результатам, а также демонстрируют ему применяемые экспертами методы решения задач, которые можно использовать в обучении.

1.3. Представление знаний экспертов

Структура знаний, используемых программами комплекса при решении задачи, изображена на рис. 1.

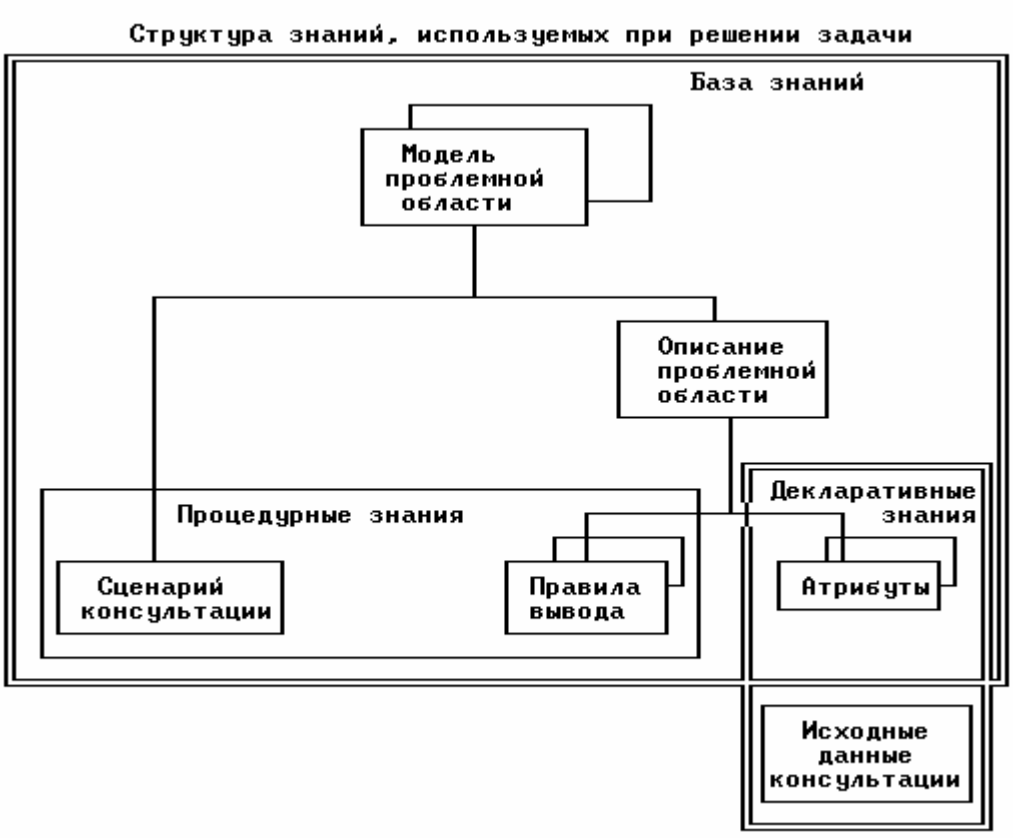


Рис. 1

База знаний может содержать несколько моделей, предназначенных для поддержания работы экспертных систем в различных областях приложения. Каждая модель включает описание проблемной области и знания о порядке решения задач (сценарий консультации).

Описание проблемной области состоит из описаний атрибутов и правил вывода. Атрибуты используются для описания состояния проблемной

области, например атрибуты "возраст", "диагноз" и т.д. Описание проблемной области должно содержать описания всех атрибутов, которые будут использоваться при решении задачи. Описание атрибута включает область определения этого атрибута, а также некоторую лингвистическую информацию, необходимую программам комплекса для ведения диалога с конечным пользователем во время консультации. Комплекс ориентирован на работу со статическими проблемными областями (значения атрибутов не изменяются в ходе решения задачи), в которых проблемная область может быть описана с помощью априорно заданного набора атрибутов (не допускается динамическое создание атрибутов во время решения задачи).

Средства комплекса позволяют представлять качественные и количественные характеристики проблемной области. Высказывания типа "A есть B", где A - атрибут, представляющий качественную характеристику, а B - элемент области определения этого атрибута, называются утверждениями о состоянии проблемной области. Например, высказывание "диагноз - острый бронхит" является утверждением в предметной области, если в ней определен атрибут "диагноз", одним из возможных значений которого является "острый бронхит". Решение задачи сводится к получению значений некоторых целевых атрибутов (например, "диагноз" или "метод обезболивания"), и/или определению истинности некоторых целевых утверждений.

Значения атрибутов и сведения об истинности утверждений о состоянии проблемной области либо запрашиваются в ходе консультации у конечного пользователя (исходные данные), либо вычисляются по правилам вывода на основе исходных данных.

Сценарий консультации указывает, значения каких атрибутов должны быть получены в результате консультации, в каком порядке их следует выводить, каким образом сообщать о полученных результатах и т.д. Сценарий и правила вывода образуют процедурные знания, причем сценарий

выполняет роль метазнаний по отношению к правилам. Описания атрибутов и исходные данные образуют декларативные знания о проблемной области.

Программные средства комплекса позволяют работать с неточно определенными (неопределенными) знаниями о качественных характеристиках проблемной области. Неточная определенность декларативных знаний представляется с помощью коэффициентов определенности, характеризующих степень уверенности в истинности утверждений о состоянии проблемной области. Неточная определенность правил может представляться с использованием формул нечеткой логики, формулы Байеса, а также произвольных эвристических формул, задаваемых экспертами.

Средствами комплекса можно создавать такие системы, которые получают все возможные решения коэффициентов определенности.

Коэффициент определенности является действительным числом с точностью до двух знаков после запятой и принимает значения из диапазона от -5.00 до 5.00.

В комплексе реализован диалоговый ввод знаний с семантико-синтаксическим контролем знаний, проверкой их целостности, а также компиляция знаний, повышающая эффективность работы комплекса в режиме консультации.

Решение задач осуществляется в режиме диалога конечного пользователя с программными средствами комплекса. В ходе решения задачи делается попытка вывести значения тех атрибутов, которые отмечены в сценарии как целевые, а также выполняются другие описанные в сценарии действия (выдаются сообщения о результатах и т.д.). При выводе значений атрибутов используется метод поиска решения " в глубину" с последующим выводом значений от данных.

В ходе решения задач пользователь может получить объяснения, для чего потребовались запрашиваемые у него данные и как получено решение задачи.

Средства комплекса могут проводить трассировку хода решения задачи, что облегчает отладку моделей.

Кроме того, программы комплекса могут обмениваться данными с внешними программами.

1.4. Структура комплекса

Комплекс состоит из основных (ОПС) и вспомогательных программных средств. Основные программные средства состоят из взаимосвязанных программных компонент, использование которых в режимах консультации и приобретении знаний обеспечивается автоматически. Вспомогательные средства реализованы в виде автономных программ, применение которых при создании экспертных систем не является обязательным. В то же время, использование этих средств позволяет улучшить качество создаваемых систем или облегчить процесс их создания.

ОПС поставляются в двух вариантах: основном и учебном. Основной вариант предназначен для создания промышленных экспертных систем. Учебный - для обучения работе с комплексом на примере "игрушечных" экспертных систем небольшого объема.

Структура основных программных средств комплекса приведена на рис. 2. Они включают:

- Редактор, обеспечивающий работу в режиме приобретения знаний;
- Консультатор, обеспечивающий работу в режиме консультации;
- Загрузчик, обеспечивающий переключение режимов;
- Помощь, которая позволяет пользователю оперативно получать доступ к справочной информации в соответствии с текущим контекстом;
- Базу знаний, которая может быть пустой (например, перед началом эксплуатации комплекса);

- Демонстратор, обеспечивающий запуск демонстрационных примеров для быстрого ознакомления с основными возможностями Редактора и Консультатора.

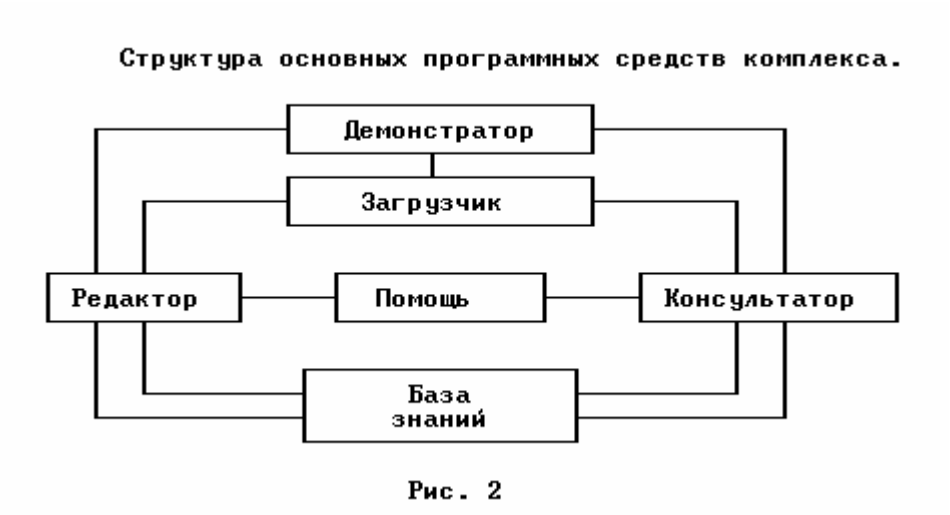


Рис. 2

В процессе функционирования комплекса Редактор и Консультатор обмениваются управлением и данными. Обмен управлением производится при помощи Загрузчика, обмен данными осуществляется через Базу знаний. Помощь может быть вызвана в процессе диалога пользователя с Редактором и Консультатором в момент, когда управление находится у пользователя.

В основном варианте ОПС имеется также пакетный Консультатор (П-Консультатор), обеспечивающий решение задач в пакетном режиме.

1.5. Структура моделей проблемной области

Модель включает в себя набор атрибутов, определяющих множество допустимых состояний проблемной области, правила вывода, описывающие знания о методах решения задач, и сценарий консультации.

Для представления количественных характеристик проблемной области используются числовые атрибуты - переменные, определяемые на числовых интервалах. Например, возраст сотрудника может задаваться с помощью числового атрибута "возраст", определенного на интервале от 18 до 90 . Числовой атрибут во время решения задачи может иметь только одно значение.

