ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ РОССИЙСКОЙ ФЕДЕРАЦИИ

МОСКОВСКИЙ ИНЖЕНЕРНО-ФИЗИЧЕСКИЙ ИНСТИТУТ (ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ)

Г.В. Рыбина, Д.В. Демидов, М.Г. Иващенко

ЛАБОРАТОРНЫЙ ПРАКТИКУМ ПО КУРСУ «ПРОЕКТИРОВАНИЕ СИСТЕМ, ОСНОВАННЫХ НА ЗНАНИЯХ»

Учебное пособие

Москва 2007

УДК 004.8(076.5)
ББК 32.813
Р93

Рыбина Г.В., Демидов Д.В., Иващенко М.Г. Лабораторный практикум по курсу «Проектирование систем, основанных на знаниях»: *Учебное пособие.* М.: МИФИ, 2007. – 72 с.

Учебное пособие, выпускаемое как лабораторный практикум, предназначено для практической поддержки лабораторных работ по курсу «Проектирование систем, основанных на знаниях», читаемому в Московском инженерно-физическом институте (государственном университете) на кафедре №22 (специальность «Прикладная математика и информатика»).

Пособие представляет собой описание 4-х лабораторных работ, связанных с построением простейших прототипов интегрированных экспертных систем (ИЭС) и их компонентов на основе использования оригинального инструментария – комплекса АТ-ТЕХНОЛОГИЯ третьего поколения, разработанного в лаборатории «Системы искусственного интеллекта» кафедры №22 МИФИ под руководством профессора Рыбиной Г.В. Каждая лабораторная работа содержит теоретический материал, включающий описание базовых функциональных возможностей комплекса АТ-ТЕХНОЛОГИЯ и технологии построения ИЭС, и практическую часть, состоящую из наборов заданий и методики выполнения работ.

Предназначено в качестве основной литературы для студентов групп К9-221, К9-222, К9-223, К9-224, К8-281, К8-282, а также может быть рекомендовано студентам, аспирантам и специалистам, занимающимся проектированием и разработкой широкого класса экспертных систем.

Рецензент кандидат технических наук, доцент
зав. кафедрой №28 Румянцев В.П.

Рекомендовано редсоветом МИФИ
в качестве учебного пособия

 ISBN 978-5-7262-0789-6

Редактор Т.В. Волвенкова

Подписано в печать Формат 60х 84 1/16
Печ. л. Уч. изд. л. 8.75 Тираж 200 экз.
Изд. № Заказ №

Московский инженерно-физический институт (государственный университет).
Типография МИФИ.
115409, Москва, Каширское ш., 31

Оглавление

Введение

Целью практической поддержки курса «Проектирование систем, основанных на знаниях» является обучение студентов высших учебных заведений навыкам создания прототипов интегрированных экспертных систем (ИЭС) с помощью специализированной программной среды на примере инструментального комплекса АТ-ТЕХНОЛОГИЯ, разработанного в лаборатории «Системы искусственного интеллекта» на кафедре Кибернетики МИФИ под руководством Лауреата премии Президента РФ в области образования профессора Г.В.Рыбиной.

Комплекс АТ-ТЕХНОЛОГИЯ представляет собой набор инструментальных средств для создания прототипов ИЭС в статических проблемных областях в соответствии с задачно-ориентированной методологией. В настоящее время разработано несколько конфигураций комплекса – минимальная, базовая, полная и веб-ориентированная, отличающихся набором программных средств, поддерживающих все или некоторые этапы жизненного цикла построения ИЭС.

Так, минимальная конфигурация, по сути, представляет собой оболочку ИЭС, в которую включен набор средств инженера по знаниям. Базовая конфигурация представляет собой полноценную среду разработки ИЭС на всех этапах жизненного цикла, включающую в себя в том числе средства автоматизированного приобретения знаний, средства построения обучающих ИЭС и др. программные средства, функционирующие под управлением интеллектуального планировщика. В полную версию дополнительно входят программные средства проектирования хранилищ данных, репозиторий проектов и узкоспециализированные компоненты. В рамках курса «Проектирование систем, основанных на знаниях» предусматривается изучение минимальной и базовой конфигураций комплекса АТ-ТЕХНОЛОГИЯ, что является достаточным для выполнения курсового проекта.

Данное пособие, выпускаемое как лабораторный практикум, является логическим продолжением изданного в 2001 году учебного пособия «Инструментальный комплекс АТ-ТЕХНОЛОГИЯ для поддержки разработки интегрированных экспертных систем» (авторы Рыбина Г.В., Пышагин С.В., Смирнов В.В., Левин Д.Е., Душкин Р.В.) [13]. В этом пособии особое вни-

мание уделялось изучению методов и средств автоматизированного приобретения знаний для ИЭС на этапах анализа и проектирования. В новом пособии акцент делается на технологические аспекты разработки ИЭС и этапы реализации и тестирования.

Пособие структурировано таким образом, что описание каждой лабораторной работы содержит теоретический материал, посвященный описанию отдельных функциональных возможностей комплекса АТ-ТЕХНОЛОГИЯ и технологии построения ИЭС, и практическую часть, включающую наборы заданий и методику их выполнения. В пособие включены также варианты домашних заданий для самостоятельной работы и подготовки к зачету, достаточно объёмный список рекомендованной литературы и Приложения, содержащие описания отдельных теоретических вопросов.

В соответствии с учебным планом предусмотрено проведение 4-х лабораторных работ на платформе IBM PC под управлением операционных систем семейства MS Windows 9x, 2000, XP. В ходе обучения студенты знакомятся с базовым инструментарием инженера по знаниям, входящим в состав комплекса АТ-ТЕХНОЛОГИЯ, получают представление о методах обработки знаний универсальным решателем, создают простейший прототип ИЭС, получают практический опыт интеграции ядра прототипа с разного рода информационными компонентами, такими как база данных и MS Excel.

Пособие предназначено в качестве основной литературы для студентов групп К9-221, К9-222, К9-223, К9-224, изучающих базовый курс «Проектирование систем, основанных на знаниях», читаемый на кафедре №22 МИФИ (9 семестр), так и для студентов кафедры №28 групп К8-281, К8-282, изучающих курс «Экспертные системы» (8 семестр), а также может быть рекомендовано студентам, аспирантам и специалистам, занимающимся проектированием и разработкой широкого класса экспертных систем.

Лабораторная работа №1. Освоение инструментария инженера по знаниям

Целью лабораторной работы является получение навыков применения инструментария инженера по знаниям комплекса АТ-ТЕХНОЛОГИЯ на примере прототипа интегрированной экспертной системы медицинской диагностики.

Работа считается выполненной, если была продемонстрирована работа прототипа с базой знаний, расширенной хотя бы одним целевым параметром, одним симптомом и одним правилом.

Предполагается, что студент, приступающий к лабораторной работе, имеет представление об общей архитектуре экспертной системы, назначении и принципах работы основных компонентов в составе экспертной системы, продукционной модели представления знаний, основных схемах вывода на продукциях.

1.1. План работы

1. Знакомство с инструментальным комплексом АТ-ТЕХНОЛОГИЯ, режим отладки прототипа.
2. Знакомство с прототипом интегрированной экспертной системы диагностирующего типа:
   - Сеанс консультации, режимы работы универсального решателя (прямой вывод, обратный вывод, смешанный вывод, уточняющие поддиалоги), формирование отчета о сеансе консультации (три представления).
   - Подсистема объяснения.
3. Знакомство с режимом инженера по знаниям:
   - Редактор базы знаний.
   - Редактор сценариев диалога.
   - Средство построения подсистемы объяснения.
4. Получение навыков работы с редактором базы знаний: расширение базы знаний демонстрационного прототипа.
5. Получение навыков работы с редактором сценариев диалога: подготовка и модификация сценариев диалога в соответствии с изменениями базы знаний.
6. Получение навыков работы со средством построения подсистемы объяснения: генерация и редактирование записанных объяснений.

1.2. Архитектура типового прототипа интегрированной экспертной системы, созданного средствами комплекса АТ-ТЕХНОЛОГИЯ

В рамках инструментального комплекса АТ-ТЕХНОЛОГИЯ прикладная интегрированная экспертная система строится в соответствии с принципами компонентной архитектуры, причем каждый компонент является COM-объектом (объектом автоматизации).

Отдельные компоненты общаются между собой посредством обмена текстовыми сообщениями. В качестве языка общения выбрано подмножество языка XML. Помимо механизма обмена сообщениями, для обмена информацией между компонентами ИЭС используется компонент "Классная доска".

Конфигурация той или иной прикладной ИЭС однозначно определяется набором компонентов, входящих в ее состав, конфигурационной информацией для каждого компонента, а также стартовым сообщением, с отправки которого начинается работа системы. Вся эта информация описывается на специальном языке (подмножестве XML) и называется конфигурацией ИЭС.

Таким образом, построение той или иной прикладной ИЭС средствами комплекса АТ-ТЕХНОЛОГИЯ сводится к созданию ее конфигурации без компиляции исходных модулей. Расширение функциональности ИЭС возможно посредством использования интерпретируемых языков типа Visual Basic или разработке совместимых компонентов, поддерживающих интерфейс обмена сообщениями.

Типовой прототип интегрированной экспертной системы, разрабатываемый средствами комплекса АТ-ТЕХНОЛОГИЯ, включает следующие компоненты:

1. Ядро интегрированной экспертной системы
   - Универсальный решатель, предоставляющий разнообразные средства вывода (динамическая библиотека SolverX.dll)
   - Компонент “Классная доска”, играющая роль рабочей памяти и обеспечивающая размещение, хранение и доступ к данным, предоставляя основным компонентам ИЭС единое адресное пространство для обмена информацией (динамическая библиотека Board.dll)

   - База знаний на языке представления знаний комплекса АТ-ТЕХНОЛОГИЯ (файл .kbs), либо в формате внутреннего представления базы знаний (файле .xml).
   - Подсистема общения
   - Диалоговый компонент (динамическая библиотека DialogerLib.dll)
   - Сценарий диалога на языке описания сценариев диалога комплекса АТ-ТЕХНОЛОГИЯ (файл .dsf).
2. Подсистема объяснения
   - Объяснительный компонент (файл ExplLib.dll)
   - Банк записанных объяснений (файл exp.xml)
3. Подсистема интеграции с базами данных и прикладными программами
   - Компонент интерпретации скриптов (динамические библиотеки ScriptLib.dll и msscript.ocx)
   - Сценарии взаимодействия ядра ИЭС с базами данных и внешними программными средствами (скриптовые файлы *.vbs или *.js)
4. Служебные интерфейсные и конфигурационные компоненты, обеспечивающие среду функционирования основных компонентов ИЭС
   - Типовой исполняемый модуль (файл es.exe)
   - Конфигурационный файл (файл config.xml)
   - Стартовый компонент, служащий для осуществления загрузки и интерпретации конфигурации ИЭС (динамическая библиотека StarterLib.dll)
   - Брокер объектов, осуществляющий регистрацию, хранение и взаимодействие основных компонентов ИЭС (динамическая библиотека BrokerLib.dll)
   - Графические файлы и файлы шаблонов отчетов
5. Средства инженера по знаниям
   - Специализированный редактор сценариев диалога (динамическая библиотека DSDLEditorLib.dll)
   - Интеллектуальный редактор базы знаний (динамическая библиотека KBTools.dll)
   - Средство верификации базы знаний (динамическая библиотека KBTools.dll)
   - Средство построения подсистемы объяснения (динамическая библиотека ExplLib.dll)

- Средство отладки универсального решателя (динамическая библиотека SolverX.dll)

Все перечисленные компоненты входят в минимальную конфигурацию инструментального комплекса АТ-ТЕХНОЛОГИЯ.

1.3. Методические указания к работе

1. Знакомство с инструментальным комплексом АТ-ТЕХНОЛОГИЯ, режим отладки прототипа

Инструментальный комплекс АТ-ТЕХНОЛОГИЯ предназначен для компьютерного построения интегрированных экспертных систем в статических проблемных областях и автоматизирует все этапы жизненного цикла разработки ИЭС. При переходе к этапу реализации автоматически создается скелет прототипа в минимальной конфигурации оболочки ИЭС, а затем на основе результатов проектирования в конфигурацию прототипа включаются остальные программные и информационные компоненты: база знаний, сценарий диалога, подсистема объяснений, средства интеграции.

В целях наглядности знакомство с комплексом начнется с уже готового демонстрационного прототипа ИЭС “Diagnosis”. Далее запустите инструментальный комплекс АТ-ТЕХНОЛОГИЯ и загрузите проект “Diagnosis” (Рис. 1).

При загрузке проекта интеллектуальный планировщик инструментального комплекса АТ-ТЕХНОЛОГИЯ активирует плановые задачи стадии анализа системных требований к системе. Для запуска прототипа необходимо перейти на стадию тестирования. Для этого в списке текущих плановых задач выберите сначала “Завершить этап анализа системных требований”, затем “Завершить этап проектирования”, затем “Завершить этап реализации” и, наконец, “Запустить прототип ИЭС в тестовом режиме” (Рис. 2).

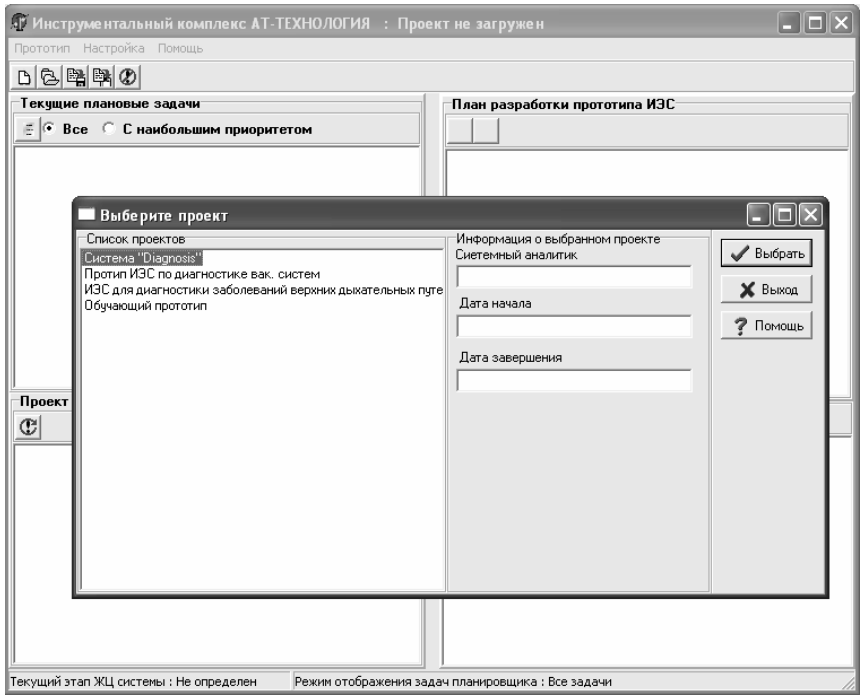

Рис. 1 Загрузка проекта в инструментальном комплексе АТ-ТЕХНОЛОГИЯ

После запуска прототипа появится основное окно интегрированной экспертной системы “Diagnosis” (Рис. 3).

2. Знакомство с прототипом интегрированной экспертной системы диагностирующего типа

Для ознакомления с интегрированной экспертной системой (ИЭС) “Diagnosis” следует самостоятельно пройти сеанс консультации. Для этого используйте пункты главного меню “Сеанс консультации\Начать сеанс…”.