Значения числовых атрибутов - действительные числа, содержащие не более двух знаков после десятичной точки и не превышающие по абсолютной величине 64 000.00.

Качественные характеристики проблемной области описываются символьными атрибутами, которые определяются на конечных множествах значений, задаваемых перечислением. Например, рекомендуемый для местного обезболивания лекарственный препарат можно представить в виде символьного атрибута "анестетик", определенного на следующем множестве: "новокаин", "тримекаин", "лидокаин" и т.д. Высказывания типа "А есть В", где А - символьный атрибут, а В - элемент области определения этого атрибута, называются утверждениями. Значением утверждения является коэффициент его определенности. Вывод символьного атрибута, т.е. вычисление его значения, состоит в определении коэффициентов определенности утверждений, образованных из всех возможных значений этого атрибута. Допускаются множественные значения символьного атрибута, т.е. возможно сосуществование в какой-либо ситуации нескольких его значений. Например, символьный атрибут "иняз", задающий иностранные языки, которыми владеет специалист, может иметь одновременно несколько точно определенных значений.

Коэффициенты определенности утверждений - это действительные числа, принимающие значения от -5.00 до 5.00. Коэффициенту определенности D(H) утверждения H можно дать следующую содержательную интерпретацию:

- если точно известно, что H истинно, то D(H) = 5.00;
- если точно известно, что H ложно, то D(H) = -5.00;
- если H может быть с одинаковой уверенностью истинно или ложно, то D(H) = 0.00;
- если H скорее истинно, чем ложно, то 0.00 < D(H) < 5.00, причем D(H) тем больше, чем больше уверенность в истинности H;

- если H скорее ложно, чем истинно, то -5.00 < D(H) < 0.00, причем D(H) тем меньше, чем больше уверенность в ложности H.

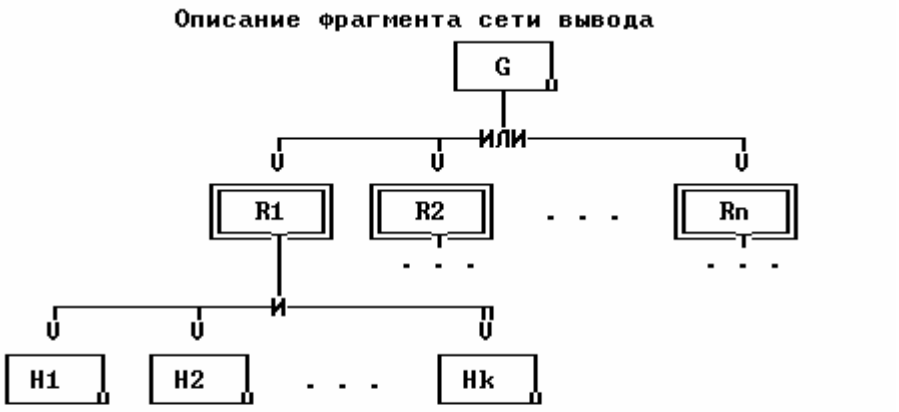
Утверждения и числовые атрибуты называются целями, а символьные атрибуты, представляющие собой множества утверждений, называются сложными целями. Значения целей определяются с помощью правил вывода. Правило вывода указывает, каким образом можно получить значение цели по значениям других атрибутов и утверждений, называемых подцелями правила. В зависимости от типа цели (простая или сложная) правила делятся на простые и сложные.

Правила могут иметь условия применимости - выражения, вычисляемые в момент обращения к правилу. Правило применяется только в том случае, если выполнено условие его применимости. Условие применимости считается выполненным, если значение соответствующего выражения больше 0.

Для определения значения одной цели разработчик экспертной системы может задавать несколько правил, образующих в модели упорядоченный список правил вывода данной цели. Порядок правил в списке отражает порядок их рассмотрения во время решения задачи.

При вводе описания модели проблемной области в Базу знаний она компилируется, что позволяет сократить время решения задачи и сэкономить оперативную память. В процессе компиляции из числовых атрибутов, утверждений и правил строится сеть вывода, в явном виде включающая все связи между атрибутами и утверждениями, обусловленные правилами вывода. Сеть вывода образует И-ИЛИ граф с вершинами двух видов: вершины первого вида соответствуют простым целям (т.е. числовым атрибутам и утверждениям), а вершины второго вида - простым правилам. Все вершины первого вида относятся к типу ИЛИ; второго – к типу И. Дуги представляют связи между простыми целями и простыми правилами: если простая цель G выводится с помощью правила Ri, то в сети имеется дуга, соединяющая вершины, соответствующие G и Ri, направленная от G к Ri;

если простая цель Hj является подцелью правила Ri, то в сети имеется дуга, соединяющая соответствующие вершины и направленная от Ri к Hj (рис. 3).



где G – простая цель;
R1, R2, ... , Rn – правила вывода значения G;
H1, H2, ... , Hk – подцели правила R1.

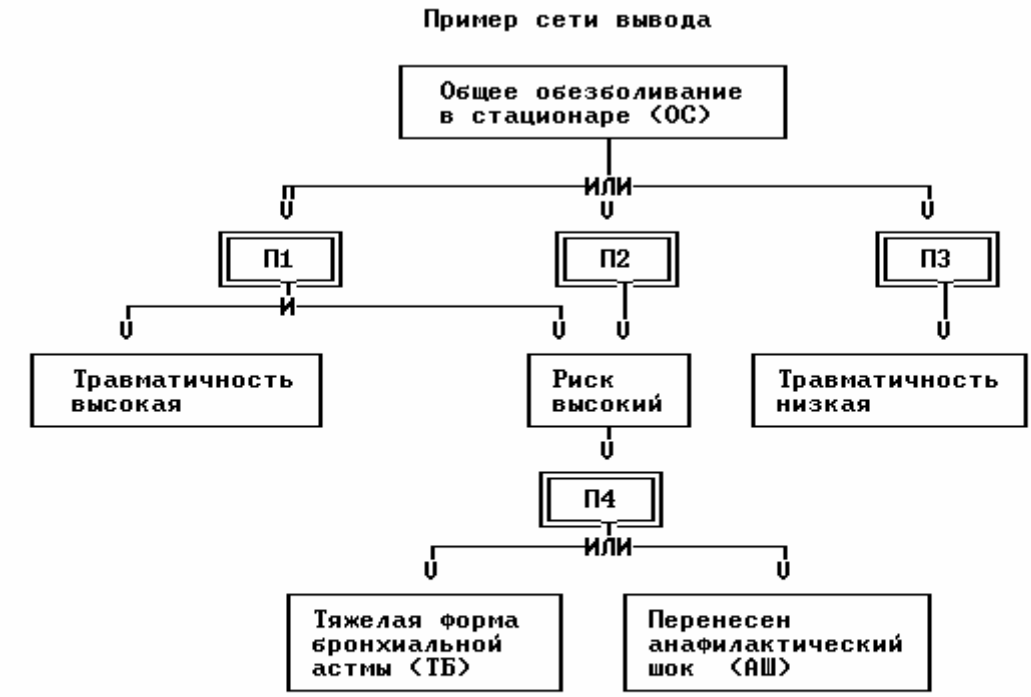
Рис. 3

Сложные правила преобразуются в результате компиляции в совокупности простых, выводящих значения всех утверждений о целевых символьных атрибутах, соответствующих сложным правилам. Особенность таких простых правил состоит в том, что применение одного из них влечет за собой применение остальных.

Сеть вывода строится следующим образом. При вводе новой простой цели (нового числового атрибута или нового значения символьного атрибута) в сеть вывода вводится вершина, соответствующая этой цели.

При вводе простого правила, т.е. правила вывода значения простой цели, в сеть добавляется вершина, соответствующая этому правилу, и дуги, представляющие связи между простой целью и правилом, между правилом и его подцелями. При вводе сложного правила из него создается нескольких простых правил, предназначенных для вывода значений утверждений о сложной цели вводимого правила. Простые правила добавляются описанным ранее образом в сеть вывода.

Пример сети вывода и соответствующих ей правил из области анестезиологии приведен на рис. 4.



П1: ЕСЛИ [Травматичность.высокая] & [Риск.высокий] ТО ОС
П2: ЕСЛИ ~[Риск.высокий] ТО ~ОС
П3: ЕСЛИ [Травматичность.низкая] ТО ~ОС
П4: ЕСЛИ ТБ!АШ ТО [Риск.высокий],

где ! – дизъюнкция, & – конъюнкция, ~ – отрицание.

Рис. 4

В правилах представлены рекомендации относительно целесообразности направления пациента в стационар для проведения хирургического вмешательства под общим обезболиванием. Если травматичность предполагаемого вмешательства высокая, а риск высокий, то рекомендуется проведение общего обезболивания в стационаре. Если риск не является высоким или если травматичность вмешательства низкая, то в проведении общего обезболивания в условиях стационара нет необходимости. Риск считается высоким, если пациент страдает тяжелой формой бронхиальной астмы либо если ранее он перенес анафилактический шок.

Для задания модели проблемной области разработчик должен указать, какие вершины сети вывода являются целевыми в данной модели, какие сообщения необходимо выдать пользователю экспертной системы в зависимости от полученных результатов, а также другую информацию о порядке ведения консультации. Знания этого типа описываются в сценарии, являющимся обязательным компонентом модели. Сценарий представляет

собой последовательность предложений, снабженных условиями применимости. Каждое предложение указывает, какое действие должна выполнить экспертная система в случае применимости этого предложения. Возможно выполнение следующих действий:

- вывести значение цели;
- выдать сообщение пользователю;
- выдать сообщение внешней программе;
- сбросить выведенные результаты;
- перейти к выполнению другого предложения;
- принять информацию от внешней программы;
- внешней программе;
- передать информацию о результатах решения
- создать контрольную точку консультации;
- загрузить контрольную точку;
- обратиться к подмодели, решающей некоторую частную подзадачу, передать ей параметры и получить выведенные в подмодели результаты;
- закончить консультацию с сообщением (СТОП).

1.6. Стратегия решения задач

В начале решения задачи выбирается первое предложение сценария, проверяется условие его применимости и, если условие вы полнено, выполняется указанное в нем действие. Если в ходе проверки условия возникает потребность в значении некоторой цели, анализ условия приостанавливается, и требуемое значение выводится из сети вывода. Далее проверяется условие текущего предложения. Затем осуществляется переход к следующему предложению и т.д., пока не будет обнаружено действие СТОП или пока не будет исчерпан сценарий.

При выводе значения цели делается попытка в соответствии с описанием проблемной области найти в сети вывода путь от вершин,

представляющих исходные данные консультации (т.е. значения, которые вводятся пользователем в ответ на вопросы), к вершине, соответствующей рассматриваемой цели. Этот путь должен обладать следующим свойством: условия применимости правил, соответствующих входящим в путь дугам, должны быть выполнены. При поиске пути используется метод поиска "в глубину" с последующим выводом значений от данных.

Алгоритм вывода значения простой цели состоит в следующем. Из списка правил вывода данной цели выбирается первое правило и делается попытка его применить (при этом проверяется условие его применимости, как это делается для предложений сценария). Если условие выполнилось, правило применяется. Если ни одно из правил не удалось применить, а цель - числовой атрибут, то в качестве значения этого атрибута принимается значение по умолчанию (в том случае, если оно указано разработчиком в описании атрибута), иначе цель помечается как не выводимая.

В том случае, когда оказалось сразу несколько применимых правил, используется первое применимое правило. Значения целей вычисляются один раз и не могут быть изменены иначе как по команде пользователя или с помощью предложения сценария СБРОС.

Исходные цели консультации задаются в сценарии; исходными являются все цели, упоминаемые в предложениях сценария. Остальные цели рассматриваются тогда, когда их значения понадобятся для применения каких-либо правил.

В режиме консультации пользователь может наблюдать ход рассуждения программы комплекса при поиске пути в сети вывода с помощью средств объяснения и трассировки.

1.7. Язык представления знаний

В описании языка представления знаний использованы следующие метасимволы и обозначения:

::= - символы, отделяющие левую часть определения от правой;

<> - скобки, в которые заключаются метапонятия; - символ, разделяющий альтернативы;

[] - скобки, в которые заключается необязательный фрагмент определяемой конструкции;

{} - скобки, обозначающие повторение (ноль или более раз) заключенной в скобки конструкции;

' - апостроф для выделения в языковых конструкциях символов языка представления знаний, совпадающих с метасимволами.

Числовые константы, допустимые в языке представления знаний, - это положительные и отрицательные числа с десятичной точкой и точностью до двух знаков после десятичной точки, не превышающие по абсолютной величине 64000.00. В описании синтаксиса языка представления знаний числовые константы обозначены метапонятием <число>.

Числовые константы, представляющие коэффициенты определенности, являются числовыми константами, не превышающими по абсолютной величине 5.00. В описании синтаксиса они обозначаются метапонятием <коэффициент определенности>.

Числовые и символьные атрибуты, а также значения символьных атрибутов, идентифицируются с помощью имен. Именем может быть любая последовательность, состоящая не более чем из 30 символов, среди которых могут быть:

- латинские прописные и строчные буквы;
- русские прописные и строчные буквы;
- цифры;
- символы: __ (подчеркивание), #.
- имя не должно начинаться с цифры.

Это имя используется в описании модели как идентификатор, но с ним не всегда удобно работать в фазе использования системы при формировании сообщений пользователю, когда может возникнуть необходимость в развернутом имени атрибута, раскрывающем его семантику. Развернутые

имена, представляющие лингвистические знания об атрибутах, задаются в описаниях атрибутов. Длина развернутого имени не должна превышать 120 символов. Например, значение символьного атрибута "метод обезболивания" может иметь развернутое имя "местное обезболивание с премедикацией седуксеном и антигистаминным препаратом". В этом случае в сообщениях, выдаваемых пользователю в ходе консультации, будет использоваться это развернутое имя.

В описании синтаксиса языка представления знаний имена и развернутые имена определяются метапонятиями <имя> и <развернутое имя> соответственно.

Употребляемые в конструкциях языка представления знаний тексты (например, тексты комментариев)- это произвольные последовательности символов ограниченной длины, не содержащие символов-кавычек ("). Эти тексты обозначаются в описании синтаксиса как <текст N>, где N - число, указывающее максимально допустимую длину текста. В примерах конструкций, приведенных в настоящем документе, тексты и развернутые имена заключены в кавычки.

Для представления лингвистических знаний о коэффициентах определенности утверждений и значениях числовых атрибутов используются специальные конструкции, называемые шкалами. Шкалы представляют собой списки равенств вида: "<текст 40>=<число>", ставящих в соответствие лингвистическим (текстовым) описаниям некоторые числовые значения. Равенства в списке разделяются запятыми. Каждая шкала, описанная в модели относится к некоторому числовому атрибуту или утверждению. Например, лингвистические знания о коэффициенте определенности некоторого утверждения могут быть представлены с помощью шкалы:

"да=5, возможно=2.5, да или нет=0, маловероятно=-2.5, нет=-5".

Все числовые константы, упоминаемые в шкале, не должны выходить за пределы диапазона допустимых значений числового атрибута (если шкала относится к числовому атрибуту) или диапазона от -5.00 до 5.00 (если шкала

относится к утверждению). Шкалы могут задаваться как в описаниях атрибутов и утверждений, так и в текстах вопросов к ним.

Шкалы применяются при формировании текстов вопросов и сообщений конечному пользователю.

Шкала используется в вопросе в том случае, если она задана в тексте вопроса или в описании утверждения или атрибута, о котором задается вопрос. При этом из лингвистических описаний, содержащихся в шкале, формируется меню, и конечному пользователю предлагается выбрать одно из этих описаний. В качестве ответа на вопрос принимается то числовое значение, которое соответствует в шкале выбранному лингвистическому описанию. Например, если будет задан вопрос с приведенной выше шкалой, и пользователь выберет ответ "возможно", то в качестве ответа на вопрос будет принято значение 2.50.

Использование шкалы при формировании сообщения о вычисленном коэффициенте определенности утверждения или значении числового атрибута осуществляется в том случае, если в описании этого утверждения или числового атрибута задана шкала. При этом выдается то лингвистическое описание, которому соответствует число, наименее отличающееся от вычисленного значения. Например, при использовании приведенной выше шкалы, для коэффициента определенности 3.0 будет выбрано и помещено в текст сообщения лингвистическое описание "возможно".

В описании синтаксиса языка представления знаний шкалы определяются метапонятием <шкала>.

Формирование текстов вопросов и сообщений осуществляется с использованием шаблонов. Шаблон представляет собой текст, в который могут подставляться сведения об атрибутах и их значениях. Позиции для подстановки отмечаются в шаблоне следующим образом:

^^ - для коэффициентов определенности утверждений и значений числовых атрибутов (или соответствующих им лингвистических описаний, описанных в шкалах);

^s - для имен символьных атрибутов;

^v - для имен значений символьных атрибутов;

^g - для имен простых целей;

^y - для имен утверждений и неразвернутых имен числовых атрибутов.

При заполнении шаблонов предпочтение всегда отдается развернутым именам, если они указаны, и лингвистическим описаниям, если заданы соответствующие шкалы.

В описании синтаксиса языка представления знаний шаблоны определяются метапонятием <шаблон N>, где N - число, указывающее максимально допустимую длину текста шаблона.

Описание модели предметной области включает в себя комментарий к модели, способ разрешения конфликтов, описание атрибутов, правила вывода, а также сценарий консультации.

Синтаксис описания модели имеет следующий вид:

<описание модели> ::= [<комментарий к модели>]

<описание атрибута> { <описание атрибута> }

<описание правила вывода> { <описание правила вывода> }

<предложение сценария> { <предложение сценария> }

<комментарий к модели> ::= <текст 160>

Комментарий к модели представляет собой произвольный текст, длина которого не должна превышать 160 символов.

Для представления количественных и качественных характеристик проблемной области используются соответственно числовые и символьные атрибуты.

Числовые атрибуты представляют собой величины, принимающие значения из некоторого диапазона. Значения числовых атрибутов считаются

точно известными. Аналогом числового атрибута в традиционном программировании является переменная вещественного типа, значение которой может быть получено только один раз и в дальнейшем не изменяется. Значения числовых атрибутов задаются с точностью до двух знаков после десятичной точки.

Задание развернутого имени атрибута не является обязательным. Если разработчик не укажет это имя, в сообщениях будет использоваться имя атрибута.

Описание шкалы начинается со специального символа '$'. Задание шкалы также не является обязательным. Если шкала есть, то при задании вопросов (выдаче сообщений) о значении числового атрибута значения будут запрашиваться (выдаваться) не в числовой форме, а терминах лингвистических описаний, упоминаемых в шкале.

Область допустимых значений числового атрибута описывается в виде диапазона, который задается двумя числовыми константами, представляющими его нижнюю и верхнюю границы. Область допустимых значений не должна выходить за пределы диапазона от -64 000.00 до 64 000.00.

Разработчик экспертной системы может задать значение числового атрибута по умолчанию (не обязательно). Это значение не должно быть выходить за пределы области допустимых значений атрибута "ОТ <число> ДО <число>". Как указано значение по умолчанию принимается при решении задачи в качестве значения числового атрибута в том случае, когда его не удалось вывести другим способом.

Пример описания числового атрибута:

ЧИСЛОВОЙ АТРИБУТ возраст

"возраст пациента"

ОТ 15 ДО 100

ПО УМОЛЧАНИЮ

Символьные атрибуты принимают значения из конечного множества значений, задаваемого перечислением в описании атрибутов:

<описание символьного атрибута> ::=

СИМВОЛЬНЫЙ АТРИБУТ <имя символьного атрибута>

[<развернутое имя символьного атрибута>]

[$ [<шкала>]]

[ШАБЛОН <шаблон утверждений>]

<описание значения> { <описание значения> }

<имя символьного атрибута> ::= <имя>

<развернутое имя символьного атрибута> ::= <развернутое имя>

<шаблон утверждений> ::= <шаблон 180>

<описание значения> ::= ЗНАЧЕНИЕ <имя значения>

[<развернутое имя значения>]

[$ [<шкала>]]

[ИМЯ УТВЕРЖДЕНИЯ <имя>]

[ПО УМОЛЧАНИЮ <коэффициент определенности по умолчанию>]

<имя значения> ::= <имя>

<развернутое имя значения> ::= <развернутое имя>

<коэффициент определенности по умолчанию> ::=

<коэффициент определенности>

Задание развернутого имени атрибута не является обязательным. Если разработчик не укажет это имя, в сообщениях будет использоваться имя атрибута.