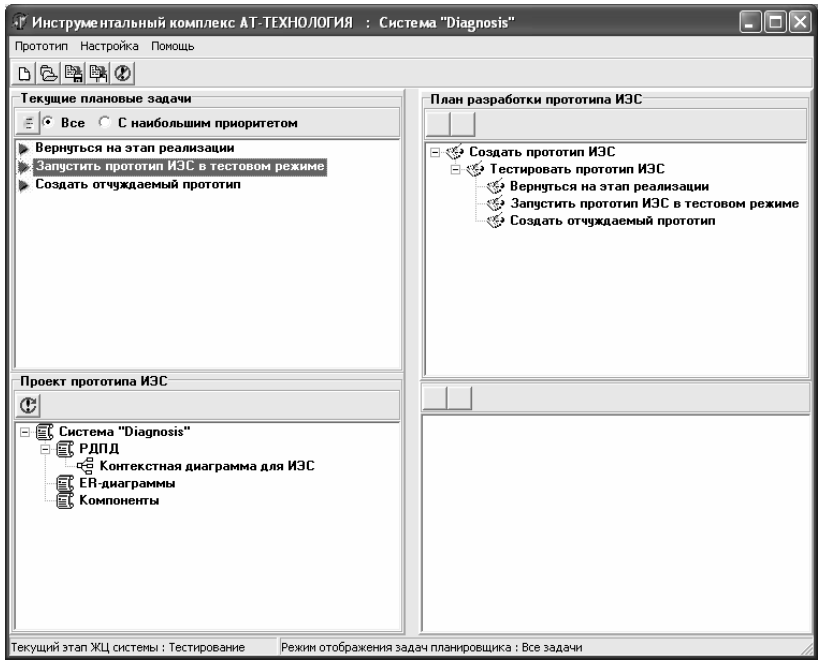

Рис. 2 Запуск прототипа в режиме отладки

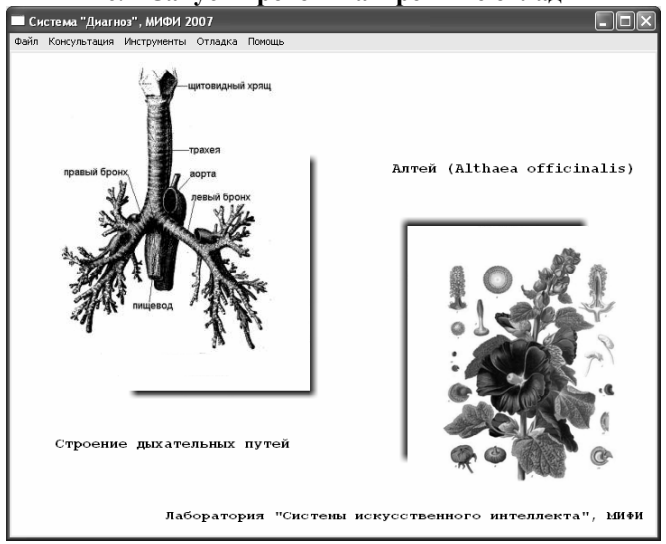

Рис. 3 Главное окно демонстрационного прототипа интегрированной экспертной системы “Diagnosis”

Предусмотрено три сеанса консультации для демонстрации режимов работы универсального решателя. В первом сценарии применяется прямой вывод на основе данных, собранных в ходе сеанса. На Рис. 4 представлены диалоговые окна, генерируемые подсистемой общения для сбора входных данных:

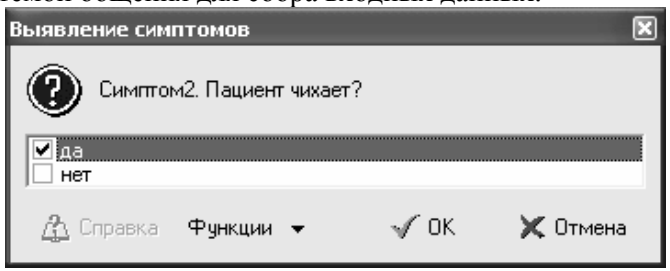

Рис. 4 Пример экрана сбора данных в ходе сеанса консультации с ИЭС “Diagnosis”

Результаты сеанса представляются в простом окне информационного типа (Рис. 5), генерируемого подсистемой общения в соответствии со сценарием.

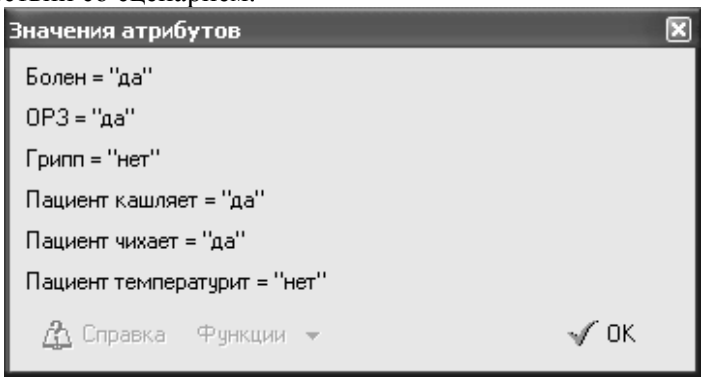

Рис. 5 Результаты сеанса консультации

Во втором сценарии применяется обратный вывод на основе неполных входных данных, а универсальный решатель инициирует уточняющие поддиалоги. Если пользователя интересует вопрос “Почему задается уточняющий вопрос?”, то с помощью меню “Функции” на диалоговой форме пользователь может активировать подсистему объяснения (Рис. 6).

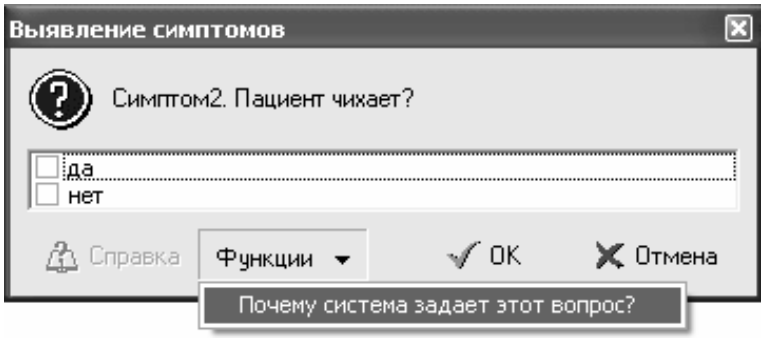

Рис. 6 Задание вопроса “Почему”

На Рис. 7 показан пример ответа на вопрос “Почему”, сгенерированный подсистемой объяснения динамически, используя информацию из базы знаний. После закрытия окна сеанс консультации будет продолжен.

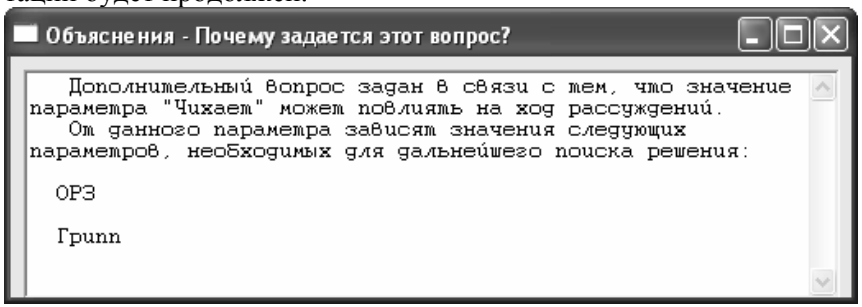

Рис. 7 Пример ответа на вопрос “Почему”, сгенерированный подсистемой объяснения

Результаты сеанса представляются в виде отчета (Рис. 8), генерируемого компонентом составления отчета Reporter в составе подсистемы общения.

В третьем сценарии применяется смешанный вывод в условиях отсутствия входных данных, где универсальный решатель инициирует уточняющие поддиалоги. Результаты представляются в Excel (Рис. 9), генерируемого компонентом интеграции Scripter, вызываемым подсистемой общения.

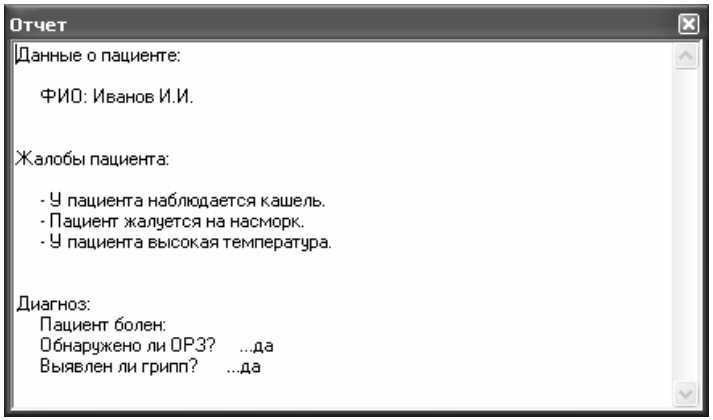

Рис. 8 Экран формы отчета о сеансе консультации

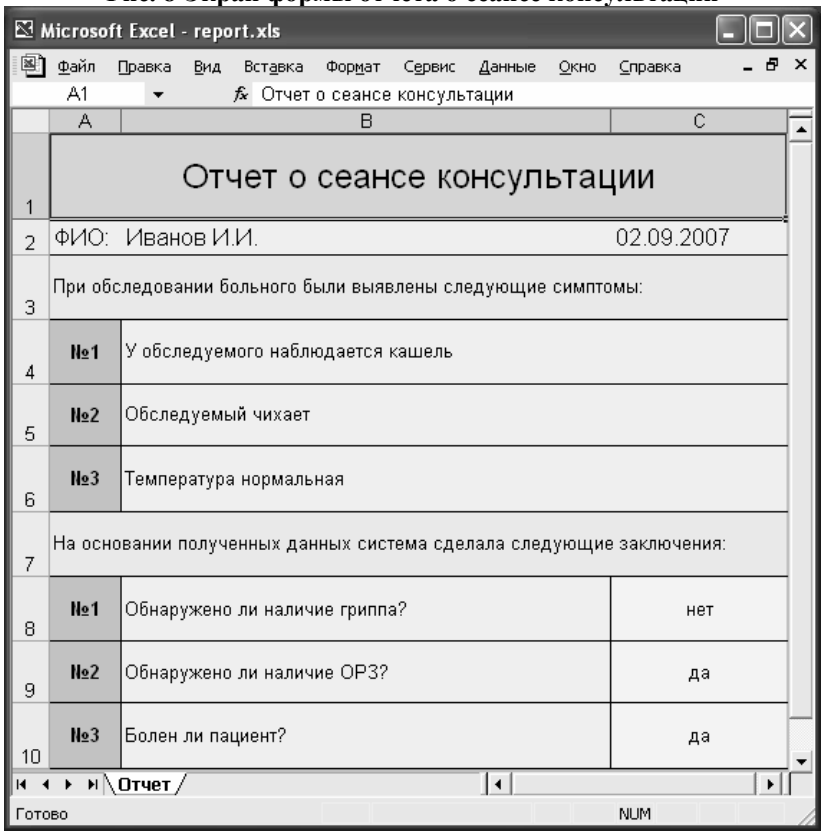

Рис. 9 Вывод отчета о сеансе консультации в Excel

Для просмотра объяснений полученных рекомендаций используйте пункт главного меню “Сеанс консультации\Объяснения”. После запуска подсистемы объяснения пользователю доступны два эквивалентных представления объяснений: гипертекстовое и когнитивно-графическое.

Экран гипертекстового представления делится на две части (Рис. 10): слева приведены объяснения заключений системы в последовательности их получения. Справа в гипертекстовой форме приведены утверждения, истинность которых была доказана в ходе рассуждений. Если пользователя интересует вопрос “Как был получен данный факт?”, то с помощью гипертекстовой ссылки можно просмотреть ответ – правило, подтверждающее данный факт.

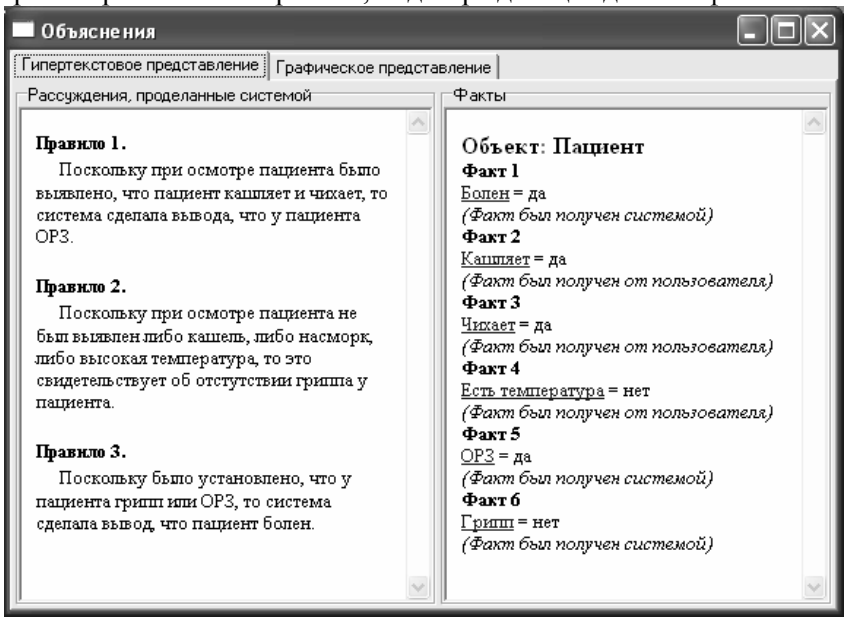

Рис. 10 Гипертекстовое представление объяснений

Экран когнитивно-графического представления также делится на две части (Рис. 11): сверху отображается дерево решений, полученное в ходе рассуждений системы, где вершинами являются утверждения, а дугами – зависимости, отраженные в правилах базы знаний. Вершины и дуги интерактивны, и при их щелчке мыши на них внизу экрана отображается поясняющий текст (правило или утверждение).

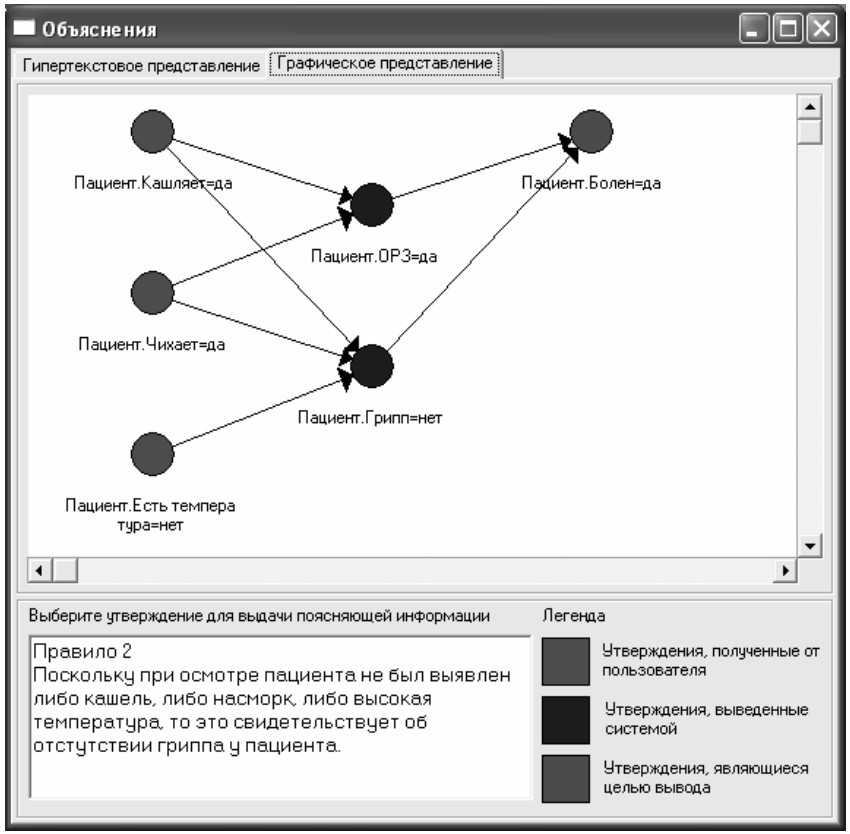

Рис. 11 Когнитивно-графическое представление объяснений

3. Знакомство с режимом инженера по знаниям

В режиме инженера по знаниям доступны следующие инструменты:

- Интеллектуальный редактор базы знаний
- Специализированный редактор сценариев диалога
- Средство построения подсистем объяснения
- Редактор конфигурации системы.