В тех случаях, когда всем значениям символьного атрибута соответствует одна и та же шкала, для сокращения записи эту шкалу можно указать один раз в описании символьного атрибута. Если указан только специальный символ '$', то будет использована шкала по умолчанию:

"Да=5, Скорее всего=4, Возможно=2, Да или нет=0, Наверное нет=-2, Маловероятно=-4, Нет=-5".

Шкала, задаваемая в описании значения символьного атрибута, относится только к данному значению. Как и в случае символьного атрибута, можно указать только символ '$', при этом будет использована шкала по умолчанию. Если указаны шкалы как для символьного атрибута, так и для некоторого его значения, то при работе с этим значением шкала для символьного атрибута игнорируется.

Шаблон утверждений используется при формировании сообщений о значениях данного атрибута. В шаблон могут подставляться: имя данного атрибута; имя того значения, сообщение о котором формируется. Если в модели не описан шаблон утверждения, используется стандартный шаблон "^s - ^v".

Для описания действий правил, а также условий применимости правил и предложений сценариев используются арифметические и логические выражения.

Арифметические выражения используются как в правилах, для задания формул, по которым следует вычислять значения числовых атрибутов и утверждений, так и в логических выражениях.

Арифметические выражения строятся стандартным образом с помощью умножения скобок (*), деления (/), , а также следующих и операций сложения (+), вычитания (-) получения остатка от деления нацело (%) арифметических функций:

abs(X) - абсолютное значение X;

exp(X,Y) - число X в степени Y;

sq(X) – квадратный корень из X;

lg(X) – десятичный логарифм X;

inth(X) – округление X в большую сторону;

intl(X) - округление X в меньшую сторону;

sin(X) - синус X(X в радианах);

cos(X) - косинус X(X в радианах);

tg(X) - тангенс X(X в радианах);

ctg(X) - котангенс X(X в радианах);

asin(X) - арксинус X(X в радианах);

atg(X) - арктангенс X(X в радианах);

rad(X) - получение значения в радианах (X в градусах);

grad(X) - получение значения в градусах (X в радианах);

Определены следующие вспомогательные функции:

more_cert(X,A) - количество значений символьного атрибута A, коэффициенты определенности которых не ниже значения X;

less_cert(X,A) - количество значений символьного атрибута A, коэффициенты определенности которых не выше значения X, где значение X задается либо числовой константой, либо числовым атрибутом. Эти функции удобно использовать в тех случаях, когда необходимо оценить выведенные результаты. Например, при решении задачи диагностики можно определить, удалось ли выявить хотя бы несколько диагнозов с достаточной, с точки зрения эксперта, степенью уверенности: если поставлен хотя бы один диагноз, т.е. more_cert(4.0,диагноз) > 0 или все диагнозы отвергнуты, т.е. less_cert(-4.0,диагноз) = N, где N - число возможных диагнозов, то можно закончить решение задачи; иначе необходимо провести более подробный анализ с привлечением дополнительных данных. Другой распространенный случай использования этих функций - оценка результатов тестирования: сколько тестов дали положительный или отрицательный результат.

В арифметических выражениях можно использовать числовые константы и простые цели, т.е. числовые атрибуты и утверждения. Ссылаться на числовые атрибуты можно по именам (не развернутым), на утверждения - по именам утверждений или по парам вида: [<имя символьного атрибута>.<имя значения>].

Арифметические выражения вычисляются слева направо. Порядком выполнения операций можно управлять с помощью круглых скобок.

Логические выражения используются для записи высказываний о состоянии проблемной области - условий применимости правил вывода и

предложений сценария. Значением логического выражения является коэффициент определенности соответствующего ему высказывания.

Логические выражения строятся из высказываний о проблемной области, значениями которых являются коэффициенты определенности, с помощью операций: конъюнкции (&), дизъюнкции (|) и отрицания (~).

В логических выражениях могут быть использованы операции: && и || (конъюнкция и дизъюнкция с принудительным завершением). Эти операции выполняются так же, как & и | соответственно, с единственным отличием: если первый операнд не положителен (в случае &) или положителен (в случае |), то второй операнд не рассматривается, и в качестве результата принимается значение первого операнда.

Как следует из приведенных определений, высказывания о проблемной области могут быть представлены:

- утверждениями;
- обращениями к логическим функциям;
- арифметическими отношениями: равно (=), больше (>) и меньше
- логическими выражениями.

Кроме того, можно задавать коэффициенты определенности с помощью числовых констант, числовых атрибутов и арифметических выражений, если их значения не выходят за пределы диапазона от -5.00 до +5.00.

При вычислении значений логических выражений используются следующие формулы:

D(H1 & H2) = min[D(H1), D(H2)]; (7)

D(H1 | H2) = max[D(H1), D(H2)]; (8)

D(~H) = -D(H), (9)

где H, H1 и H2 - высказывания о проблемной области (утверждения, обращения к логическим функциям и т.д.);

D(H) - коэффициент определенности высказывания H (т.е. его значение).

Для сокращения записи логических выражений в языке представления знаний определены 4 стандартные логические функции, вычисляющие коэффициенты определенности следующих высказываний:

AF(A) - все утверждения о всех возможных значениях атрибута A ложны (All False);

EF(A) - среди всех утверждений A есть ложные (Exist False);

AT(A) - все утверждения о всех возможных значениях символьного атрибута A истинны (All True);

ET(A) - среди всех утверждений о значениях символьного атрибута A есть истинные (Exist True).

Перечисленные логические функции могут быть представлены следую образом:

AF(A) = ~a1 & ~a2 & … & ~aN,

EF(A) = ~a1 | ~a2 | … | ~aN,

AT(A) = a1 & a2 & … & aN,

ET(A) = a1 | a2 & … | aN,

где a1, a2, . aN - утверждения о значениях символьного атрибута A. Следует отметить, что в функциях ET и EF рассматриваются только те из всех возможных утверждений об атрибуте A, коэффициенты определенности которых удалось вывести. Для вычисления значений функций AT и AF необходимо рассмотрение всех возможных утверждений.

L-S функции используются при описании нечетких понятий. Каждая L-S функция предоставляет эксперту стандартные средства для вычисления того коэффициента определенности, с которым значение числового атрибута удовлетворяет нечеткому понятию. Таким образом, эти функции обеспечивают параметрическое задание функций принадлежности нечетких понятий в терминах коэффициентов определенности.

Функции L_ll, L_rr, L_lr являются линейными, а функции S_ll, S_rr и S_lr - квадратичными, что обозначается первыми символами в их именах (L и S соответственно).

Функции L_ll, L_rr, L_lr задаются с помощью параметров A, B, C и D (рис. 5), которые должны удовлетворять условию A < B =< C < D, следующим образом (X - аргумент функции):

$$L\_ll(A,B,X) = \begin{cases} -5, & \text{если } X <= A; \\ \dfrac{10*X - 5*A - 5*B}{B - A}, & \text{если } A < X <= B; \\ 5, & \text{если } X > B, \end{cases}$$

$$L\_rr(C,D,X) = \begin{cases} 5, & \text{если } X <= C; \\ \dfrac{-10*X + 5*C + 5*D}{D - C}, & \text{если } C < X <= D; \\ -5, & \text{если } X > D, \end{cases}$$

$$L\_lr(A,B,C,D,X) = \begin{cases} L\_ll(A,B,X), & \text{если } X < C; \\ L\_rr(C,D,X), & \text{если } X >= C. \end{cases}$$

Например, понятие "около 7" может быть описано как L_lr(5,7,7,9,X). При X, равном 6.5, коэффициент определенности утверждения о том, что X имеет значение около 7, равен 2.5.

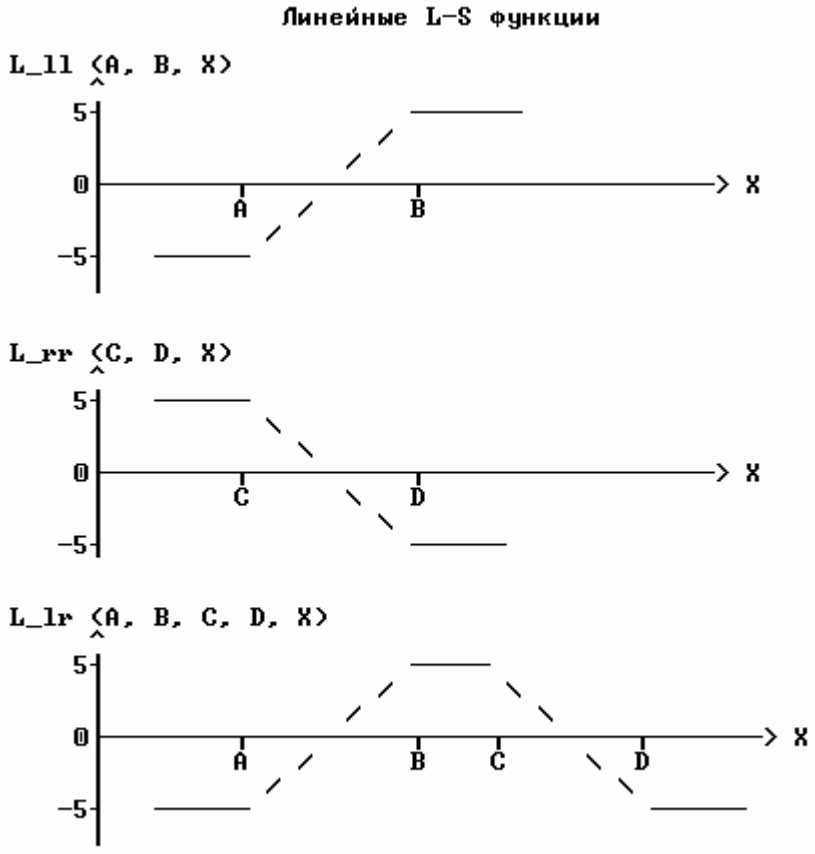


Рис. 5.