Далее подробно рассматриваются перечисленные средства.

Интеллектуальный редактор базы знаний

Для запуска редактора базы знаний воспользуйтесь пунктом главного меню “Инструменты\Интеллектуальный редактор базы знаний…”.

Для редактирования доступны правила базы знаний, объекты проблемной области, их атрибуты, типы атрибутов, значения типов. С помощью окон навигации можно просмотреть список объектов. При выборе объекта автоматически показываются его атрибуты и их типы (Рис. 12). При выборе атрибута или типа автоматически отображается список его значений. Для навигации по списку правил следует использовать выпадающий список с названиями правил.

В один и тот же момент редактор позволяет редактировать либо правила, либо объекты, атрибуты и типы. Для переключения между режимами следует выбрать мышкой окно навигации, после чего становятся доступны соответствующие пункты меню.

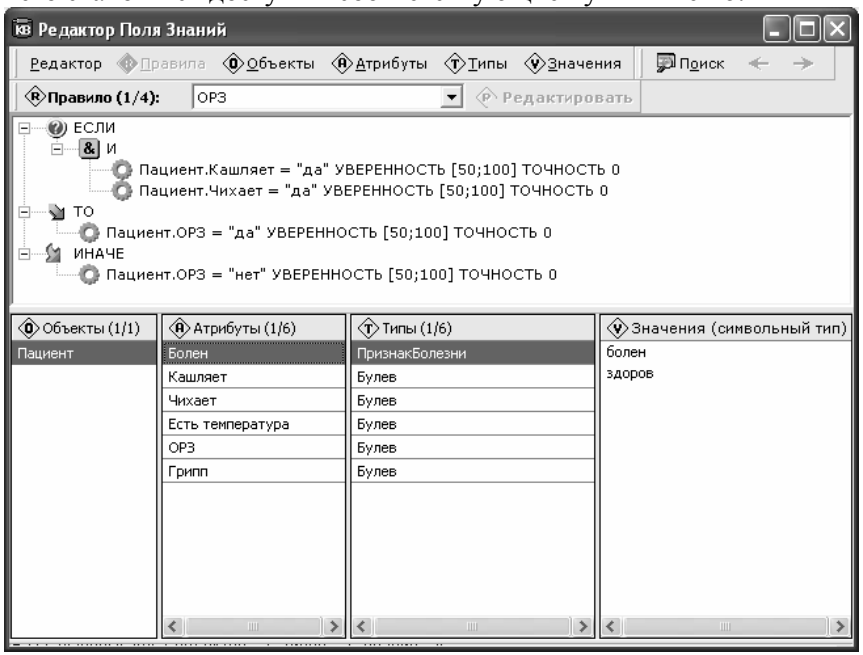

Рис. 12 Пример экрана редактора базы знаний

Чтобы изменить правило, необходимо выбрать его из списка и нажать кнопку “Редактировать”. После этого можно редактировать утверждения и логические операции по двойному щелчку мыши.

В посылке правила допускаются логические операции конъюнкции, дизъюнкции и отрицания. Для добавления оператора

необходимо выбрать место добавления (посылка правила или вложенный оператор) и в контекстном меню выбрать “Добавить оператор”. Далее следует указать нужный оператор (Рис. 13) и нажать “OK”.

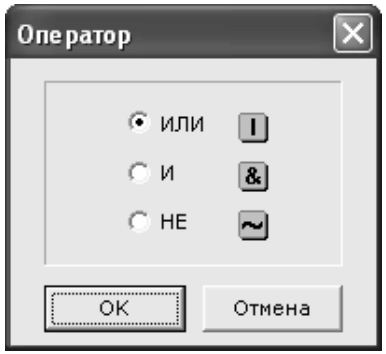

Рис. 13 Окно добавления оператора

Для добавления утверждения необходимо выбрать место добавления (посылка правила или оператор) и в контекстном меню выбрать “Добавить выражение”.

Утверждения представляют собой сопоставление атрибута со значением. При этом значение выбирается из списка значений типа атрибута, который должен быть определен заранее (Рис. 14). Коэффициенты уверенности и точности принимают значения от 0 до 100.

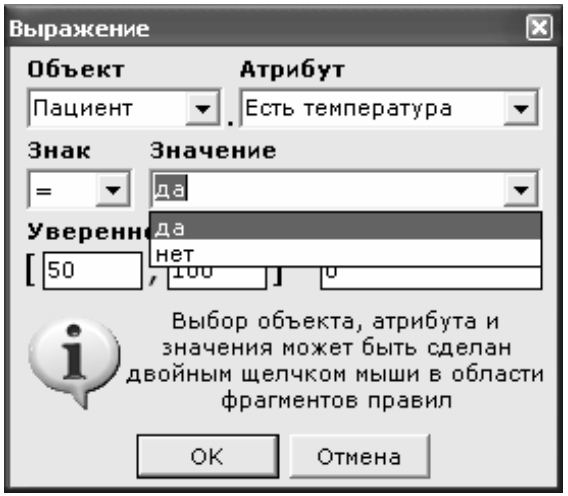

Рис. 14 Окно добавления утверждения

Для выхода из режима редактирования правила необходимо нажать кнопку “OK” или “Cancel”.

Специализированный редактор сценариев диалога

Для запуска редактора сценариев диалога воспользуйтесь пунктом меню “Инструменты\Редактор сценариев диалога…”. Рабочее окно редактора представлено на Рис. 15.

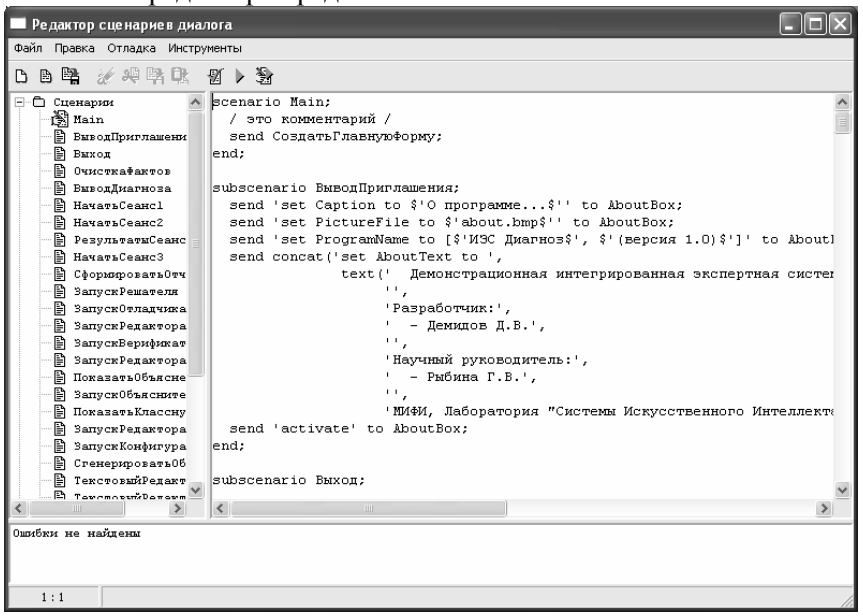

Рис. 15 Главное окно редактора сценария диалога

Слева отображается структура сценария: главный сценарий и вспомогательные сценарии, а также сообщения, отсылаемые подсистеме общения для генерации диалоговых окон. Справа расположена область редактирования сценариев и сообщений.

Студенту предлагается самостоятельно найти и проанализировать типовые сценарии “Main”, “НачатьСеанс”, “РезультатыСеанса”, “ОчисткаФактов”, “ ОбъяснитьПочему” и сообщения “СоздатьГлавнуюФорму”, “УзнатьАтрибут”.

Редактор сценариев диалога обладает следующими сервисными функциями: проверка синтаксической корректности текста (вызов через меню “Отладка\Проверить на наличие ошибок…”), запуск сценария или сообщения в тестовом режиме (вызов через

меню “Отладка\Выполнить текущий элемент…” и “Отладка\Выполнить главный сценарий…”).

Средство построения подсистем объяснения

Средство построения подсистемы объяснения в инструментальном комплексе АТ-ТЕХНОЛОГИЯ используется на этапе проектирования и представляет собой специализированный редактор записанных объяснений с их предварительной автоматической генерацией (Рис. 16).

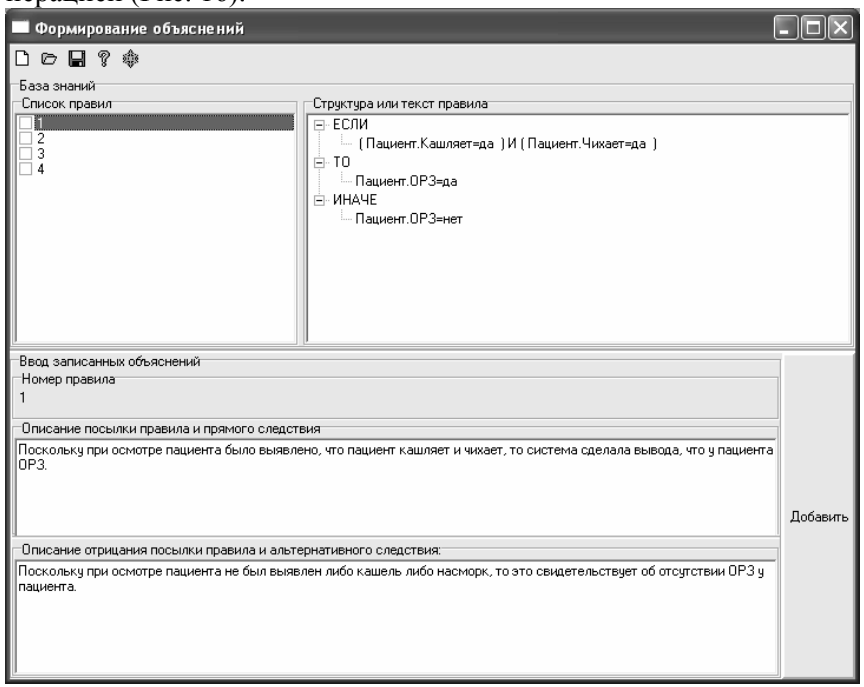

Рис. 16 Главное окно средства построения подсистемы объяснения

В режиме инженера по знаниям для запуска генератора записанных объяснений следует воспользоваться меню “Инструменты\Сгенерировать записанные объяснения…”. После генерации будет создан файл genexp.xml, который будет автоматически открыт на редактирование.

Пример сгенерированного текста для правила выглядит следующим образом:

```
Поскольку Кашляет=да Чихает=да Есть температура=да , то
Грипп=да
```

Данный текст следует вручную исправить, например, следующим образом:

```
Поскольку при осмотре пациента было выявлено, что
пациент кашляет и чихает, а также наблюдается
повышенная температура, то система сделала вывод, что у
пациента грипп.
```

После завершения редактирования необходимо сохранить файл под именем exp.xml и в дальнейшем использовать меню “Инструменты\Редактор объяснений…”.

Редактор конфигурации системы

Для конфигурирования системы следует воспользоваться меню “Инструменты\Конфигурация прототипа”, после файл конфигурации config.xml будет открыт на редактирование (Рис. 17).

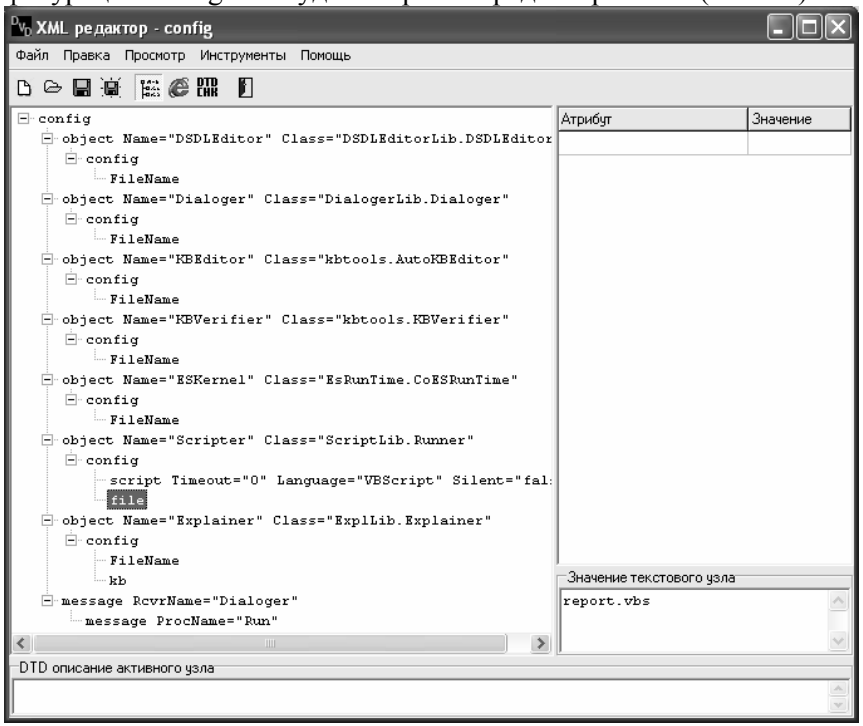

Рис. 17 Главное окно редактора конфигурации

В узел config находятся объявления компонентов, входящих в состав прототипа, а в узле message указывается компонент,

которому передается управление при запуске системы (обычно это диалоговый компонент).

4. Получение навыков работы с редактором базы знаний

Для получения навыков работы с редактором базы знаний студенту предлагается самостоятельно расширить базу знаний демонстрационного прототипа “Diagnosis” новым заключением, новым симптомом и новым правилом. Для этого следует:

1) Открыть редактор базы знаний.
2) Выбрать тип “Заключение” и добавить еще одно значение к данному типу с помощью меню “Типы\Добавить” (Рис. 18).

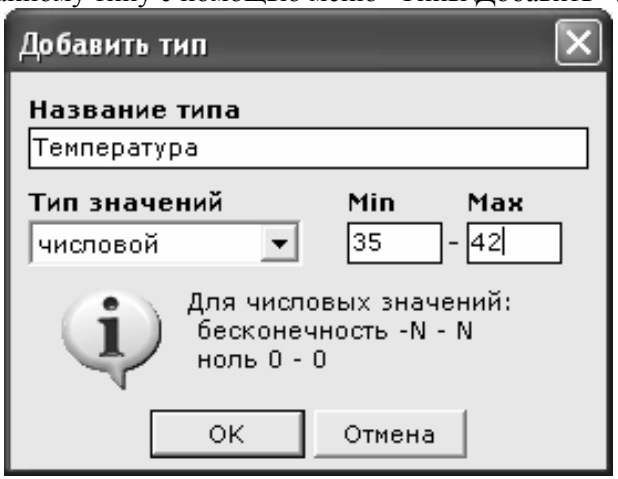

Рис. 18 Пример добавления типа в базу знаний

3) Добавить в базу знаний новый тип для придуманного заранее симптома, указать значения типа.
4) Выбрать объект и добавить новый симптом, указав для него добавленный ранее тип.
5) Перейти в режим редактирования правил и добавить новое правило, обязательно указав его название, например, “Простуда”.
6) Сформировать логическую структуру посылки, используя опцию контекстного меню “Добавить оператор” для узла “Если”.
7) Добавить утверждения в посылку правила, используя опцию контекстного меню “Добавить выражение”.
8) Добавить утверждение о заключении в следствие правила, выбрав узел “То” (“Иначе”).