Функции S_ll, S_rr, S_lr задаются с помощью параметров A, B, C, D, E и F (рис. 6), которые должны удовлетворять условию A =< B =< C =< D =< E =< F, A < C , D < F, следующим образом (X - аргумент функции):

$$S\_ll(A,B,C,X) = \begin{cases} -5, & \text{если } X <= A; \\ 5*\left[\frac{X-A}{B-A}\right]^2 - 5, & \text{если } A < X <= B; \\ 5 - 5*\left[\frac{C-X}{C-B}\right]^2, & \text{если } B < X <= C; \\ 5, & \text{если } X > C, \end{cases}$$

$$S\_rr(D,E,F,X) = \begin{cases} 5, & \text{если } X <= D; \\ 5 - 5*\left[\frac{X-D}{E-D}\right]^2, & \text{если } D < X <= E; \\ 5*\left[\frac{F-X}{F-E}\right]^2 - 5, & \text{если } E < X <= F; \\ -5, & \text{если } X > F, \end{cases}$$

$$S\_lr(A,B,C,D,E,F,X) = \begin{cases} S\_ll(A,B,C,X), & \text{если } X < D; \\ S\_rr(D,E,F,X), & \text{если } X >= D. \end{cases}$$

Например, если понятие "около 7" описано как S_lr(5,6,7,7,8,9,X), то при X, равном 6.5, коэффициент определенности утверждения о том, что X имеет значение около 7, равен 4.75.

С помощью правил вывода разработчик экспертной системы указывает, каким образом должны вычисляться значения атрибутов и утверждений. Числовой атрибут или утверждение, значение которого вычисляется с помощью некоторого правила, называется целью этого правила. Числовые атрибуты и утверждения являются простыми целями, а символьные атрибуты (т.е. множества утверждений) - сложными. В зависимости от целей все правила подразделяются на простые и сложные.

Для вычисления значения простой цели можно указывать несколько правил, которые образуют упорядоченный список правил вывода данной цели. Правила помещаются в список в том порядке, в каком они описаны в

модели. Список правил используется в режиме консультации при попытке вывести значение данной цели.

Сложные правила в результате компиляции знаний преобразуются в совокупности простых, предназначенных для вывода утверждений о значениях соответствующего символьного атрибута. Особенность этих простых правил состоит в том, что применение одного из них в ходе решения задачи влечет за собой применение всех остальных.

В языке представления знаний определены следующие типы правил:

- простой вопрос;
- сложный альтернативный вопрос;
- сложный дистрибутивный вопрос;
- арифметическое правило;
- логическое правило;
- байесовское правило.

Правила-вопросы позволяют запрашивать данные о конкретной ситуации в проблемной области - исходные данные консультации. Возможны два способа ввода этих данных: посредством задания вопроса пользователю или обмена информацией с внешними программами. Выбор способа ввода данных определяется разработчиком ЭС.

Простой вопрос позволяет получать либо значение числового атрибута, либо коэффициент определенности отдельного утверждения. Сложный вопрос позволяет получить распределение коэффициентов определенности по всем возможным значениям символьного атрибута. Альтернативный вопрос используется в тех случаях, когда известно, что символьный атрибут имеет точно одно значение из множества возможных значений. Дистрибутивный вопрос используется в тех случаях, когда символьный атрибут может иметь одновременно несколько или ни одного значений.

Арифметические, логические и байесовские правила относятся к простым правилам.

Арифметические правила предназначены для вычисления значений числовых атрибутов, а также для получения коэффициентов определенности утверждений по эвристическим формулам, предложенным экспертами.

Логические правила предназначены для вычисления коэффициентов определенности утверждений по формулам нечеткой логики. Они применяются только в случае выполнения условия применимости, при этом значение соответствующего логического выражения становится значением утверждения - цели правила.

Байесовские правила применяются для вычисления коэффициентов определенности тех утверждений, об истинности которых можно судить по выполнению ряда факторов (симптомов), имеющих разную значимость.

Именами правил могут быть произвольные последовательности сим волов длиной до 20 байт. Имена правил должны быть уникальными. Первый символ имени идентифицирует тип правила.

Комментарии к правилам - это произвольные тексты длиной до 300 байт, которые используются в режиме консультации для формирования объяснений.

Целью простого правила может быть числовой атрибут или утверждение, сложного - символьный атрибут.

Условие применимости правила - это логическое выражение, значение которого вычисляется в момент обращения к правилу.

Действие правила указывает, каким образом должно вычисляться значение цели правила в том случае, когда правило применимо. Вид действия правила определяется типом правила.

В случае простого вопроса действие правила имеет вид:

<действие правила> ::= [<шаблон>] [$ [<шкала>] | @ <текст 299>

Действием для правила типа "простой вопрос" является текст сообщения пользователю или внешней программе. Текст сообщения пользователю формируется с помощью шаблона, в который могут подставляться: имя символьного атрибута, имя значения этого атрибута и

имя утверждения (составленного с помощью шаблона утверждений для данного атрибута), если вопрос относится к утверждению; имя числового атрибута, если вопрос относится к числовому атрибуту. Если текст не указан, то в режиме консультации он формулируется автоматически. Стандартный текст вопроса о значении числового атрибута имеет вид: "Каково значение атрибута ^g ?", При построении вопроса об утверждении текст вопроса имеет вид "Насколько Вы уверены в том, что ^g ?", В простом вопросе можно указать шкалу, в соответствии с которой пользователь может вводить ответ не в числовой форме, а с помощью некоторых лингвистических описаний. Шкала отделяется от текста основного вопроса символом '$'. В случае утверждений может быть использована стандартная шкала: для этого достаточно указать символ '$' без задания самой шкалы. Если для утверждения или числового атрибута, о котором задается вопрос, уже определена шкала, она будет проигнорирована.

Если для ответа на вопрос требуется вызвать внешнюю программу, в качестве первого символа текста сообщения следует поставить специальный символ '@', тогда остальной текст будет интерпретироваться как командная строка.

В случае сложного альтернативного или дистрибутивного вопроса действие правила имеет вид:

<действие правила> ::= [<шаблон>] [$ [<шкала>] | @ <текст 299>

Действие для правил типа "сложный альтернативный вопрос" и "сложный дистрибутивный вопрос" определяются так же, как для простых вопросов. Если шаблон сообщения пользователю опущен, то используется стандартный шаблон: "Каково значение атрибута ^s ?". Шкалы могут использоваться только в дистрибутивных вопросах. Помимо текста вопроса пользователю во время консультации будет выдан список развернутых имен значений атрибута. Если вопрос альтернативный, пользователю предлагается выбрать одно из значений. После того как пользователь укажет выбранный вариант, соответствующее утверждение получает значение 5.00, а

утверждения обо всех остальных значениях символьного атрибута - значение -5.00. Если вопрос дистрибутивный, пользователю предлагается задать коэффициенты определенности для значений из списка. Если пользователь не укажет коэффициенты для некоторых значений, будет сделана попытка вывести значения соответствующих утверждений (когда в них будет возникать потребность) с помощью других правил вывода, если они есть.

Передача данных от внешней программы в ЭС производится через файл EKO.ANS в соответствии с требованиями.

В случае арифметического правила действие имеет вид:

<действие правила> ::= <арифметическое выражение>

Действием для арифметического правила должно быть арифметическое выражение, значение которого, в случае когда правило применимо и все упоминаемые в выражении подцели выведены, присваивается цели правила. Если вычисленная с помощью данного выражения величина выходит за пределы допустимого для цели диапазона, в качестве значения этой цели принимается соответствующая граница диапазона.

В случае логического правила действие имеет вид:

<действие правила> ::= [<арифметическое выражение>]

Действием в логических правилах может быть арифметическое выражение. Значение этого выражения не должно превышать по абсолютной величине 1.00. Это выражение называется коэффициентом определенности правила и используется в случае отсутствия полной уверенности эксперта в надежности данного правила.

Задание действия логического правила не обязательно. Если разработчик опускает действие, то по умолчанию коэффициент определенности правила принимается равным 1.0. Если в результате вычисления значения действия правила получается недопустимая величина, то в качестве коэффициента определенности правила принимается:

1.00 - в случае положительного результата;

-1.00 - в случае отрицательного результата.

В отличие от всех остальных правил задание условия применимости для логических правил является обязательным.

Применение логического правила осуществляется следующим образом: вычисляется условие применимости, и если оно положительно, то его значение, умноженное на коэффициент определенности правила, принимается в качестве значения цели правила; в противном случае правило считается неприменимым.

В случае байесовского правила действие имеет вид:

<действие правила>:= <описание фактора> {<описание фактора>}

<описание фактора> := <ссылка на утверждение> <разделитель>
<вес> <разделитель> <вес> <разделитель>

<разделитель> ::= <пробелы> | <запятая>

<вес> ::= <число> | <имя числового атрибута>

Действие байесовского правила задается в виде таблицы с тремя столбцами. В первом столбце перечисляются утверждения-факторы, коэффициенты определенности которых влияют на коэффициенты определенности целевого утверждения.

Во втором столбце указываются подтверждающие, а в третьем - опровергающие веса факторов. Подтверждающий вес утверждения-фактора должен быть равен такому коэффициенту определенности, который, по мнению эксперта, следует приписать целевому утверждению в случае истинности утверждения-фактора. Опровергающий вес равен коэффициенту, приписываемому целевому утверждению экспертом в случае ложности утверждения-фактора.

При работе с байесовскими правилами коэффициент определенности D(H) высказывания H связывается с вероятностью того, что это высказывание истинно в рассматриваемой ситуации в проблемной области, следующим образом:

$D(H) = 10\ P(H) - 5,$

где D(H) - коэффициент определенности H;

P(H) вероятность истинности H.