9) Сохранить правило, нажав на кнопку “ОК”.
10) Дополнить правила “Болен” и “Здоров” для учета нового заключения.
11) Сохранить БЗ и перезапустить прототип.

Для просмотра базы знаний на языке представления знаний можно воспользоваться меню “Инструменты\База знаний на ЯПЗ”, либо открыв файл diagnosis.kbs на редактирование в Notepad.

5. Получение навыков работы с редактором сценариев диалога

Для того, чтобы система смогла корректно учесть изменения в базе знаний необходимо модифицировать сценарий диалога. Далее студенту предлагается самостоятельно добавить в сценарий диалога сообщение для ввода значения нового симптома, модифицировать сценарии очистки начальных данных, ввода данных, вывода отчета о сеансе консультации. Для этого следует:

1) Открыть редактор сценариев диалога.
2) Добавить новое сообщение, аналогично сообщениям для ввода значений других симптомов.
3) Найти сценарий “ОчисткаФактов” и добавить строку для очищения значения нового симптома.
4) В сценарий по вводу данных “НачатьСеанс” добавить отсылку нового сообщения.
5) Модифицировать сценарии по выдаче рекомендаций “ВыводДиагноза”, “РезультатыСеанса2” с учетом нового диагноза.
6) Добавить вызов сценария обработки вопроса “Почему” для нового симптома. Для этого в тексте сообщения “УзнатьАтрибут” необходимо указать соответствующий сценарий “ОбъяснитьПочему”.

```
message УзнатьАтрибут1_5 to Asker about
#ОБЪЕКТ1.АТРИБУТ5#;
  line ‘on $’Почему система задает этот вопрос?$’
execute ОбъяснитьПочему1_5’;
 …
end;
```

7) В сценарии описать вызов подсистемы объяснения. Подсистема активируется путем отсылки XML-сообщения “Why”, где указывается номер соответствующего объекта базы знаний и номер атрибута.


```
subscenario ОбъяснитьПочему1_5;
  send ‘<message ProcName="Why">
          <obj>1</obj>
          <attr>5</attr>
        </message>’ to Explainer;
end;
```

8) Проверить синтаксическую корректность сценария диалога с помощью опции “Проверить на наличие ошибок” и исправить возможные неточности.
9) Сохранить сценарий и перезапустить прототип.
10) Проверить, что в ходе сеанса консультации запрашивается значение нового атрибута.

6. Получение навыков работы со средством построения подсистемы объяснения

Далее студенту предлагается самостоятельно доработать подсистему объяснения для учета нового правила. Для этого следует:

1) Отредактировать записанные объяснения для измененных правил “Болен”, “Здоров” с помощью XML-редактора.
2) Добавить записанные объяснения для нового правила.
3) Перезапустить прототип.
4) Проверить, что система выдает соответствующие объяснения в случае применения нового правила и при задании вопроса “Почему” во время сеанса консультации.

После завершения отладки демонстрационного прототипа необходимо показать результаты работы преподавателю.

1.4. Домашнее задание

С помощью программных средств комплекса АТ-ТЕХНОЛОГИЯ можно строить прототипы ИЭС, решающие задачи медицинской и технической диагностики, проектирования, прогнозирования и др. К следующей лабораторной работе студенту необходимо выбрать проблемную область и сформулировать конкретную задачу, подходящую для решения ИЭС.

Далее, следует выявить несколько закономерностей в проблемной области в виде продукционных правил, образующих дерево решений глубиной 2-3 правила и шириной 3-4 правила. Таким образом, для следующей лабораторной работы студент должен

подготовить фрагмент поля знаний, содержащий порядка 10 правил типа Если-То, причем посылки правил должны содержать логические связки И, ИЛИ, НЕ, а действие правил должно представлять собой одно или несколько утверждений.

Лабораторная работа №2. Разработка прототипа интегрированной экспертной системы

Целью работы является получение навыков создания прототипа интегрированной экспертной системы с помощью инструментального комплекса АТ-ТЕХНОЛОГИЯ.

Работа считается выполненной, если продемонстрирована работа созданного прототипа, прокомментированы правила базы знаний и сценарий диалога.

Предполагается, что студент, приступающий к лабораторной работе, владеет навыками работы с редакторами базы знаний и сценариев диалога.

2.1. План работы

1. Анализ системных требований к прототипу ИЭС для придуманной проблемной области. Построение архитектуры прототипа. Приобретение знаний.
2. Проектирование прототипа: формирование перечня программных компонентов в составе прототипа ИЭС, включение необходимых программных компонентов в состав прототипа и конфигурирование каждого компонента.
3. Реализация: разработка сценария диалога и подсистемы объяснения в режиме инженера по знаниям. Настройка средств вывода и других компонентов. Разработка сценария взаимодействия со средством формирования отчетности.
4. Отладка работы прототипа для различных стратегий вывода в режиме консультации, проверка корректности рекомендаций системы.

2.2. Представление и обработка знаний универсальным решателем комплекса АТ-ТЕХНОЛОГИЯ

Универсальный решатель реализован в виде COM-объекта и оформлен как повторно-используемый компонент в соответствие с требованиями, предъявляемыми к разработке компонентов для инструментального комплекса АТ-ТЕХНОЛОГИЯ.

Внутреннее представление знаний в рабочей памяти осно-

вывается на сращивании продукционной модели и объектно-ориентированного подхода. Сущности проблемной области отображаются в иерархию классов, а закономерности в инкапсулированные правила. Кроме того, для представления императивных знаний служат инкапсулированные процедуры и функции, причем поддерживается вызов методов внешних COM-объектов из правил и процедур.

Универсальный решатель поддерживает следующие стратегии рассуждений: прямой вывод, обратный вывод и смешанный вывод. Решатель реализует оригинальный метод ведения уточняющих поддиалогов и подтверждения гипотез.

Представление и обработка неопределенности

В модель представления знаний комплекса АТ-ТЕХНОЛОГИЯ для описания знаний с неопределенностью введены два коэффициента уверенности (уверенность и возможность), приписываемые утверждениям, входящим в посылки и следствия правил, отражающих закономерности в проблемной области. Коэффициенты могут принимать значения в интервале [0; 100], где 100 соответствует абсолютной достоверности.

Для обработки неопределенности, а именно для расчета результатов логических операций и неопределенности следствий правил в универсальном решателе используются методы Байеса и Демпстера-Шейфера. В том случае, когда значениям параметров приписывается один коэффициент уверенности, используется метод вывода по схеме Байеса, в противном случае для обработки интервала уверенности применяется метод Демпстера-Шейфера.

Кроме того, для оптимизации вычислений применяется бесконечнозначная логика, где значение истинности меняется в интервале [0; 1], расширенном специальным значением ND (не определено).

Представление и обработка неточности

Неточность проявляется в том случае, если значения числового параметра получается в результате измерений с некоторой погрешностью. Для представления неточной информации используется коэффициент точности, который в большинстве случаев можно трактовать как относительную погрешность. Коэффициент точности может принимать значения в интервале [0; 100], причем 0 означает абсолютно точное число, а 100 – стопроцентную относительную погрешность числа. Для обработки неточности в универ-

сальном решателе используется метод неточных сравнений и неточная арифметика.

Представление и обработка нечеткости

Нечеткость проявляется в рассуждениях эксперта в тех случаях, когда для описания количественных или качественных величин используются качественные оценки, например, “очень большой”, “медленно”, “довольно тепло”, “темно зеленый”. Для представления подобных оценок используется аппарат функций принадлежности. Для проведения рассуждений (корректного сопоставления фактического значения параметра с нечеткой оценкой) используется аппарат нечеткой логики и ряд вспомогательных методов (нечеткие сравнения и арифметика).

Универсальный решатель инструментального комплекса АТ-ТЕХНОЛОГИЯ помимо специфических методов обработки каждого НЕ-фактора также реализует методы совместной обработки нечеткости, неопределенности и неточности. В частности, ряд методов фаззификации, дефаззификации, в том числе оригинальный метод дефаззификации многомодальных функций принадлежности.

2.3. Методические указания к работе

Перед началом работы необходимо подготовить скелетный прототип будущей системы. Для этого создайте новую папку с названием системы и скопируйте в нее файлы оболочки из папки Shell.

1. Анализ системных требований

На данном этапе с помощью средств приобретения знаний студент должен сформировать базу знаний прототипа на языке представления знаний инструментального комплекса АТ-ТЕХНОЛОГИЯ, используя домашние заготовки правил. Для этого следует выполнить следующие действия.

1) Запустите прототип и перейдите в режиме инженера по знаниям, вызвав интеллектуальный редактор базы знаний.
2) Выделите объекты проблемной области и добавьте их с помощью меню “Объекты\Добавить”.
3) Представьте на бумаге каждое утверждение в посылке и следствии правил в виде `<объект>.<атрибут> <операция сравнения> <значение>` и выделите допус-

тимые значения для каждого атрибута. Классифицируйте атрибуты как символьные, числовые, нечеткие.

4) Для каждого атрибута определите тип в базе знаний с помощью меню “Типы\Добавить”, указывая осмысленное имя типа. Возможно, какие-то атрибуты будут иметь один и тот же тип. Если тип символьный, то выберите в списке добавленный тип и укажите его значения с помощью меню “Значения\Добавить”.
5) Для каждого объекта опишите атрибуты, указывая их тип. Для этого необходимо выбрать объект из списка и создать необходимые атрибуты с помощью меню “Атрибуты\Добавить”.
6) После формирования структуры предметной области сохраните базу знаний и перейдите в режим редактирования правил. Создайте правила, указывая их названия.
7) Сохраните базу знаний и перезапустите прототип.

2. Проектирование прототипа

На данном этапе студент самостоятельно должен сформировать перечень программных компонентов в составе прототипа ИЭС для придуманной заранее проблемной области, воспользовавшись примером демонстрационной системы Diagnosis.

Для включения необходимых программных компонентов в состав прототипа следует сконфигурировать систему. Для этого воспользуйтесь меню “Инструменты\Конфигурация прототипа…”, либо откройте файл config.xml на редактирование, переключившись в Windows. Конфигурационная информация записывается на языке XML в следующем формате:

```
Config = '<config>'
         {Object}*
         Message
         '</config>'

Object = '<object Name=' ObjName 'Class=' ClsName '>'
         '<config>'
         CfgText
         '</config>'
         '</object>'

Message = '<message RcvrName=' ObjName '>'
          MsgText
          '</message>'
```



```
ObjName =  {'A' | ... | 'Z' | 'a' | ... | 'z' |
            '0' | ... | '9' | '_'}+
ClsName = DLLName '.' CoClassName

CfgText, MsgText - произвольное XML-выражение, либо
текст
DLLName, CoClassName – имена библиотеки и класса
```

Выполните действия:

1) Дополните перечень компонентов, входящих в состав ИЭС по аналогии с системой Diagnosis. В состав оболочки уже включено большинство необходимых компонентов.
2) Опишите конфигурационную информацию для каждого включаемого компонента (секции object). Обычно конфигурация включает в себя имена файлов с базой знаний, сценарием диалога, VB-скриптами и настройки программных средств.

Стартовым компонентом обычно является диалоговый компонент, поэтому команда запуска уже прописана в секции Message.

3. Реализация прототипа

На данном этапе студенты разрабатывают сценарий диалога и подсистему объяснения в режиме инженера по знаниям, осуществляют настройку средств вывода и других компонентов, разрабатывают сценарий взаимодействия со средством формирования отчетности.

Разработка сценария диалога

Для ознакомления с возможностями языка описания сценариев диалога студенту предлагается самостоятельно ознакомиться с примером в папке Demo – Dialoger. Пример содержит в себе описание сценариев по вводу данных различных типов, в том числе с учетом НЕ-факторов, и сценариев выдачи данных пользователю. Даются примеры всех конструкций языка описания сценариев диалога независимо от проблемной области.

Так, для ввода данных можно использовать конструкции типа “input … as …” для следующих типов:

String - произвольная строка,
Number - произвольное число,
Variant - значение из списка,
Checked – булево значение,

InexactNumber - неточное число,

IndefVariant - значение с одним фактором уверенности,

SubdefVariant - значение с двумя факторами уверенности.

Далее рассматриваются примеры сценариев диалога оболочки и демонстрационной системы Diagnosis. Пример главного сценария по созданию основного рабочего окна системы:

```
scenario Main;
  send СоздатьГлавнуюФорму;
end;

message СоздатьГлавнуюФорму to Alternativer;
  line 'set Caption to $'Оболочка ЭС (С) МИФИ$'';
  line 'set PictureFile to $'example.bmp$'';
  /формирование меню/
  line 'on $'Файл/Выход$' execute Выход';
  line 'on $'Консультация/Начать сеанс 1$' execute
НачатьСеанс1';
  line 'on $'Консультация/Трасса вывода$' execute
ПоказатьОбъяснения';
  line 'on $'Консультация/Объяснения$' execute
ЗапускОбъяснителя';
  line 'on $'Инструменты/Редактор базы знаний...$'
execute ЗапускРедактора';
  line 'on $'Инструменты/Редактор сценариев
диалога...$' execute ЗапускРедактораСценария';
  line 'on $'Помощь/О программе...$' execute
ВыводПриглашения';
  line 'activate';
end;
```

Как видно из примера, с помощью инструкций line на специальном подъязыке формируются команды для внутреннего компонента Alternativer, который отвечает за создание главной формы и меню. Так, с помощью конструкций “set Caption to …” и “set PictureFile to …” устанавливается внешний вид окна, а с помощью конструкций “on … execute …” формируется главное меню окна, где к каждому пункту меню привязывается сценарий.

Рассмотрим примеры вызова инструментов инженера по знаниям:

```
subscenario ЗапускРедактораБазыЗнаний;
  send '<message ProcName="Run"></message>' to
KBEditor;
end;

subscenario ЗапускРедактораСценария;
```



```
  send '<message ProcName="Run"></message>' to
DSDLEditor;
end;

subscenario ПоказатьОбъяснения;
  send '<message
ProcName="TKnowledgeBase.ShowTrassa"></message>' to
ESKernel;
end;
```

Все инструменты расположены в динамических библиотеках, поэтому для их вызова отсылается XML-сообщение специального вида, понимаемое соответствующим компонентом KBEditor, DSDLEditor, ESKernel и т.д. Все компоненты должны быть описаны в конфигурационном файле.

Пример запуска решателя и очистки фактов:

```
subscenario ЗапускРешателя;
  send '<message ProcName="TWorkMemoryConfigurator"/>'
to ESKernel;
  send '<message ProcName="TSolve"/>' to ESKernel;
end;

subscenario ОчисткаФактов;
  set #ОБЪЕКТ1.АТРИБУТ1# to '';
  set #ОБЪЕКТ1.АТРИБУТ2# to '';
  set #ОБЪЕКТ1.АТРИБУТ3# to '';
  send '<message
ProcName=$'TKnowledgeBase.ClearWorkMemory$'></message>'
to ESKernel;
end;
```

Пример сообщения с вопросом о значении строкового атрибута:

```
message УзнатьАтрибут2 to Asker about
#ОБЪЕКТ1.АТРИБУТ2#;
  line 'set Caption to $'Выявление симптомов$'';
  line 'output $'Симптом1. Пациент кашляет?$' as
Question';
  line 'input $'$' to #ОБЪЕКТ1.АТРИБУТ2#
        as Variant from [$'да$', $'нет$'];
  line 'activate';
end;
```

Пример сообщения с вопросом о значении числового атрибута.

```
message УзнатьАтрибут3 to Asker about
#ОБЪЕКТ1.АТРИБУТ3#;
  line 'set Caption to $'Выявление симптомов$'';
```

```
  line 'output $'Симптом3. Какая температура?$' as
Question';
  line 'input $'Число:$' to #ОБЪЕКТ1.АТРИБУТ3# as
Number';
  line 'activate';
end;
```

Пример сценария сеанса консультации.

```
subscenario НачатьСеанс1;
  execute ОчисткаФактов;

  send УзнатьАтрибут4;
  send УзнатьАтрибут3;
  send УзнатьАтрибут2;

  execute ЗапускРешателя;
  execute ВыводДиагноза;
end;
```

Как видно из примера, структура сценария сеанса консультации такова: инициализация рабочей памяти, ввод начальных данных, запуск универсального решателя и выдача результатов. В ходе сеанса возможны уточняющие поддиалоги, если для соответствующих атрибутов предусмотрены сценарии ввода значений.

Пример сценария выдачи результата.

```
subscenario ВыводДиагноза;
  send 'set Caption to $'Значения атрибутов$'' to
Informer;
  send concat('output ',
    string(concat('Болен = "', #ОБЪЕКТ1.АТРИБУТ1#,
'"')),
    'as String on Left') to Informer;
  send concat('output ',
    string(concat('ОРЗ = "', #ОБЪЕКТ1.АТРИБУТ5#, '"')),
    'as String on Left') to Informer;
  send concat('output ',
    string(concat('Грипп = "', #ОБЪЕКТ1.АТРИБУТ6#,
'"')),
    'as String on Left') to Informer;
  send 'activate' to Informer;
end;
```

Разработка подсистемы объяснения

Создание подсистемы объяснения имеет смысл начать с автоматической генерации записанных объяснений по сформированной базе знаний, воспользовавшись меню “Инструменты\Сгенерировать записанные объяснения”.

Далее следует отредактировать записанные объяснения, вставив логические связки в условия правил и исправив падежи. Имя файла с объяснениями следует указать в конфигурационной информации для подсистемы объяснения.

Разработка сценария взаимодействия со средством формирования отчетности

Данный шаг является опциональным и выполняется в случае избытка времени, отпущенного на выполнение лабораторной работы. Далее приводится пример запуска сценария формирования отчета о сеансе консультации в Excel:

```
subscenario СформироватьОтчетExcel;
  send concat('<message ProcName="Run">',
  '  <func name="form" module="report">',
  '    <param type="string">Иванов И.И.</param>',
…
  '    <param type="string">',
         #ОБЪЕКТ1.АТРИБУТ2#,
  '    </param>',
…
  '  </func>',
  '</message>') to Scripter;
end;
```

Как видно из примера, запуск сценария происходит через обращение к компоненту Scripter. Сценарий с именем Form размещается в модуле report (файл report.vbs) и представляет собой процедуру, в которую передаются параметры – значения некоторых атрибутов.

Далее студенту предлагается самостоятельно ознакомиться с процедурой Form и создать аналогичный сценарий для своей проблемной области. По сути, процедура открывает шаблон report.xls и заполняет ячейки шаблона конкретными значениями, полученными в результате работы универсального решателя. Эти значения могут быть переданы как напрямую через параметры (как в системе Diagnosis), так и взяты с классной доски с помощью глобальной переменной bb из раздела “bb.wm.facts”.

Настройка универсального решателя

В конфигурационной информации для компонента ESKernel (универсальный решатель) выберите режим вывода: прямой (forward), обратный (backward) или смешанный (mixed) и укажите методы обработки недостоверных знаний (bayes/dempster/none).

4. Отладка прототипа для различных стратегий вывода в ре-

жиме консультации

На данном этапе студенту необходимо проверить работу прототипа для различных стратегий вывода в режиме консультации и удостовериться, что система дает корректные рекомендации в соответствии с придуманными правилами.

2.4. Домашнее задание

Следующая лабораторная работа посвящается вопросам интеграции прототипа ИЭС с базами данных: прототип будет расширен тематической базой данных, работа с которой будет проводиться в ходе сеанса консультации.

В связи с этим, студенту необходимо продумать информационную модель базы данных; выявить ряд параметров-характеристик объектов предметной области, значения которых могут быть получены в результате запроса к базе данных с определенными критериями, например, на основе значений других атрибутов. Также требуется построить модель базы данных и подготовить SQL-запросы. Кроме того, база данных может хранить в себе какую-либо информацию, полезную для выдачи в ходе сеанса консультации.

Лабораторная работа №3. Интеграция с базами данных

Целью работы является получение навыков интеграции ядра ИЭС с базой данных.

Работа считается выполненной, если продемонстрировано взаимодействие прототипа с базой данных, причем значение одного из атрибутов должно вычисляться в результате обращения к базе данных.

Предполагается, что студент, приступающий к лабораторной работе, имеет представление об архитектуре интегрированной экспертной системы, создаваемой средствами комплекса АТ-ТЕХНОЛОГИЯ, об интерфейсах обращения к базам данных ADO, ODBC, а также знаком с языком интеграции приложений Visual Basic или JavaScript.

3.1. План работы

1. Знакомство с демонстрационным прототипом работы с базой данных Demo-Database.
2. Модификация базы знаний: расширение базы знаний атрибутом, значение которого планируется брать из базы данных.
3. Создание базы данных в СУБД MS Access и наполнение ее данными.
4. Создание сценария для работы с базой данных на основе SQL-запроса с помощью интерфейсов ODBC и ADO.
5. Доработка сценария диалога для запуска сценария обращения к базе данных.
6. Отладка работы прототипа

3.2. Принципы интеграции прикладных программ и СУБД с ядром ИЭС, разрабатываемых с помощью комплекса АТ-ТЕХНОЛОГИЯ

В инструментальном комплексе АТ-ТЕХНОЛОГИЯ роль связующего звена между компонентами прототипа ИЭС и внешними прикладными программами и базами данных играет компонент Scripter. Компонент позволяет вызывать внешние процедуры и функции, написанные на скриптовых языках типа Javascript и VB

Script, и использует стандартный интерпретатор Windows - Microsoft Script Engine, реализующий вызов интерпретируемых функций с передачей параметров и возвратом значения.

Указанные языки позволяют использовать серверы автоматизации и ActiveX-библиотеки, поэтому особенно удобна работа с прикладными программами, поддерживающими COM-технологию. Кроме того, для разработки сценариев взаимодействия с внешними программными средствами не требуется компиляция вызывающей программы.

В рамках прототипа ИЭС для работы компонента требуется разработка модулей на языке VBScript, содержащих параметризованных специализированных функций, и разработка сценариев их вызова на языке описания сценариев диалога.

Использование компонента накладывает следующие требования на загружаемые модули:

1. В каждом модуле (файле *.vbs) должны быть объявлены следующие глобальные переменные:

CurDir – переменная, в которую будет помещаться путь к текущему каталогу при загрузке модуля в память.

bb - переменная, в которую будет помещаться ссылка на экземпляр компонента “Классная доска”.

2. Также требуется наличие метода SetGlobals, устанавливающего значения глобальных переменных CurDir и bb.

Таким образом, любой используемый модуль должен включать следующие объявления:

```
Dim CurDir
Dim bb

Sub SetGlobals(aCurDir, aBB)
  CurDir = aCurDir
  Set bb = aBB
End Sub
```

Конфигурационная информация для компонента Scripter включает значения следующих параметров:

Timeout – время ожидания выполнения процедуры или функции. Значение 0 соответствует бесконечному ожиданию. Если по истечении времени ожидания процедура или функция не возвращает управление, то выполнение процедуры прерывается компонентом.

Language – язык, на котором написаны загружаемые модули. Все загружаемые модули, используемые вместе, должны быть написаны на одном и том же языке. Возможны следующие варианты JavaScript или VBScript.

Silent – признак того, что не следует выводить сообщения об ошибках на экран.

Кроме того, в конфигурации описываются загружаемые модули (и пути к ним, если нужно).

Пример конфигурационной информации:

```
<config>
  <script Timeout="0" Language="VBScript"
Silent="false" />
  <file>db.vbs</file>
  <file>report.vbs</file>
</config>
```

Для того, чтобы вызвать какую-либо функцию модуля необходимо отослать XML-сообщение компоненту Scripter. Пример сообщения по вызову функции GetAttrValue из модуля Database.vbs показан ниже:

```
<message ProcName="Run">
  <func name="GetAttrValue" module="database">
    <param type="number">120</param>
  </func>
</message>
```

Как видно из примера, функции передается один параметр, который будет преобразован к числовому типу. Кроме чисел параметры можно преобразовать к строковому типу (string, по умолчанию) и логическому (boolean).

Методические указания к работе

1. Знакомство с демонстрационным прототипом работы с базой данных Demo-Database

1) Запустите демонстрационный прототип demodb.exe и откройте редактор базы знаний. В базе знаний имеется правило:

```
Если
  планируется использование управляемых коммутаторов
и
  существуют коммутаторы с поддержкой NWay,
то
  использование технологии NWay возможно,
иначе
```

```
использование технологии NWay невозможно.
```

В соответствии со сценарием диалога, значение атрибута “планируется использование управляемых коммутаторов” запрашивается у пользователя. Значение атрибута “существуют коммутаторы с поддержкой NWay” рассчитывается в результате отбора подходящих коммутаторов в базе данных сетевого оборудования.

2) Пройдите сеанс консультации.

Следует отметить, что максимальное число портов у коммутаторов, поддерживающих технологию NWay – 8, что видно по таблице Hub базы данных Network. Во время сеанса консультации будет осуществлено обращение к базе данных, а результаты обращения будут выведены на экран (Рис. 19):

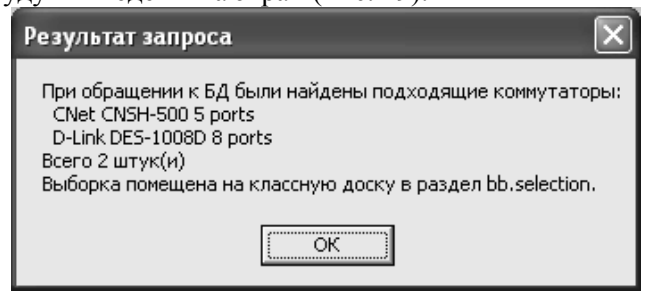

Рис. 19 Результаты обращения к базе данных в ходе сеанса консультации

После осуществления выборки из базы данных функция помещает значение атрибута на классную доску в раздел bb.wm.facts, делая значения атрибута видимым для универсального решателя. Кроме того, результаты выборки также помещаются на классную доску в специальный раздел bb.selection для дальнейшего использования.

3) Просмотрите содержимое раздела классной доски bb.selection с помощью меню “Инструменты\Классная доска”.
4) Запустите подсистему объяснения прототипа.

Обратите внимание, что для атрибута, значение которого было получено из базы данных, объяснения также формируются автоматически. Пример объяснений показан на Рис. 20. Следует отметить, что для построения объяснений типа “Как получено значение параметра?” подсистема объяснений анализирует модули с именами DB и Database.

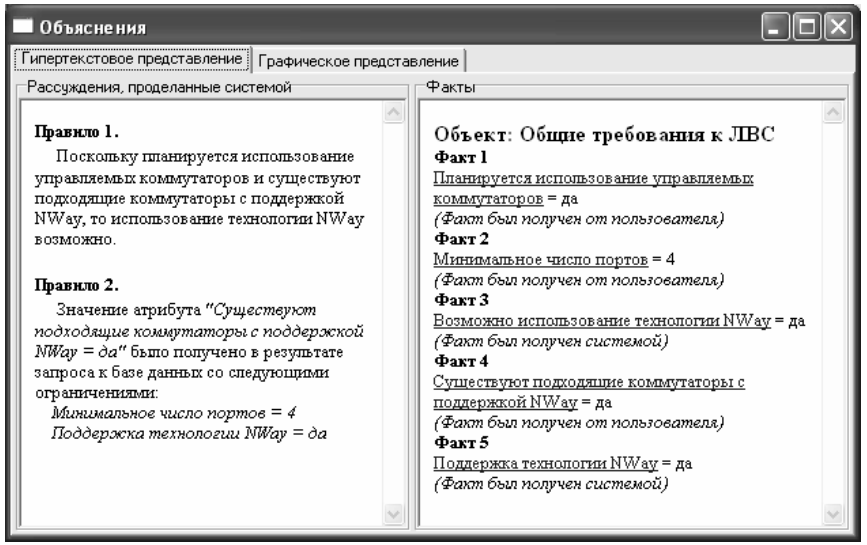

Рис. 20 Пример объяснений при получении значения атрибута из базы данных

5) Откройте редактор сценария диалога.

Сообщение для ввода значения атрибута “существуют коммутаторы с поддержкой NWay” выглядит следующим образом:

```
message УзнатьАтрибут4 to Scripter about
#ОБЪЕКТ1.АТРИБУТ4#;
  line concat(
    '<message ProcName="Run">',
    '  <func name="GetAttrValue" module="database">',
    '    <param type="number">', #ОБЪЕКТ1.АТРИБУТ2#,
    '    </param>',
    '    <param type="string">', #ОБЪЕКТ1.АТРИБУТ5#,
    '    </param>',
    '  </func>',
    '</message>');
end;
```

Далее студенту предлагается самостоятельно рассмотреть функцию GetAttrValue в модуле database. В соответствии с определением функции, подходящим коммутатором считается коммутатор с числом портов не меньшим чем значение первого параметра и поддерживающим технологию NWay. Тело функции делится на три части: формирование SQL-запроса и выборка данных, помещение значение атрибута на классную доску в раздел фактов, помещение выборки на классную доску в раздел bb.selection.

2. Модификация базы знаний

Для выполнения лабораторной работы необходимо расширить базу знаний атрибутом, значение которого планируется брать из базы данных.

1) Определите тип атрибута. Добавьте атрибут в базу знаний.
2) Далее следует учесть значение данного атрибута в посылке существующего правила, либо создать новое правило.

Дополнительно можно определить атрибуты-параметры запроса к базе данных с произвольным типом. Они будут нужны для автоматического построения объяснений.

3. Создание базы данных в СУБД MS Access

Для подготовки базы данных выполните следующее:

1) Создайте структуру базы данных в СУБД MS Access и сохраните базу данных в папке с вашим прототипом.
2) Наполните таблицы данными.
3) Создайте ODBC-соединение с базой данных. Для этого откройте меню "Пуск" → "Настройка" → "Панель управления" → "Администрирование" → "Источники данных (ODBC)" → Закладка "Системный DSN".
4) Нажмите кнопку "Добавить", выберите "Microsoft Access Driver" и нажмите кнопку "Готово".
5) Введите имя соединения demodb и нажмите кнопку "Выбрать". Укажите путь к файлу с вашей базой данных и нажмите "OK". Закройте окно "Источники данных (ODBC)".

4. Доработка сценария диалога для запуска скрипта

Если значение атрибута запрашивается у пользователя, то сообщение о вводе данных отсылается компоненту Asker. Если же значение запрашивается из базы данных, то сообщение следует направить к компоненту Scripter, который вызовет указанный сценарий доступа к базе данных.

Аналогично демонстрационному примеру Demo-Database, далее дополните сценарий диалога нужным сообщением для активации компонента Scripter. Укажите необходимые параметры при вызове процедуры для выполнения SQL-запроса. Параметрами могут быть как конкретные значения, так и атрибуты из рабочей памяти.

5. Создание сценария для работы с базой данных

Выполните следующие действия:

1) Создайте модуль database.vbs и включите его в состав разрабатываемого прототипа.

2) Разработайте процедуру расчета значения атрибута базы знаний на основе обращения к базе данных.