Коэффициент определенности целевого утверждения байесовского правила вычисляется следующим образом. Если подтверждающий вес некоторого фактора равен 5.0 (-5.0), т.е. наличие фактора точно свидетельствует в пользу (против) целевого утверждения, а коэффициент определенности этого фактора, вычисленный в ходе решения задачи, равен 5.0, то целевое утверждение считается точно истинным (точно ложным). Если опровергающий вес некоторого фактора равен 5.0 (-5.0), т.е. отсутствие фактора свидетельствует в пользу (против) целевого утверждения, а вычисленный коэффициент определенности этого фактора равен -5.0, то целевое утверждение считается точно истинным (точно ложным). В противном случае вычисляются величины, представляющие свидетельства факторов в пользу или против целевого утверждения, а затем, для получения коэффициента определенности целевого утверждения, происходит объединение полученных свидетельств. При вычислении свидетельств используются следующие формулы:

$$De(Hi) = \begin{cases} Danp(G) + \dfrac{Ai - Danp(G)}{5 - Danp(Hi)} * (D(Hi) - Danp(Hi)), \\ \qquad\qquad \text{если } D(Hi) >= Danp(Hi); \\ Danp(G) + \dfrac{Danp(G) - Bi}{5 + Danp(Hi)} * (D(Hi) - Danp(Hi)), \\ \qquad\qquad \text{если } D(Hi) < Danp(Hi), \end{cases}$$

где De(Hi) – свидетельство i-го фактора;

Danp(G) - априорное значение коэффициента определенности целевого утверждения;

Danp(Hi) - априорное значение коэффициента определенности фактора Hi;

D(Hi) - текущее значение коэффициента определенности i-го фактора;

Ai - подтверждающий вес фактора i-го фактора;

Bi - опровергающий вес фактора i-го фактора.

Свидетельства объединяются следующим образом:

$$D(G/H1,H2,\ldots,HM) = 5 * \frac{De - 1}{De + 1},$$

где

$$De = \frac{5 + Danp(G)}{5 - Danp(G)} * I(H1) * I(H2) * \ldots * I(HM);$$

$$I(Hi) = \frac{5-Danp(G)}{5+Danp(G)} * \frac{5+De(Hi)}{5-De(Hi)}.$$

Сценарий консультации представляет собой последовательность предложений, определяющую порядок действий при решении задачи.

Синтаксис описания предложения сценария имеет следующий вид:

<предложение сценария> ::=

<имя предложения> <имя действия> <параметры действия>

[ЕСЛИ <условие применимости предложения>]

[<комментарий>]

<имя предложения> ::= <имя>

<имя действия> ::= ЦЕЛЬ | РЕЗУЛЬТАТ | СООБЩЕНИЕ | ПЕРЕХОД |
СБРОС | СБРОС_ВЫВОДА | СТОП | ПРИЕМ | ВОЗВРАТ |
ВЫПОЛНИТЬ | ЗАДАЧА | АРГУМЕНТЫ |
ЗАГРУЗИТЬ_ТОЧКУ | СДЕЛАТЬ_ТОЧКУ | ВЫХОД |
СДЕЛАТЬ_ТОЧКУ_ВЫХОД | НАСТРОЙКА

<условие применимости предложения> ::= <логическое выражение> |
<ссылка на условие предыдущего действия>

<ссылка на условие предыдущего действия> ::= "-"

<комментарий> ::= <текст 300>

Именем предложения может быть любая последовательность, содержащая до 20 символов, без пробелов. Например, в качестве имени можно использовать номер предложения.

Условие применимости можно задавать либо так же, как в правилах либо в виде ссылки на условие предыдущего действия. В первом случае

проверка условия применимости осуществляется аналогично проверке условий в правилах. Во втором случае действие выполняется только тогда, когда выполнилось условие применимости предыдущего предложения, при этом считается, что условие применимости данного предложения также выполнилось.

Вид параметров действия зависит от типа действия.

В шаблонах сообщений, используемых в сценарии, предусмотрена дополнительная возможность подстановки имени наиболее вероятного значения символьного атрибута: из всех значений выбирается то, у которого коэффициент определенности наибольший, а если таких значений несколько, то выбирается первый по порядку. Для задания такой ссылки необходимо в символах, отмечающих позицию для подстановки, поставить знак '!' (восклицательный знак) после '^', а в списке имен для подстановки указать имя символьного атрибута.

Параметры действия **ЦЕЛЬ** определяются следующим образом:

<параметры действия ЦЕЛЬ> ::= { <имя цели> , } <имя цели>

Параметры действия ЦЕЛЬ задают в виде списка имен, разделенных запятыми или пробелами, исходные цели консультации, а также порядок их рассмотрения. Параметрами могут быть утверждения, числовые и символьные атрибуты.

Параметры действия **РЕЗУЛЬТАТ** определяются следующим образом:

<параметры действия РЕЗУЛЬТАТ> ::= [<шаблон 1600>,]<уровень>,
{ <имя цели> , } <имя цели>

<уровень> ::= <коэффициент определенности | <имя простой цели>

<имя цели> ::= <имя простой цели> | <имя символьного атрибута>

Действие РЕЗУЛЬТАТ задает список целей, сообщения о значении которых должны быть выданы на экран, а также уровень выдачи результатов. Сообщения будут выдаваться только о тех утверждениях, коэффициенты определенности которых не ниже уровня выдачи. Сообщения выдаются в виде списка развернутых имен целей и их значений. Если параметры

действия содержат символьный атрибут, то утверждения о данном атрибуте при выдаче упорядочиваются по убыванию коэффициентов определенности. Утверждения о значениях символьного атрибута строятся исходя из шаблона утверждений и развернутых имен.

Сообщения о результате могут выдаваться по шаблону выдачи, задаваемому в качестве первого (необязательного) параметра. Если шаблон выдачи не указан, используется стандартный.

Перед выдачей сообщения просматривается список целей-параметров выводятся те из них, значения которых еще не были получены.

Параметры действия **СООБЩЕНИЕ** определяются следующим образом:

<параметры действия СООБЩЕНИЕ> ::= <шаблон 1600>{,<имя простой цели>}

Действие СООБЩЕНИЕ, как и действие РЕЗУЛЬТАТ, предназначено для описания сообщений конечному пользователю, однако в действиях данного типа разработчик может указывать произвольный текст сообщения. Текст задается с помощью шаблона.

Первым параметром действия является шаблон сообщения, за ним должны быть указаны простые цели в том порядке, в каком их значения должны подставляться в сообщение. Если к моменту выполнения действия какие-либо из требуемых значений не будут известны, то сначала будет произведен вывод их значений и только после этого будет выдано сообщение. Если для каких-либо простых целей в списке параметров определены шкалы, то выдача их значений будет осуществляться в символьном виде в соответствии с этими шкалами.

Если требуется вывести сообщение не на экран монитора, а передать внешней программе (через файл EKO.MSG), то перед текстом сообщения ставится специальный символ '$'.

Если после выдачи текста сообщения требуется приостановить консультацию (сделать паузу), то перед текстом сообщения ставится

специальный символ '@'. Эта возможность используется для того, чтобы обратить на сообщение внимание пользователя экспертной системы. В этом случае для продолжения консультации следует нажать любую клавишу (Все сказанное справедливо и по отношению к действиям СТОП и РЕЗУЛЬТАТ).

Параметр действия **ПЕРЕХОД** определяется следующим образом:

<параметры действия ПЕРЕХОД> ::= <имя предложения>

Действие ПЕРЕХОД позволяет осуществлять условный переход в сценарии. Параметром действия является имя предложения, к которому необходимо перейти.

Параметр действия **СБРОС** определяется следующим образом:

<параметры действия СБРОС> ::= [{ <имя цели> , } <имя цели>]

Действие СБРОС предназначено для отмены тех значений, которые указаны в качестве параметров этого действия. Если параметры не указаны, происходит отмена всех значений.

Действие **СБРОС ВЫВОДА** предназначено для отмены всех значений, кроме тех, которые были введены пользователем в ответ на вопросы. Это действие не требует параметров.

Параметры действия **СТОП** определяются следующим образом:

<параметры действия СТОП> ::= [<шаблон 1600> {, <имя простой цели> }]

Действие СТОП предназначено для принудительного прекращения решения задачи. В качестве параметров действия можно задать описание заключительного сообщения. Параметры данного действия задаются так же, как и в случае действия СООБЩЕНИЕ. Если параметры не заданы, заключительное сообщение не выдается. Решение задачи заканчивается также тогда, когда достигнут конец сценария (нет следующего предложения).

Действие **ПРИЕМ** предназначено для приема исходных данных из файла входных данных, имя которого задается в качестве параметра действия. Файл входных данных может быть подготовлен внешней программой или вручную. Информация, содержащаяся в файле входных

данных, представляет собой текст. Каждая строка этого текста задает значение, которое следует присвоить некоторой цели. Строка определяется следующим образом:

<строка файла входных данных> ::= <имя простой цели>=<значение>

<значение> ::= <число>

Если диапазон значений коэффициентов во внешнем файле отличается от [-5.0, 5.0] (например, используется диапазон [0.0, 1.0] или [-100.0, 100.0]), то при выполнении действия пропорциональное преобразование коэффициентов может осуществляться автоматически - для этого достаточно в качестве второго и третьего параметров этих действий указать нижнюю и верхнюю границы диапазона, используемого во внешнем файле.

При выполнении данного действия каждой цели, указанной в строке файла входных данных и к настоящему моменту не выведенной, присваивается соответствующее значение. Если значение выходит за пределы допустимого диапазона для данной цели, оно игнорируется. Строки с неверно заданными именами простых целей не обрабатываются.

Параметры действия **ВОЗВРАТ** определяются следующим обра-

<параметры действия ВОЗВРАТ> ::= <имя файла>, <уровень>
[,<нижняя граница>,
<верхняя граница>],
<имя простой цели>
{,<имя простой цели>}

<нижняя граница> ::= <число>

<верхняя граница>::= <число>

<имя простой цели> ::= <имя числового атрибута>
<ссылка на утверждение>

<имя числового атрибута> ::= <имя>

<ссылка на утверждение> ::= <имя утверждения> |
'['<имя символьного атрибута>.<имя значения>']'

<имя утверждения> ::= <имя>

<имя символьного атрибута> ::= <имя>

<имя значения> ::= <имя>

Действие ВОЗВРАТ аналогично действию РЕЗУЛЬТАТ. Разница заключается в следующем:

1) результаты решения задачи выдаются не на экран дисплея, а в файл выходных данных, имя которого задается в параметрах действия.
2) вместо развернутых имен целей всегда используются имена целей.
3) если диапазон значений коэффициентов во внешнем файле отличается от [-5.0, 5.0] (например, используется диапазон [0.0, 1.0] или [-100.0, 100.0]), то при выполнении действия пропорциональное преобразование коэффициентов может осуществляться автоматически -для этого достаточно в качестве второго и третьего параметров этих действий указать нижнюю и верхнюю границы диапазона, используемого во внешнем файле.