Для обращения к базе данных удобно использовать ADO компоненты, SQL-запрос и ODBC-соединение, как показано в следующем примере:

```
Dim Connection
Set Connection = CreateObject("ADODB.Connection")
Connection.ConnectionTimeOut = 15
Connection.CommandTimeOut = 300
Connection.Open "demodb", "", ""

strSelect =
  "select [name], ports, speed, price from Hub where
  (ports >= " + CStr(MinPortNum) + ") and nway"

Dim RS
Set RS = CreateObject("ADODB.Recordset")

RS.Open strSelect, Connection, 1, 1
RS.MoveFirst()
while not RS.EOF
  'обработка данных
  …
  RS.MoveNext()
wend
RS.Close
Connection.Close
```

Получив значение атрибута от пользователя, компонент Asker помещает его значение на классную доску в раздел bb.wm.facts. Поэтому сценарий доступа к данным в модуле database также должен предусматривать работу с классной доской. Например, следующим образом:

```
call bb.FindObject("bb.wm.facts.fact", "AttrPath",
  "ОБЪЕКТ1.АТРИБУТ4", i)
if i < 0 then
  call bb.AddObject("bb.wm.facts.fact", exists)
  call bb.GetChildCount("bb.wm.facts", "fact", i,
    exists)
  i = i-1
end if

bb.SetParamValue "bb.wm.facts.fact["+CStr(i)+"]",
  "AttrPath", "ОБЪЕКТ1.АТРИБУТ4", exists
bb.SetParamValue "bb.wm.facts.fact["+CStr(i)+"]",
```

```
  "Value", str, exists
bb.SetParamValue "bb.wm.facts.fact["+CStr(i)+"]",
  "Belief", "100", exists
bb.SetParamValue "bb.wm.facts.fact["+CStr(i)+"]",
  "Accuracy", "100", exists
```

Здесь, сначала рассчитывается индекс последнего факта на классной доске, а затем на классной доске создается новый факт для атрибута с идентификатором ОБЪЕКТ1.АТРИБУТ4 и со значением, хранимым в переменной str.

3) Дополните сценарий действиями по помещению значения запрашиваемого атрибута на классную доску в раздел bb.wm.facts.

6. Отладка работы прототипа

Перед сдачей работы преподавателю необходимо проверить работу сценария обращения к базе данных для различных значений параметров запроса.

3.4. Домашнее задание

Следующая лабораторная работа посвящена вопросам интеграции прототипа ИЭС с прикладными программами и разработкой нового информационного компонента.

Студент должен придумать новый атрибут и правило, причем значение атрибута должно вычисляться по определенным формулам на основе значений других атрибутов в пакете MS Excel.

Также необходимо расширить архитектуру прототипа новым информационным компонентом, который разрабатывается самостоятельно:

1) Придумать функциональность компонента и описать сценарий работы с компонентом в ходе сеанса консультации.
2) Разработать формат обрабатываемых сообщений.
3) Разработать компонент типа Automation Server в DLL-библиотеке в среде Delphi в соответствии с требованиями комплекса АТ-ТЕХНОЛОГИЯ на разработку компонентов для ИЭС.

Лабораторная работа №4. Интеграция с прикладной программой вычислительного характера

Целью работы является получение навыков интеграции ядра ИЭС с пакетом прикладных программ на примере MS Excel, а также получение навыков создания компонентов ИЭС на примере придуманного заранее.

Работа считается выполненной, если продемонстрирован сеанс консультации, в котором отчет о сеансе представляется в виде книги MS Excel, и используется функциональность разработанного компонента.

Предполагается, что студент, приступающий к лабораторной работе, имеет представление о COM-технологии, интерфейсах автоматизации приложений (в частности Excel), знаком с языком интеграции приложений Visual basic.

4.1. План работы

1. Модификация базы знаний: расширение базы знаний атрибутом, значение которого планируется рассчитывать в Excel.
2. Подготовка расчетного документа в Excel.
3. Создание сценария работы с Excel.
4. Включение информационного компонента в состав прототипа и создание сценария работы с компонентом.
5. Доработка сценария диалога и отладка работы прототипа.

4.2. Программный интерфейс компонентов в составе прототипа интегрированной экспертной системы

Любой компонент ИЭС должен обладать следующими свойствами и методами:

Name: WideString – свойство строкового типа, хранящее имя данного компонента. Значение свойства устанавливается брокером при регистрации данного компонента и не должно изменяться в течение всего времени работы ИЭС.

Broker: OleVariant – свойство, хранящее ссылку на брокера. Значение свойства устанавливается брокером при регистрации данного компонента и в дальнейшем может быть использовано

компонентом для вызова методов брокера.

Configurate(Config: WideString) – метод, осуществляющий конфигурирование данного компонента, причем в Config содержится информация, полученная на этапе реализации ИЭС (т.е. в комплексе АТ-ТЕХНОЛОГИЯ). После окончания конфигурирования компонент должен быть полностью готов к работе. Данный метод вызывается в начале работы прототипов ИЭС, причем в качестве параметра передается текст, описанный в конфигурационном файле config.xml для данного компонента.

ProcessMessage(SndrName, MsgText: WideString; out Output: OleVariant) – метод, осуществляющий обработку входящих сообщений. Здесь, в параметре MsgText передается тело сообщения. Принято, что сообщение представляет собой XML-текст следующего формата:

```
MsgText = ProcCall
ProcCall = '<message ProcName = ' ProcName '>'
           {Param}* | ParamXML
           '</message>'
Param = '<' ParamName '>'
        ParamXML
        '</' ParamName '>'
ProcName = Id
ParamName = Id
Id =  {'A'| ... |'Z' |'a' |... |'z' |'0' |... |'9'}+
------------------------------
Id - Первая литера в каждом слове Id - заглавная,
     остальные - прописные.
     Примеры: Run, AttrName, Attr12 и т.п.
ParamXML - произвольное XML-выражение, либо текст.
```

Допускается определение любого числа обрабатываемых сообщений. Принято, что сообщения имеют разные имена ProcName. В качестве параметров может выступать список тегов Param, либо любой другой XML-текст, понимаемый компонентом.

Stop() – метод, осуществляющий остановку выполнения текущей функции компонентом. В случае если на момент вызова этого метода компонент не выполнял никакой работы, команда должна игнорироваться. Важно отметить, что остановка, так же как и активизация компонентов может производиться неоднократно в процессе функционирования ИЭС.

Перечисленные свойства и методы представляют собой минимальный набор и могут быть расширены.

4.3. Методические указания к работе

Дальнейшие методические указания приводятся для примера медицинской диагностики - расчет дозы рекомендованного лекарства в зависимости от возраста и веса пациента с целью назначения курса лечения обнаруженного заболевания.

Вес необходимой дозы будет рассчитываться на основе следующих правил в Excel: на полных 10 кг веса требуется доза лекарства в 1 мг. Рекомендованная доза рассчитывает следующим образом: если возраст меньше 2 лет, то использовать четверть нормы, если возраст меньше 15 лет, то использовать половину нормы, иначе использовать полную норму.

1. Модификация базы знаний

1) Откройте редактор базы знаний и добавьте в базу знаний числовой атрибут, представляющий собой дозу лекарства в миллиграммах “Доза лекарства (мг)”.
2) Добавьте в базу знаний числовые атрибуты “Возраст пациента” и “Вес пациента”, на основе которых будет рассчитывать норма лекарства.

2. Подготовка расчетного документа в Excel

Для создания расчетного шаблона понадобятся встроенные функции Excel **ОТБР** – функция отбрасывания дробной части и **ВПР** – функция поиска значения в таблице.

Функция ОТБР принимает один числовой параметр и возвращает целую часть. Функция будет применяться для расчета дозы с учетом веса.

Функция ВПР будет применяться для поиска в таблице возрастов нужного коэффициента уменьшения дозы. Функция принимает три параметра: 1 – значение, которое требуется найти в первом столбце диапазона, 2 – диапазон ячеек, 3 – номер столбца диапазона, из которого следует взять результат для возврата.

Сформируйте расчетный шаблон на основе Табл. 1.

Табл. 1 Пример расчета нормы лекарства в Excel

	A	B
1	**Возраст (лет)**	67
2	**Вес (кг)**	58
3	**Норма (мг) на 10 кг**	12,00
4	**Расчетная доза (мг)**	60,00
5	**Доза с учетом возраста (мг)**	**60,00**

6		
7	**Возраст пациента**	**Коэффициент**
8	0	0,25
9	2	0,50
10	15	1,00

3. Создание сценария работы с Excel

Для создания сценария для работы с Excel на Visual Basic удобно воспользоваться примером функции формирования отчета в Excel системы Diagnosis.

1) Для расчета необходимо поместить в ячейку B1 значение атрибута “Возраст пациента”, а в ячейку B2 - “Вес пациента”.
2) Значение в ячейке B5 (доза с учетом возраста (мг)) будет рассчитываться автоматически. Данное значение следует поместить в качестве значения атрибута “Доза лекарства (мг)” на классную доску.
3) Вывести рекомендованную дозу на экран конечному пользователю.

Пример расчетов приведен в Табл. 1.

После отладки взаимодействия с Excel следует перейти ко второй части лабораторной работы.

4. Включение информационного компонента в состав прототипа и создание сценария работы с компонентом

1) С помощью редактора конфигурационной информации добавьте разработанный компонент в состав прототипа, указав его название и ProgID.
2) Если компонент предусматривает конфигурирование при запуске, то опишите нужную конфигурацию.
3) Сохраните конфигурацию и перезапустите прототип.

5. Доработка сценария диалога и отладка работы прототипа

В сценарии диалога опишите пример обращения к компоненту. Компонент может вызываться из главного меню, в ходе сеанса консультации, с помощью сценария Visual Basic через компонент Scripter. По завершению отладки продемонстрируйте работу прототипа преподавателю.

Заключение

Целью данного лабораторного практикума являлось обучение студентов технологии построения такого класса интеллектуальных систем как интегрированные экспертные системы в среде инструментального комплекса АТ-ТЕХНОЛОГИЯ.

В ходе выполнения лабораторных работ студенты поочередно оказываются в роли эксперта, инженера по знаниям, программиста, конечного пользователя. Благодаря слиянию парадигм инженерии знаний и традиционного программирования, усилению программной составляющей лабораторных работ удается выйти за рамки “игрушечной” экспертной системы и явно показать преимущества интегрированных экспертных систем.

Список рекомендованной литературы

1.Гаврилова Т. А., Хорошевский В. Ф. Базы знаний интеллектуальных систем. СПб: Питер, 2000. – 384 с.

2.Джексон П. Введение в экспертные системы. М: Издательский дом “Вильямс”, 2001. – 624 с., ил.

3.Лорьер Ж.-Л. Системы искусственного интеллекта. М: Мир, 1991. – 568 с., ил.

4.Попов Э. В. Экспертные системы. Решение неформализованных задач в диалоге с ЭВМ. М: Наука, 1987. – 288 с.

5.Попов Э. В., Фоминых И. Б., Кисель Е. Б., Шапот М. Д. Статические и динамические экспертные системы. М: Финансы и статистика, 1996. – 320 с.

6.Поспелов Д. А. Моделирование рассуждений. Опыт анализа мыслительных актов. М: Радио и связь, 1989. – 184 с.

7.Рассел С., Норвиг П. Искусственный интеллект: современный подход, 2-ое изд. Пер. с англ. – М.: Издательский дом «Вильямс», 2006. – 1408 с., ил.

8.Рыбина Г. В. Автоматизированное построение баз знаний для интегрированных экспертных систем. В кн.: Известия РАН. Теория и системы управления. 1998, №5, с. 152-166.

9.Рыбина Г.В. Введение в интеллектуальные системы: Учебное пособие. М.: МИФИ, 2006. – 140 с.

10. Рыбина Г. В. Проектирование систем, основанных на знаниях. Учебное пособие. М: МИФИ, 2000. – 104 с.

11. Рыбина Г.В., Демидов Д.В. Модели, методы и программные средства вывода в интегрированных экспертных системах // Инженерная физика. №2, 2007. с.51-60.

12. Рыбина Г. В., Душкин Р.В., Демидов Д.В. Модели и методы обработки недостоверных знаний в инструментальном комплексе АТ-ТЕХНОЛОГИЯ // Интегрированные модели и мягкие вычисления в искусственном интеллекте. Сб. научных трудов. М.: Физматлит, 2003, с.401-407.

13. Рыбина Г.В., Пышагин С.В., Смирнов В.В., Левин Д.Е., Душкин Р.В. Инструментальный комплекс АТ-ТЕХНОЛОГИЯ для поддержки разработки интегрированных экспертных систем. М: МИФИ, 2001.

14. Форсайт Ф. Экспертные системы. Принципы работы и примеры. М: Радио и связь, 1987. – 224 с.

15. Частиков А.П., Гаврилова Т.А. Белов Д.Л. Разработка экспертных систем. Среда CLIPS. – СПб.: БХВ – Петербург, 2003. – 608 с.

16. Ярушкина Н.Г. Основы теории нечетких и гибридных систем. М.: Финансы и статистика, 2004. - 320 с.: ил.

Также рекомендуется литература по COM-технологии, по технике работы с базами данных через интерфейсы ODBC и ADO, по программированию на языке Visual Basic, среде Delphi, языку XML

Приложение 1. Язык представления знаний

Для проведения лабораторных работ по курсу “Проектирование систем, основанных на знаниях” используется инструментальный комплекс АТ-ТЕХНОЛОГИЯ. Далее описывается язык представления знаний (ЯПЗ) комплекса.

Алфавит

Приведем основные символы языка:

Идентификатор – текстовая строка, идентифицирующая тип, объект, атрибут, правило, отношение, группу. На идентификаторы накладываются ограничения: идентификатор типа имеет вид ‘ТИП’n, где n – порядковый номер типа; идентификатор объекта имеет вид ‘ОБЪЕКТ’n, где n – порядковый номер объекта; идентификатор атрибута имеет вид ‘АТРИБУТ’n, где n – порядковый номер атрибута в объекте; идентификатор правила имеет вид ‘ПРАВИЛО’n, где n – порядковый номер правила и т.п. Например, ТИП2, АТРИБУТ5, ПРАВИЛО234.

Название – строка символов после ключевого слова КОММЕНТАРИЙ до конца строки, незаключенная в кавычки.

Строка – строка символов, заключенная в двойные кавычки.

Ключевое слово – специальный символ ЯПЗ, несущий определенную семантическую нагрузку для интерпретатора. В ЯПЗ используются следующие ключевые слова:

АТРИБУТ	**ИНАЧЕ**	**СВЯЗИ**
АТРИБУТЫ	**ИСТОЧНИК**	**СИМВОЛ**
ГРУППА	**КОММЕНТАРИЙ**	**ТИП**
ДАННЫЕ	**НЕЧЕТКИЙ**	**ТО**
ДО	**ОБЪЕКТ**	**ТОЧНОСТЬ**
ЕСЛИ	**ОТ**	**УВЕРЕННОСТЬ**
ЗНАЧЕНИЯ	**ПРАВИЛО**	**УПРАВЛЕНИЕ**
ИМЯ	**ПРИЕМНИК**	**ЧИСЛО**

Знак – специальный символ второстепенного значения. В

ЯПЗ используются следующие знаки:

&|~()<=>{};.‘”[]+-*/

Синтаксис ЯПЗ инструментального комплекса АТ-ТЕХНОЛОГИЯ

Приведем синтаксис ЯПЗ в расширенной форме Бэкуса-Наура:

```
база_знаний =
  {определение_типа}
  {определение_объекта}
  {определение_связи}
  {правило}
  [словарь_лингвистических_переменных]

определение_типа =
  определение_числового_типа |
  определение_символьного_типа |
  определение_нечеткого_типа

определение_числового_типа =
  ‘ТИП’ имя_типа
  ‘ЧИСЛО’
  ‘ОТ’ минимальное_значение
  ‘ДО’ максимальное_значение
  ‘КОММЕНТАРИЙ’ строка120

определение_символьного_типа =
  ‘ТИП’ имя_типа
  ‘СИМВОЛ’
  ‘ЗНАЧЕНИЯ’
  {значение}
  ‘КОММЕНТАРИЙ’ строка120

определение_нечеткого_типа =
  ‘ТИП’ имя_типа
  ‘СИМВОЛ’
  ‘ЗНАЧЕНИЯ’
  {значение}
  ‘НЕЧЕТКИЙ’ число_ФП {определение_ФП}+
  ‘КОММЕНТАРИЙ’ строка120

определение_объекта =
  ‘ОБЪЕКТ’ имя_объекта
  [‘ГРУППА’ имя_группы]
  ‘АТРИБУТЫ’ список_атрибутов
```

```
'КОММЕНТАРИЙ' строка120

определение_атрибута =
  'АТРИБУТ' имя_атрибута
  'ТИП' имя_типа
  'КОММЕНТАРИЙ' строка120

определение_связи =
  ('ДАННЫЕ' | 'УПРАВЛЕНИЕ')
  'ИМЯ СВЯЗИ' имя_связи
  'ИСТОЧНИК' имя_объекта
  'ПРИЕМНИК' имя_объекта
  'КОММЕНТАРИЙ' строка120

правило =
  'ПРАВИЛО' имя_правила
  'ЕСЛИ' логическое_выражение
  'ТО' {действие}
  ['ИНАЧЕ' {действие} ]
  'КОММЕНТАРИЙ' строка120

логическое_выражение =
  (логическое_выражение '&' логическое_выражение) |
  (логическое_выражение '|' логическое_выражение) |
  ('~'логическое_выражение) |
  ('('логическое_выражение ')') |
  утверждение

утверждение =
  утверждение_для_объектов |
  утверждение_для_числового_атрибута |
  утверждение_для_символьного_нечеткого_атрибута

утверждение_для_объектов =
  имя_объекта имя_связи имя_объекта

утверждение_для_числового_атрибута =
  имя_объекта '.' имя_атрибута отношение
арифм_выражение
  точность ФУ

арифм_выражение =
  значение |
  (арифм_выражение оператор арифм_выражение ) |
  ('(' арифм_выражение ')') |
  ('-' арифм._выражение) |
  (имя_объекта '.' имя_атрибута)
утверждение_для_символьного_нечеткого_атрибута =
```



```
  имя_объекта '.' имя_атрибута '=' значение точность ФУ

отношение = '<' | '>' | '=' | '>=' | '<=' | '!='
оператор = '+' | '-' | '*' | '/'
точность = 'ТОЧНОСТЬ' число100
ФУ = 'УВЕРЕННОСТЬ [' число100 ';' число100 ']'

действие =
  утверждение_для_числового_атрибута |
утверждение_для_символьного_нечеткого_атрибута |
  ограничение | вызов_процедуры

словарь_лингвистических_переменных =
  'СЛОВАРЬ'
  число_статей
  {определение_лингвистической_переменной}

определение_лингвистической_переменной =
  имя_ЛП
  число_ФП
  {определение_ФП}+
  'КОММЕНТАРИЙ' строка120

определение_ФП =
  имя_ФП
  начальная_абсцисса
  конечная_абсцисса
  количество_точек
  '={' список_точек '}'

список_точек =
  {абсцисса '|' степень ';'} абсцисса '|' степень

начальная_абсцисса = числоПЗ
конечная_абсцисса = числоПЗ
абсцисса = числоПЗ
степень = число01

имя_… -     идентификатор
значение -  строка, заключенная в двойные кавычки
строкаN -   любая последовательность символов, не
заключенная в кавычки.  Длина строки должна быть не
более N символов.
число -     любое число
число01 -   число с плавающей запятой от 0 до 1
число100 -  целое число от 0 до 100
числоПЗ -   число с плавающей запятой (с точностью до 2
знаков после запятой)
```

Семантика ЯПЗ инструментального комплекса АТ-ТЕХНОЛОГИЯ

Модель и язык представления знаний, используемые на данный момент в комплексе АТ-ТЕХНОЛОГИЯ, можно кратко охарактеризовать следующим образом:

Понятийная структура предметной области выражается в терминах объектов и атрибутов, а также отношениях между объектами.

Все знания о решении задач представляются с помощью продукционных правил. Посылка правила, описывающая ситуацию в проблемной области, представляет собой логическую формулу, атомами которой являются утверждения об объектах и атрибутах.

Недостоверные знания представлены в виде нескольких коэффициентов уверенности в истинности утверждений, входящих в посылку и следствие правил, неточных значений атрибутов, нечетких значений атрибутов и словаря функций принадлежности.

Согласно изложенному выше синтаксису, база знаний структурно представляет собой совокупность следующих блоков:

- Определения типов параметров предметной области (допускается определение символьных типов, числовых типов, имеется также расширение языка для определения нечетких типов данных).
- Описание объектов и их атрибутов, групп объектов и отношений между объектами, где каждый объект представляет собой сущность, обладающую рядом характеристик.
- Описание правил, устанавливающих зависимости между конкретными атрибутами объектов.
- Словарь функций принадлежности, представляющий собой набор определений лингвистических переменных (расширение ЯПЗ для представления нечетких знаний).

Рассмотрим более подробно каждый из указанных блоков.

Описание типов

Язык представления знаний позволяет определять типы на основе трех метатипов: числового, символьного, нечеткого. Числовые типы определяются в соответствии с синтаксисом

определение_числового_типа =

‘**ТИП**’ имя_типа
‘ЧИСЛО’

```
‘ОТ’ минимальное_значение
‘ДО’ максимальное_значение
‘КОММЕНТАРИЙ’ строка120
```

Для числового типа после ключевых слов “ОТ” и “ДО” указываются минимальное и максимальное значение. Для неограниченного диапазона значений следует указать нули.

Символьные типы определяются в соответствии со следующим синтаксисом:

определение_символьного_типа =
‘**ТИП**’ имя_типа
‘СИМВОЛ’
‘ЗНАЧЕНИЯ’

```
{значение}
‘КОММЕНТАРИЙ’ строка120
```

Для символьных типов следует указать список принимаемых значений после ключевого слова “ЗНАЧЕНИЯ”.

Объявление нечеткого типа соответствует введению лингвистической переменной и предусматривает указание предопределенных значений, описываемых функциями принадлежности.

определение_нечеткого_типа =
‘**ТИП**’ имя_типа
‘СИМВОЛ’
‘ЗНАЧЕНИЯ’

```
{значение}
‘НЕЧЕТКИЙ’ число_ФП {определение_ФП}+
‘КОММЕНТАРИЙ’ строка120
```

После ключевого слова “НЕЧЕТКИЙ” следует определить несколько функций принадлежности и сопоставить их с символьными значениями.

Пример определения типов:

```
ТИП ТИП1
ЧИСЛО
ОТ 0
ДО 1024
КОММЕНТАРИЙ Количество компьютеров

ТИП ТИП2
```

```
СИМВОЛ
"На уровне ресурсов"
"На уровне пользователей"
"На уровне ресурсов и на уровне пользователей"
КОММЕНТАРИЙ Способ управления доступом

ТИП ТИП3
СИМВОЛ
"Малая"
"Средняя"
"Большая"
НЕЧЕТКИЙ
3
“Малая” 0 120 5 ={0|0.5; 10|1; 30|0.9; 80|0.3; 120|0}
“Средняя” 0 120 5 ={0|0; 10|0.3; 40|1; 80|0.8; 120|0}
“Большая” 0 120 5 ={0|0; 30|0.3; 50|0.5; 80|0.8; 120|1}
КОММЕНТАРИЙ Скорость
```

Описание объектов, их атрибутов и отношений между объектами

Объекты описывают сущности проблемной области, атрибуты объектов – характеристики сущностей, связи - отношения между объектами. В соответствии с синтаксисом, после ключевого слова “АТРИБУТЫ” помещаются объявления атрибутов, при описании атрибутов следует указывать тип атрибута после ключевого слова “ТИП”:

```
определение_объекта =
   'ОБЪЕКТ' имя_объекта
   ['ГРУППА' имя_группы]
   'АТРИБУТЫ' список_атрибутов
  'КОММЕНТАРИЙ' строка120

определение_атрибута =
   'АТРИБУТ' имя_атрибута
   'ТИП' имя_типа
   'КОММЕНТАРИЙ' строка120

определение_связи =
   'ИМЯ СВЯЗИ' имя_связи
   ('ДАННЫЕ' | 'УПРАВЛЕНИЕ')
   'ИСТОЧНИК' имя_объекта
   'ПРИЕМНИК' имя_объекта
   'КОММЕНТАРИЙ' строка120
```

Унарные и бинарные отношения между объектами описы-

ваются связями двух видов: по данным и по управлению, что указывается после имени связи. Для связи после ключевых слов ИСТОЧНИК и ПРИЕМНИК следует указать имена соответствующих объектов.

Пример описания объекта:

```
ОБЪЕКТ ОБЪЕКТ10
АТРИБУТЫ
АТРИБУТ АТРИБУТ1
ТИП ТИП46
КОММЕНТАРИЙ Фирма-разработчик
АТРИБУТ АТРИБУТ2
ТИП ТИП106
КОММЕНТАРИЙ Область применения сервера
АТРИБУТ АТРИБУТ3
ТИП ТИП107
КОММЕНТАРИЙ Тип процессора
АТРИБУТ АТРИБУТ4
ТИП ТИП109
КОММЕНТАРИЙ Тип сервера
АТРИБУТ АТРИБУТ5
ТИП ТИП110
КОММЕНТАРИЙ Количество процессоров
КОММЕНТАРИЙ Сервер
```

Описание правил

Согласно синтаксису, условие правила описывается после ключевого слова “ЕСЛИ”, действие правила описывается после ключевых слов “ТО” и, если уместно, “ИНАЧЕ” и представляет собой набор утверждений:

```
правило =
   'ПРАВИЛО' имя_правила
   'ЕСЛИ' логическое_выражение
   'ТО' {действие}
   ['ИНАЧЕ' {действие} ]
   'КОММЕНТАРИЙ' строка120
```

Логическое выражение описывается в инфиксной форме с помощью логических операторов И (“&”), ИЛИ (“|”), НЕ (“~”) и круглых скобок и состоит из утверждений об атрибутах и отношениях между объектами.

```
логическое_выражение =
   (логическое_выражение '&' логическое_выражение) |
   (логическое_выражение '|' логическое_выражение) |
   ('~'логическое_выражение) |
```

```
  ('('логическое_выражение ')') |
  утверждение
```

Утверждение представляет собой сопоставление атрибута со значением, причем для атрибутов числового типа сопоставление осуществляется с помощью операторов сравнения, а вместо значения допускается указать арифметическое выражение в инфиксной форме. Атрибут объекта указывается в точечной записи.

```
утверждение_для_числового_атрибута =
  имя_объекта '.' имя_атрибута отношение
арифм_выражение
  точность ФУ
отношение = '<' | '>' | '=' | '>=' | '<=' | '!='

утверждение_для_символьного_нечеткого_атрибута =
  имя_объекта '.' имя_атрибута '=' значение точность ФУ
```

Значениям атрибутов в утверждениях могут быть поставлены в соответствие коэффициенты точности и интервал уверенности, отражающие недостоверность утверждения, в соответствии с синтаксисом:

```
точность = 'ТОЧНОСТЬ' число100
ФУ = 'УВЕРЕННОСТЬ [' число100 ';' число100 ']'
```

Пример определения правил:

```
ПРАВИЛО ПРАВИЛО21
ЕСЛИ
 (ОБЪЕКТ2.АТРИБУТ11="Звезда" |
 ОБЪЕКТ2.АТРИБУТ10="Шина" ) &
 ОБЪЕКТ2.АТРИБУТ27="Arcnet"
ТО
ОБЪЕКТ2.АТРИБУТ8="Arcnet"
КОММЕНТАРИЙ Arcnet

ПРАВИЛО ПРАВИЛО22
ЕСЛИ
 ОБЪЕКТ2.АТРИБУТ8="Ethernet" &
 ОБЪЕКТ2.АТРИБУТ1<="30" &
 ОБЪЕКТ2.АТРИБУТ5<="925"
ТО
 ОБЪЕКТ2.АТРИБУТ22="10Base-2"
 ОБЪЕКТ2.АТРИБУТ23="4"
КОММЕНТАРИЙ 10Base-2
```

Словарь лингвистических переменных

Словарь не является обязательной секцией базы знаний.

После ключевого слова “СЛОВАРЬ” следует указать число определений и описать предопределенные значения лингвистических переменных и функции принадлежности:

```
словарь_лингвистических_переменных =
  'СЛОВАРЬ'
  число_статей
  {определение_лингвистической_переменной}
```

Для определения лингвистической переменной следует указать ее имя, число определяемых функций принадлежности и далее определить поименованные функции согласно синтаксису:

```
определение_лингвистической_переменной =
  имя_ЛП
  число_ФП
  {определение_ФП}+
  'КОММЕНТАРИЙ' строка120
```

При определении функции принадлежности указывается ее имя, начальная точка шкалы, конечная точка шкалы, общее число определяемых точек для кусочно-линейного представления и список точек согласно синтаксису:

```
определение_ФП =
  имя_ФП
  начальная_абсцисса
  конечная_абсцисса
  количество_точек
  '={' список_точек '}'
```

Точки шкалы перечисляются через символ “;”, где через символ “|” указывается степень принадлежности точки нечеткому множеству, лежащая в отрезке [0; 1].

```
список_точек =
  {абсцисса '|' степень ';'} абсцисса '|' степень
абсцисса = числоПЗ
степень = число01
```

Пример словаря с одной словарной статьей для лингвистической переменной “Скорость”:

```
СЛОВАРЬ
“Скорость”
3
“Малая” 0 120 5 ={0|0.5; 10|1; 30|0.9; 80|0.3; 120|0}
“Средняя” 0 120 5 ={0|0; 10|0.3; 40|1; 80|0.8; 120|0}
“Большая” 0 120 5 ={0|0; 30|0.3; 50|0.5; 80|0.8; 120|1}
```

Приложение 2. Язык описания сценариев диалога

Далее описывается язык описания сценариев диалога (ЯОСД) инструментального комплекса АТ-ТЕХНОЛОГИЯ.

Алфавит

Приведем основные символы языка:

Идентификатор – текстовая строка, которая может содержать латинские и русские буквы, цифры и знак подчеркивания. Единственным ограничением на используемые идентификаторы является то, что они не могут совпадать с ключевыми словами ЯОСД (см. ниже).

Имя атрибута – строка символов, заключенная в маркеры "#". Использование подобной нотации вызвано тем, что интерпретатор ЯОСД не рассматривает имя атрибута как совокупность имени объекта, символ разделитель и, собственно имени атрибута (например, Человек.Имя или Человек.Пол). Непосредственно процесс означивания или получения значения того или иного атрибута возлагается на ядро экспертной системы. Поэтому, чтобы упростить процесс считывания данного символа из текста на ЯОСД, а, кроме того, обеспечить независимость ядра диалогового компонента от конкретной реализации остальных компонент ИЭС, для задания имени атрибута используется приведенная нотация.

Строка – строка символов, заключенная в кавычки.