Файл выходных данных имеет формат, полностью аналогичный формату файла входных данных.

Действие **ВЫПОЛНИТЬ** предназначено для выполнения команды операционной системы. Шаблон используется так же, как и в действии СООБЩЕНИЕ.

Параметры действия **ЗАДАЧА** определяются следующим образом:

<параметры действия ЗАДАЧА> ::= <имя модели> [{,<простой параметр>}]

<простой параметр> ::= <имя простой цели> | <число>

Действие ЗАДАЧА описывает обращение одной модели к другой для решения некоторой частной задачи, как это имеет место в традиционном программировании при вызове подпрограммы. Первый параметр указывает имя вызываемой модели (к началу консультации эта модель должна находится в Базе знаний). Далее перечисляются простые цели, с помощью которых происходит обмен данными между вызывающей и вызываемой моделями. Параметрами могут быть простые цели и числовые константы.

При вызове модели происходит передача не только значений, но и имен (включая развернутые) простых целей, являющихся параметрами действия ЗАДАЧА. Если простая цель является числовым атрибутом, в вызываемую модель передается также значение по умолчанию и диапазон возможных значений. Если простая цель является утверждением, будут переданы имена (простое и развернутое) соответствующего символьного атрибута, а также шаблон утверждения.

Если значение переданной в модель цели не было известно в момент обращения, но оказалось вычисленным к моменту возвращения в вызывающую модель, результат вычисления будет передан в вызывающую модель. Значения целей, вычисленных к моменту обращения, не могут быть изменены в вызванной модели.

Параметры действия **АРГУМЕНТЫ** определяется следующим образом:

<параметры действия АРГУМЕНТЫ> ::=

{ <имя простой цели> , } <имя простой цели>

Данное действие предназначено для описания тех простых целей, которые данная модель принимает при ее вызове из другой модели. Оно должно быть первым в сценарии. При вызове модели перечисленные в действии АРГУМЕНТЫ простые цели заменяются целями из вызывающей модели. Если в подмодель передавалось число, то передается только значение подцели, а имя ее не изменяется. Имена простых целей в подмодели должны быть не короче имен соответствующих целей в вызывающей модели.

Параметр действия **СДЕЛАТЬ_ТОЧКУ** определяется следующим образом:

<параметр действия СДЕЛАТЬ_ТОЧКУ> ::= [<имя файла>]

Текст указывает имя файла, в котором должна быть создана контрольная точка. Если текст опущен, в качестве имени принимается имя текущей модели с расширением ".PNT". При создании контрольной точки происходит запоминание во внешнем файле всех выведенных к текущему

моменту значений, включая значения в моделях, вызвавших данную, если такие имеются. Таким образом, в отличие от действия ВОЗВРАТ, удается передать во внешний файл значения из всех выполняемых моделей.

Внешний файл представляет собой последовательность строк, описывающих выполняемые модели. Модели описываются в порядке их вызова: сначала - исходная модель, далее - модель, вызванная из исходной, и т.д., до текущей модели. Описание каждой модели начинается с заголовка, состоящего из двух строк:

<первая строка заголовка> ::= '>'<имя модели>

<вторая строка заголовка> ::= ПРЕДЛОЖЕНИЕ = <номер>

где номер указывает, какое по порядку (начиная с первого) предложение сценария выполняется в модели в момент создания контрольной точки. Остальные строки содержат описания данных, и их структура определяется следующим образом:

<строка описания данных> ::= <имя простой цели>=<значение>

<значение> ::= <число>

Параметр действия **ЗАГРУЗИТЬ_ТОЧКУ** определяется следующим образом:

<параметр действия ЗАГРУЗИТЬ_ТОЧКУ> ::= [<имя файла>]

Текст указывает имя контрольной точки; если текст опущен, будет загружаться точка с именем текущей модели и расширением ".PNT". Выполнение данного предложения приводит к загрузке из внешнего файла информации о значениях простых целей в модели. Внешний файл может быть либо получен в результате выполнения действия СДЕЛАТЬ ТОЧКУ (при этом он доступен для последующей корректировки пользователем), либо создан вручную. Поэтому все загруженные значения рассматриваются в ходе решения задачи как полученные от пользователя.

Параметр действия **ВЫХОД** определяется следующим образом:

<параметр действия ВЫХОД> ::= [<число>]

Действие ВЫХОД предназначено для принудительного прекращения консультации. Если не задан параметр действия то после выполнения данного предложения управление получат средства просмотра оперативного протокола консультации. Если параметр задан (какое либо число), то произойдет выход из Консультатора.

Параметр действия **СДЕЛАТЬ_ТОЧКУ_ВЫХОД** определяется следующим образом:

<параметр действия СДЕЛАТЬ_ТОЧКУ_ВЫХОД> ::= [<имя файла>]

Данное предложение аналогично предложению СДЕЛАТЬ ТОЧКУ и отличается от него только тем, что после создания контрольной точки следует немедленное завершение консультации. Это удобно использовать для описания прерывания консультации (и сохранения всех выведенных результатов) с возможностью возобновления решения задачи с момента прерывания.

Параметр действия **НАСТРОЙКА** определяется следующим образом:

<параметр действия НАСТРОЙКА> ::= <шаблон 1600> [{,<имя простой цели>}]

С помощью этого действия разработчик ЭС может задавать реакцию Консультатора на клавиши F9 и F10 в ходе решения задачи.

Параметры действия НАСТРОЙКА аналогичны параметрам действия СООБЩЕНИЕ. В шаблоне сообщения указывается, какая команда DOS должна быть выполнена в том случае, когда пользователь, вместо ответа на вопрос, нажмет клавишу F9 или F10. Описание реакции на каждую клавишу начинается в шаблоне сообщения с первой позиции строки и имеет вид:

F<номер клавиши>: <шаблон команды DOS>

где номер клавиши - число 9 или 10, а шаблон команды содержит не более 160 символов. Для обозначения имени активной цели (т.е. той, к которой задается вопрос) используются символы "^x".

По умолчанию клавиша F9 не используется, а по клавише F10 выполняется команда DOS: EHELP ^x exp lst.pic.

Настройка клавиш осуществляется при каждом выполнении действия НАСТРОЙКА. При выполнении общего сброса или сброса выведенных результатов отменяется текущая настройка клавиш и принимается настройка по умолчанию.

Данное действие удобно использовать для описания контекстно-зависимой помощи в ходе консультации. По одной из клавиш можно получать, например, помощь для начинающего пользователя ЭС, а по другой - сжатую справочную информацию для специалиста.

1.8. Режим приобретения знаний

Режим приобретения знаний предназначен для ввода описаний моделей в Базу знаний с проведением семантико-синтаксического контроля вводимой информации. Этот режим используется разработчиком в процессе создания и модификации экспертной системы.

В режиме приобретения знаний средства комплекса позволяют осуществлять диалоговый ввод и корректировку Базы знаний с семантико-синтаксическим контролем вводимой информации, тестировать полученные модели и компилировать введенные знания.

Ввод и коррекция описаний осуществляется с помощью шаблонов ввода, соответствующих основным конструкциям языка представления знаний. Шаблоны ввода изображаются на экране дисплея в виде окон.

В режиме приобретения знаний обеспечивается работа с Базой знаний как с иерархией элементов различных типов. Определены следующие типы элементов:

- модели;
- предложения сценария
- числовые атрибуты;
- символьные атрибуты;
- значения символьных атрибутов;
- правила.

База знаний представляет собой совокупность различных моделей. Каждая модель включает в себя описания предложений сценария, числовые и символьные атрибуты. Описания символьных атрибутов включают в себя описания значений (т.е. утверждений о состоянии проблемной области). Атрибутам и утверждениям соответствуют правила вывода их значений. Сложные правила соответствуют символьным атрибутам, простые - числовым и утверждениям.

Структура диалога по приобретению знаний отражает синтаксическую структуру Базы знаний. В ходе диалога пользователь может работать на различных уровнях иерархии, при этом состояние диалога и допустимые действия пользователя определяются тем уровнем, на котором ведется работа.

В начальном состоянии диалога пользователю предлагается работа с оглавлением Базы знаний - списком имен и комментариев к моделям, находящимся в Базе знаний. Пользователь может выбрать модель, при этом диалог переходит в состояние работы с моделью в целом. В этом состоянии пользователь может устанавливать режим разрешения конфликтов, запускать средства проверки целостности или средства распечатки текста модели, отменять сделанные в ходе работы с данной моделью изменения, а также выбирать тип элементов, с которыми он хочет работать. В последнем случае произойдет переход к более низкому уровню иерархии. Например, если пользователь выберет числовые атрибуты, то он перейдет к работе с числовыми атрибутами данной модели. Затем пользователь может спуститься по иерархии и начать работу с правилами вывода значения какого-либо числового атрибута. Потом он может вернуться к работе с числовыми атрибутами данной модели.

В каждом состоянии диалога (кроме состояния работы с моделью в целом) пользователь может:

- работать со списком элементов текущего уровня, относящихся к одному и тому же элементу предыдущего уровня;

- просматривать и корректировать описание одного элемента.

При работе со списком выводится на экран терминала список имен соответствующих элементов, выделяется текущий элемент и высвечивается текст комментария к нему. Выделение текущего элемента называется установкой внимания на этом элементе. При работе со списком пользователь имеет возможность:

- перемещать, переименовывать или уничтожать текущий элемент;
- создавать новые элементы;
- корректировать текст комментария текущего элемента;
- переходить к работе с описанием текущего элемента;
- переходить к работе со списком элементов более низкого уровня, относящихся к текущему элементу (если они есть);
- перемещать внимание по элементам списка;
- возвращаться к предыдущему уровню иерархии.

Например, при работе с числовыми атрибутами, описанными в модели, пользователь может просматривать список имен этих атрибутов, переименовывать текущий атрибут, вводить или корректировать его развернутое имя, просматривать и корректировать описание текущего атрибута, переходить к работе со списком правил вывода значения текущего атрибута и т.д.

При работе с описанием отдельного элемента на экран терминала выводится шаблон описания этого элемента, соответствующий синтаксису языка представления знаний. Пользователь может заполнить или откорректировать поля данного шаблона. При вводе описаний элементов осуществляется контроль вводимой информации и в случае обнаружения синтаксических ошибок пользователю выдаются сообщения о них.