Ключевое слово – специальный символ ЯОСД, несущий определенную семантическую нагрузку для интерпретатора. С помощью ключевых слов строятся "скелеты" всех используемых в ЯОСД синтаксических конструкций (в грамматике, наравне со знаками, выделены "жирным" шрифтом). В ЯОСД используются следующие ключевые слова:

about	**goto**	**stop**
all	**line**	**subscenario**
answer	**message**	**to**
break	**scenario**	**when**
end	**send**	

execute	**set**

Знак – специальный символ ЯОСД, несущий определенную семантическую нагрузку для интерпретатора, но, в отличие от ключевых слов, имеющий второстепенное значение для построения "скелетов" используемых в ЯОСД синтаксических конструкций. В ЯОСД используются следующие знаки:

':', '(', ')', '[', ']'

Синтаксис ЯОСД

Приведем синтаксис языка описания сценариев диалога в расширенной форме Бэкуса-Наура:

```
описание_диалога =
  {(сообщение | доп_сценарий)}
  глав_сценарий
  {(сообщение | доп_сценарий)}

сообщение =
  'message' имя_сообщения ['to' адресат]
  {'about' имя_атрибута}
  {стр_сообщения}
  'end'

адресат = имя_компонента | 'all'
стр_сообщения = 'line' выражение

гл_сценарий =
  'scenario' имя_сценария
  {оператор}
  'end'

доп_сценарий =
  'subscenario' имя_сценария
  {оператор}
  'end'

оператор =
  отправка_сообщения | вып_сценария |
  присваивание | усл_переход |
  безусл_переход | остановка |
  прерывание | метка

отправка_сообщения =
```

```
  'send'
  (имя_сообщения ['to' адресат] |
  выражение 'to' адресат)

безусл_переход = 'goto' имя_метки

усл_переход =
  'when' выражение 'goto' имя_метки

присваивание =
  'set' имя_аттрибута 'to' выражение

вып_сценария = 'execute' имя_сценария
остановка = 'stop'
прерывание = 'break'
метка = имя_метки ':'

выражение =
  значение | имя_атрибута |
  вызов_функции | ответ

значение = строка | текст
текст = '[' [строка { ',' строка}]']'
вызов_функции = имя_функции параметры
параметры =
  '(' [выражение { ',' выражение}] ')'

ответ = 'answer'

имя_сообщения - идентификатор;
имя_сценария - идентификатор;
имя_функции - идентификатор;
имя_компонента - идентификатор;
имя_метки - идентификатор;
имя_атрибута - строка символов, заключенная в маркеры
'#'.
строка - строка символов, заключенная в кавычки (в
случае, когда кавычку нужно поместить внутрь строки,
следует использовать символ цитирования '$'; в этом
случае вместо ' следует писать $');
```

Семантика ЯОСД

Опишем семантику ЯОСД посредством специфицирования принципов работы интерпретатора данного языка.

Вначале рассмотрим общую структуру описания некоторого диалога. Итак, согласно изложенному выше синтаксису, описание диалога представляет собой совокупность следующих блоков:

- описание сообщений;
- описание главного сценария;
- описание дополнительных сценариев (подсценариев или сценариев уточняющих поддиалогов).

Рассмотрим более подробно каждый из указанных блоков.

Описание сообщений

Итак, согласно устоявшимся представлениям, программная система, построенная и функционирующая в соответствии с принципами объектно-ориентированного подхода, представляет собой взаимосвязанную совокупность объектов, которые в процессе своей "жизни" обмениваются сообщениями. Поэтому, чтобы осуществить гармоничную интеграцию разрабатываемой подсистемы общения в общую архитектуру ИЭС, в ЯОСД были введены специальные конструкции, предназначенные для описания, отправки сообщений (а именно, команд и данных) и обработки полученных ответов.

В ряде случаев (далее будет ясно, в каких именно) может возникнуть необходимость в предварительном задании информационного наполнения и указании адресата для того или иного сообщения. Для этих целей предлагается описывать сообщения в соответствии со следующим синтаксисом:

```
сообщение = 'message' имя_сообщения ['to' адресат]
            {'about' имя_атрибута}
              {стр_сообщения}
            'end'
адресат = имя_компонента | 'all'
стр_сообщения = 'line' выражение
```

Поясним принципы использования описанного механизма на нескольких примерах.

Пример.

```
message ВыдатьПреамбулуПоПрименимости to Informer;
  line 'set Caption to $'Исследование на применимость
технологии СОЗ$'';
  line concat('output ', text(concat(
    'Уважаемый(ая) ',
    #knowleges.data.Отчет.ФИО#, '.'),
    '',
    'Система начинает консультацию по применимости',
    'технологии СОЗ для решения Вашей задачи.',
```

```
    '',
    'Будет проведено три исследования:',
    ' 1. Исследование на уместность разработки СОЗ.',
    ' 2. Исследование на оправданность разработки
СОЗ.',
    ' 3. Исследование на возможность разработки СОЗ.',
    '',
    'При проведении каждого исследования система ',
    ' будет задавать Вам вопросы, касающиеся Вашей ',
    ' проблемной области. Затем система обработает ',
    ' полученные ответы и выдаст заключение.'),
  'as Attention');
  line 'activate';
end;
```

В данном примере описывается сообщение под именем “ВыдатьПреамбулуПоПрименимости”, предназначенное для передачи его компоненту Informer. В теле сообщения описаны передаваемые строки сообщения в виде совокупности текстовых строк с префиксом ‘line’.

В следующем примере описывается сообщение с привязкой к некоторым атрибутам рабочей памяти “knowleges.data.Отчет.ФИО” и “knowleges.data.Отчет.Задача”:

```
message ЗарегистрироватьПользователя to Asker
about #knowleges.data.Отчет.ФИО#
about #knowleges.data.Отчет.Задача#;
  line 'set Caption to $'Регистрация пользователя$'';

  line concat('output ',
    text('Уважаемый пользователь!',
    '',
    'Чтобы система знала, как Вас зовут, введите,',
    'пожалуйста, Ваши фамилию, имя и отчество’,
    'в поле "ФИО".',
    '',
    'Также системе необходимо знать, какую ',
    'задачу Вы хотите решить, используя ',
    'технологию СОЗ. Поэтому введите ',
    'название Вашей задачи в поле "Задача".'),
    ' as Attention');

  line 'input $'ФИО   $' to #knowleges.data.Отчет.ФИО#
as String';
  line 'input $'Задача$' to
#knowleges.data.Отчет.Задача# as String';
  line 'activate';
end;
```


Необходимость в таком описании возникает в связи с тем, что в процессе решения задачи ИЭС может столкнуться с ситуацией, когда решателю не хватает данных для получения ответа. В этом случае он формирует запрос к диалоговому компоненту на означивание того или иного атрибута. Для означивания каждого конкретного атрибута диалоговому компоненту необходимо иметь информацию о том, как должен быть означен данный атрибут (посредством выдачи вопроса к пользователю, посредством организации запроса к базе данных либо посредством вычислений). Кроме того, диалоговый компонент должен обладать различного рода вспомогательной информацией, например для организации ввода информации пользователем необходимы текст вопроса и тип запрашиваемого значения и т.д. Эта проблема решается путем использования специальных директив, привязывающих некоторые сообщения к одному или нескольким атрибутам рабочей памяти и, кроме того, к компоненту, который будет осуществлять означивание указанных атрибутов. Таким образом, при получении диалоговым компонентом запроса от решателя на означивание того или иного атрибута рабочей памяти, требуемую информацию диалоговый компонент может получить из описания соответствующего сообщения. В данном примере при запросе со стороны решателя на означивание атрибутов с именами “knowleges.data.Отчет.ФИО” и “knowleges.data.Отчет.Задача” указанное сообщение будет передано компоненту “Asker”. Компонент “Asker”, в свою очередь, сформирует диалоговое окно, выдаст вопросы и предложит пользователю ввести информацию в виде строк символов. Затем компонент сформирует и отправит сообщение к ядру ИЭС на присваивание заданным атрибутам полученных значений. Аналогичным образом описываются сообщения, необходимые для получения информации из базы данных, либо любых других компонент ИЭС.

Описание главного сценария

Главный сценарий представляет собой последовательность шагов (операторов), описывающих диалог наиболее общим образом. Синтаксис описания главного сценария имеет следующий вид:

```
гл_сценарий = ‘scenario’ имя_сценария
                {оператор}
              ‘end’
```

Важно отметить, что в описании любого диалога всегда существует главный сценарий, причем всегда только один. Именно с интерпретации данного сценария и начинается работа ДК и ИЭС в целом. Приведем список операторов ЯОСД, которые можно использовать в теле главного сценария:

- отправка сообщения с заданным именем (если адресат не указан, то он должен быть задан в описании сообщения);
- отправка сообщения с заданным значением (с обязательным указанием адресата)
- выполнение дополнительного сценария с заданным именем;
- присваивание атрибуту рабочей памяти заданного значения;
- остановка работы системы;
- прерывание выполнения текущего сценария;
- осуществление безусловного перехода на метку;
- осуществление условного перехода на метку;
- метка (не выполняет никаких действий и поэтому могла бы быть приписана к следующему за ней оператору; однако в случае, когда метка находится в конце сценария и после нее нет ни одного оператора, приписать ее к какому-либо из перечисленных операторов не удастся; поэтому рассматривается как отдельный оператор, для общности).

Детальная спецификация каждого описанного оператора приведена ниже.

Описание дополнительных сценариев

Каждый дополнительный сценарий (сценарий уточняющего поддиалога) описывается в соответствии со следующим синтаксисом:

```
доп_сценарий =
  'subscenario' имя_сценария
  {оператор}
  'end'
```

Основное назначение дополнительных сценариев заключается в детализации (уточнении) отдельных шагов главного сценария. Такой прием призван улучшить структурированность текста описание диалога и, как следствие, повысить его читабельность. В теле дополнительных сценариев допустимо использовать те же

операторы, что и в теле главного сценария.

Далее более детально остановимся на тех или иных конструкциях языка.

Вычисление выражений

Итак, рассмотрим правила вычисления выражений. Приведем фрагмент описания синтаксиса ЯОД, содержащий данные о выражениях:

```
выражение =
  значение | имя_атрибута |
  вызов_функции | ответ
значение = строка | текст
текст = '[' [строка { ',' строка}]']'
вызов_функции = имя_функции параметры
параметры =
  '(' [выражение { ',' выражение}] ')'
ответ = 'answer'
```

Как видно из РБНФ-определения, выражение может быть либо значением, либо именем атрибута, либо вызовом некоторой функции, либо ответом на отправленное предварительно сообщение. Рассмотрим каждый из приведенных случаев.

В случае если выражение задается конкретным значением, результатом вычисления выражения будет данное значение.

Если выражением является имя некоторого атрибута, то результатом вычисления выражения будет значение указанного атрибута. Получение значения того или иного атрибута осуществляется посредством обращения к ядру ИЭС.

Если выражение задано вызовом некоторой функции, то результатом вычисления данного выражения будет значение этой функции с предварительно вычисленными параметрами (более подробное описание механизма вызова функций смотрите ниже).

Если выражение представлено ключевым словом **answer**, то результатом вычисления данного выражения будет текст ответа на последнее посланное сообщение к какому-либо компоненту системы.

Вызов функций

Опишем далее принципы, лежащие в основе интерпретации вызовов функций. Если обратиться к описанию синтаксиса ЯОСД,

то можно видеть следующее:

```
вызов_функции = имя_функции параметры
параметры = '(' [выражение { ',' выражение}] ')'
```

Вызову функции предшествует вычисление параметров ее вызова. После того, как все параметры вычислены, осуществляется вызов функции по ее имени с передачей ей вычисленных параметров. Именем функции может быть имя одной из базовых функций ЯОСД.

Оператор отправки сообщения

Как уже упоминалось выше, важнейшим элементов межкомпонентного взаимодействия внутри системы является отправка сообщений от одних компонент системы к другим. В связи с этим, одним из основных операторов ЯОСД является оператор оправки сообщения от ДК к другим компонентам системы. Предлагается две модификации данного оператора:

- оператор отправки сообщения с заданным именем;
- оператор отправки сообщения с заданным значением.

Приведем синтаксис рассматриваемого оператора в обеих его модификациях:

```
отправка_сообщения =
  'send'
  (имя_сообщения ['to' адресат] |
  выражение 'to' адресат)
```

В случае использования оператора отправки сообщения с заданным именем без указания адресата предполагается, что адресат уже определен при описании сообщения. В случае использования оператора отправки сообщения с заданным значением указание адресата является обязательным. Здесь важно отметить, что в случае использования оператора отправки сообщения с заданным именем с указанием адресата, оно будет послано тому объекту, имя которого указано в операторе, даже если изначально сообщение предназначалось другому объекту (что определялось посредством использования соответствующей директивы в описании сообщения).

Пример

```
send 'solve' to solver;
send Идентифицировать;
send Идентифицировать to Asker;
```



```
send 'stop' to all;
```

Оператор выполнения дополнительного сценария

В ЯОСД предусмотрен специальный оператор, с помощью которого осуществляется выполнение заданного дополнительного сценария. Синтаксис данного оператора имеет вид:

```
вып_сценария = 'execute' имя_сценария
```

При интерпретации данного оператора осуществляется переход к выполнению операторов, находящихся в теле поддиалога с заданным именем, после чего осуществляется возврат и обработка операторов, следующих за оператором выполнения поддиалога.

Пример

```
scenario ГлавныйСценарий;
  execute Приглашение;
  execute ВводИсходныхДанных;
  send 'solve' to solver;
  execute ВыдачаРешения;
end;
```

Оператор присваивания

Оператор присваивания имеет следующий вид:

```
присваивание = 'set' имя_атрибута 'to' выражение
```

При интерпретации данного оператора вначале вычисляется выражение, стоящее после ключевого слова **to**. Затем полученное значение присваивается атрибуту с именем, заданным после ключевого слова **set**.

Пример

```
set #Человек.Имя# to 'Ваня';
send concat('name = ', #Человек.Имя#) to storage;
send 'get sex' to storage;
set #Человек.Пол# to answer;
```

Оператор безусловного перехода

Рассмотрим оператор безусловного перехода. Синтаксис его написания имеет следующий вид:

```
безусл_переход = 'goto' имя_метки
```

При интерпретации данного оператора осуществляется по-

иск в рамках текущего диалога (поддиалога) метки с именем, заданным после ключевого слова **goto**, после чего осуществляется переход к интерпретации оператора, которому предшествует данная метка, затем к следующему за ним и т.д.

Оператор условного перехода

В соответствии с описанным выше синтаксисом, условный оператор имеет следующий вид:

```
усл_переход = 'when' выражение 'goto' имя_метки
```

При интерпретации условного оператора вначале вычисляется выражение, стоящее после ключевого слова **when** (в соответствии с правилами, описанными выше). Здесь следует отметить, что в ЯОСД не существует логического типа данных, точно так же, как не существует специального типа для представления чисел и т.д. Предполагается, что любое значение, которое может быть получено в результате вычислений, посредством обращения к ядру ИЭС, либо явно задано, представляется строкой символов. В случае если ядро ИЭС поддерживает какие-либо другие типы данных, при передаче информации в ДП, она должна быть приведена к строковому типу. Легко видеть, что это не снижает универсальности диалогового компонента, т.к. любой тип данных может быть посредством применения некоторой процедуры перекодирования (например, преобразование числа в строку или преобразование массива в список, заданный строкой и т.п.) преобразован в строковый тип. С другой стороны, такой подход не исключает наличие в библиотеках диалогового компонента функций, служащих для выполнения операций над числами, массивами, векторами, множествами и т.д. В этом случае такие функции сначала преобразуют переданные им аргументы к тому типу, который им необходим, обрабатывают полученные значения, а результат преобразуют к строковому типу.

Вернемся, однако, к описанию оператора условного перехода. Итак, в случае если вычисленное выражение имеет значение ‘**T**’, то выполняются оператор перехода на указанную метку, заданную после ключевого слова **goto**. В случае если значение выражения равно ‘**F**’, интерпретация оператора условного перехода заканчивается и осуществляется переход к следующему за ним оператору.

Оператор остановки системы

Для управления функционирования системы, частью которой является диалоговый компонент, был введен оператор остановки. Синтаксис данного оператора имеет вид:

```
остановка = 'stop'
```

При выполнении данного оператора производится остановка системы, т.е. завершается выполнение всех ее компонент.

Оператор прерывания выполнения текущего сценария

Для удобства написания сценариев диалога, а также с целью предоставления другим компонентам системы больших возможностей по управлению интерпретацией сценариев диалога посредством отправки диалоговому компоненту соответствующих сообщений был введен т.н. оператор прерывания. Рассмотрим его синтаксис:

```
прерывание = 'break'
```

При выполнении данного оператора интерпретация текущего сценария прекращается. В случае использования данного оператора в теле главного сценария работа диалогового компонента завершается.