На последовательность ввода описаний элементов модели накладывается единственное ограничение, состоящее в том, что в описании вводимого элемента не допускаются ссылки на элементы, описания которых

еще не введены. Например, прежде чем ввести предложение сценария с параметром " диагноз", необходимо ввести описание атрибута "диагноз".

Описание модели может вводиться по частям, поэтому при вводе может возникнуть потеря целостности модели. Модель считается целостной, если выполнены следующие условия:

- введен сценарий консультации;
- во всех предложениях сценария определены действия и, если это необходимо, параметры действий;
- в сценарии консультации нет ссылок на несуществующие предложения;
- для каждой простой цели указано хотя бы одно правило вывода ее значения;
- для каждого символьного атрибута указано хотя бы одно значение;
- для всех правил, атрибутов и значений символьных атрибутов указаны имена;
- для всех числовых атрибутов указан диапазон допустимых значений;
- в правилах определены типы и в соответствии с этими типами - условия применимости и действия;
- в правилах и предложениях сценария не упоминаются не описанные в данной модели цели.

Контроль синтаксической корректности арифметических и логических выражений, а также контроль отсутствия в них ссылок на неописанные цели осуществляются непосредственно при вводе выражений в Базу знаний. Остальные условия целостности проверяются отдельно по завершении ввода описания модели с помощью средств тестирования.

В одной модели пользователь может описать не более:

- 500 (30-учебный вариант) утверждений о значениях символьных атрибутов ;
- 100 символьных атрибутов; □

- 300 числовых атрибутов;
- 1000 (30-учебный вариант) правил;
- 100 предложений сценария.

Область данных в описании модели, содержащая тексты имен, сообщений и откомпилированные арифметические и логические выражения, не может превышать 64 тыс. байт.

Режим консультации предназначен для решения задач конечного пользователя на основе моделей в Базе знаний и исходных данных о конкретной ситуации в проблемной области. Этот режим используется разработчиком на этапе тестирования экспертных систем, а также конечным пользователем при решении конкретных задач с помощью разработанных систем.

1.9. Режим консультации

В режиме консультации обеспечиваются следующие возможности:

- решение конкретной задачи на основе выбранной модели с формированием объяснений ЗАЧЕМ в ходе консультации задается тот или иной вопрос;
- просмотр информации об описанных в модели целях, в том числе - объяснений, КАК получены значения этих целей, и изменение значений любых целей;
- просмотр информации о правилах в модели;
- сброс значений всех целей либо значений всех выведенных целей (т.е. отмена всех значений, не являющихся исходными данными);
- получение трассы решения задачи;
- запись протокола консультации в файл;
- создание и загрузку контрольных точек.

Решение задачи осуществляется в ходе диалога экспертной системы с пользователем. На экран выдаются сообщения в соответствии со сценарием консультации, а также задаются вопросы, описанные в применяемых

правилах-вопросах. При задании вопроса пользователю выдается краткое поясняющее сообщение о виде ожидаемого ответа.

Значения числовых атрибутов должны вводиться в виде действительных чисел с точностью до двух знаков после десятичной точки и не выходящих за пределы диапазона допустимых значений этих атрибутов. Диапазоны допустимых значений выдаются пользователю в текстах подсказок во время задания вопросов.

Значения утверждений должны вводиться в виде коэффициентов определенности. Если при задании вопроса пользователю предлагается меню, составленное на основе шкалы, то пользователь имеет возможность как ввести числовое значение коэффициента, так и выбрать в меню лингвистическое описание коэффициента.

При задании сложного вопроса пользователю выводится на экран список значений целевого символьного атрибута. Если вопрос альтернативный, то пользователь в соответствии с поясняющим сообщением системы должен выбрать то значение, которое имеет место в рассматриваемой ситуации. Если вопрос дистрибутивный, то пользователь должен указать коэффициенты определенности (в числовом виде или с помощью меню если имеется шкала), для всех тех значений символьного атрибута, о которых он имеет информацию.

Если пользователь не имеет информации, позволяющей ему ответить на вопрос, он может ввести ответ НЕ ЗНАЮ. Тогда будут применяться другие правила вывода искомого значения, если такие правила есть в модели.

Вместо ответа на вопрос пользователь может ввести команду ЗАЧЕМ. В этом случае консультация будет прервана и пользователю будет сообщено, для вывода какой цели задается вопрос. Если пользователь снова введет команду ЗАЧЕМ, то он получит информацию о том, подцелью какого правила является эта цель, и т.д. Таким образом, многократный ввод

команды ЗАЧЕМ позволяет просматривать цели и правила на пути в сети вывода от исходной цели к текущей, т.е. к той, о которой задается вопрос.

Пользователь имеет возможность вместо ответа на вопрос откладывать решение задачи и переходить к работе с моделью в целом. По окончании этой работы он имеет возможность либо вернуться к прерванной консультации, либо завершить ее.

При выдаче информации о целях пользователю сообщаются:

- имена целей;
- типы (числовой атрибут или утверждение);
- развернутые имена;
- значения и их источники - для выведенных целей (в случае утверждений выдаются коэффициенты определенности);
- статус цели (значение неизвестно, цель активна, значение не удалось получить).

Пользователь может изменить значение любой цели. При внесении изменений он должен учитывать возможность нарушения корректности текущего решения, поскольку автоматически не пересматриваются те значения, которые были выведены исходя из измененных целей.

Просмотр источников значений целей обеспечивает объяснение того, КАК был получен тот или иной результат.

При выдаче информации о правилах пользователю сообщаются:

- имена правил;
- комментарии к правилам; цели и подцели правил.

Если пользователь хочет повторить консультацию с другими исходными данными, он может сбросить все значения по команде СБРОС.

Программа обеспечивает сброс либо всех значений, либо только тех, которые не были получены в ответ на вопрос к пользователю, т.е. выведенных значений. С помощью сброса выведенных значений, изменения некоторых исходных данных и повторного решения задачи пользователь может проверять гипотезы типа "что если".

Решение задач может, по желанию пользователя, проводиться с трассировкой консультации. При этом на экране появляется дополнительное окно для трассы, в которое помещаются сообщения о выполняемых действиях сценария, анализируемых целях и рассматриваемых правилах. Трассировка может использоваться как для отладки Базы знаний, так и для получения объяснений о работе экспертной системы.

При решении задач с протоколированием все сообщения, выдаваемые в ходе консультации, и все ответы, вводимые пользователем в ответ на вопросы, будут помещены в текстовой файл, называемый протоколом консультации. Если решение задачи проводится с трассировкой, то в протокол будет помещена и трасса консультации.

Контрольные точки используются для отладки моделей и создаются по команде пользователя во время консультации. При создании точки информация о состоянии всех моделей, загруженных к моменту выдачи команды, запоминается во внешнем файле, имя которого запрашивается у пользователя. В отличие от файлов, создаваемых действием СОЗДАТЬ ТОЧКУ из сценариев моделей, данный файл является двоичным и не должен корректироваться. В дальнейшем пользователь может загрузить контрольную точку из данного файла и продолжить консультацию. Загрузка может быть успешно осуществлена, если с момента создания точки модели не корректировались и не тестировались.

1.10. Вспомогательные средства комплекса

Программное средство ELK предназначено для подготовки гипертекстов. Оно позволяет значительно облегчить работу с большими объемами текстовой информации, что достигается путем предоставления пользователю простых и удобных средств структурирования текстов, возможности связывания полученных структурных единиц в конструкцию, отражающую семантику текста и поисковых процедур, эффективно реализующих эти связи.

В комплексе программ ЭКО средство ELK имеет следующие применения:

- при помощи ELK создана контекстно-зависимая Помощь, содержащая всю справочную информацию, имеющуюся в документации на комплекс, и доступная пользователю в любой момент в обоих режимах работы ОПС;
- при помощи ELK пользователь может разрабатывать собственные справочные средства для оказания помощи в процессе работы с экспертными системами;
- совместно используя средства ELK и XHT, пользователь может получать отчеты по Базе знаний в виде гипертекстов.

Программное средство XHT представляет собой генератор отчетов по Базе знаний. Оно предназначено для того, чтобы помочь пользователю при изучении, анализе и отладке Базы знаний, созданной с помощью комплекса ЭКО. При помощи XHT могут быть получены отчеты трех типов: текстовые, гипертекстовые и графические.

Текстовый отчет представляет собой твердую копию некоторой модели в Базе знаний и включает распечатку атрибутов, правил и сценария этой модели, а также таблицы перекрестных ссылок. Пользователю предоставляются широкие возможности по управлению содержанием текстовых отчетов.

Гипертекстовый отчет включает ту же информацию, что и текстовый, однако отличается от последнего способом использования. Если работа с текстовым отчетом может производиться без помощи ПЭВМ, то гипертекстовый отчет, как и любой другой гипертекст, существенно использует ПЭВМ для реализации поисковых процедур, поддерживающих связи между его фрагментами.

Графический отчет представляет собой графическое изображение сети вывода некоторой модели в Базе знаний в виде И-ИЛИ графа. Пользователь

может распечатать графический отчет или работать с ним непосредственно на ПЭВМ, используя возможность быстрого перехода к гипертекстовому отчету по соответствующей модели.

Для получения гипертекстовых отчетов необходимо совместно использовать средства XHT и ELK.

В состав комплекса программ ЭКО входят два стандартных интерфейса: графический и информационный.

Средства графического интерфейса предназначены для организации диалога с пользователем с помощью графических изображений (рисунков, схем, карт, диаграмм и т.д.).

Преимущества представления информации в виде рисунка по сравнению с текстовым представлением той же информации в некоторых случаях весьма существенны. Например: указать местоположение на плане.

Комплекс программ ЭКО допускает использование рисунков как в качестве иллюстраций, так и для задания вопросов.

Экспертная система, созданная при помощи комплекса ЭКО, может использоваться как подзадача в системах обработки данных, использующих файловую структуру DOS для хранения данных. Программная и информационная совместимость системы обработки данных и экспертной системы осуществляется через так называемый информационный интерфейс.

Свои функции информационный интерфейс осуществляет при помощи следующих операций:

- получение данных от систем обработки данных;
- интерпретация полученных данных в терминах переменных экспертной системы.