СИСТЕМЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

УДК 004.891, 004.82

О.Н. Куделин, Е.С. Кукин, С.В. Липатова, А.А. Смагин

СРЕДСТВО ПОСТРОЕНИЯ ЭКСПЕРТНЫХ СИСТЕМ

Куделин Олег Николаевич, *кандидат физико-математических наук, окончил физико-технический факультет Ульяновского государственного университета. Ведущий инженер-программист ФНПЦ ОАО «НПО «Марс». Специализируется в области создания экспертных систем. [e-mail: mars@mv.ru].*

Кукин Евгений Серафимович, *кандидат технических наук, доцент, окончил физический факультет Воронежского государственного университета. Заместитель главного конструктора, начальник отделения ФНПЦ ОАО «НПО «Марс». Имеет статьи в области разработки программного обеспечения для АСУ. [e-mail: mars@mv.ru].*

Липатова Светлана Валерьевна, *кандидат технических наук, окончила факультет «Информационные телекоммуникационные технологии» УлГУ. Старший преподаватель кафедры «Телекоммуникационные технологии и сети» УлГУ. Специализируется в области создания экспертных систем.[e-mail: dassegel@mv.ru].*

Смагин Алексей Аркадьевич, *доктор технических наук, окончил радиотехнический факультет Ульяновского политехнического института. Заведующий кафедрой «Телекоммуникационные технологии и сети» УлГУ. Имеет свыше 100 статей, изобретений, монографий в области разработки информационных систем различного значения. [e-mail: smaginaa1@mail.ru].*

Аннотация

В статье описывается унифицированная система, позволяющая создавать экспертную систему (ЭС) без участия программистов непосредственно экспертами, специализирующимися в какой-либо узкой области знаний.

Ключевые слова: проектирование АСУ, экспертная система, искусственный интеллект.

Abstract

The article describes an unified system which allows creating an expert system without programmers, directly by experts who specialize in a narrow area of knowledge.

Key words: design of computer-aided control system, expert systems, artificial intelligence.

Введение

Необходимость использования опыта специалистов в различных информационных процессах делает востребованными технологии экспертных систем. Их можно применять в различных предметных областях для решения задач поддержки принятия решений, управления, обучения, мониторинга и т. д.

Структура различных ЭС, решающих однотипные задачи в разных предметных областях, одинакова. Именно поэтому часто для их создания (исследовательских и демонстрационных стадий существования) используют оболочки ЭС, которые поддерживают различные модели представления знаний, свои языки их описания и предоставляют слабо развитый интерфейс взаимодействия с пользователем. При создании ЭС на базе какой-либо оболочки необходимо привлекать экспертов, когнитолога, программиста, владеющего языком оболочки ЭС, и программиста, разрабатывающего интерфейс создаваемой ЭС.

Для исключения из коллектива разработчиков программистов (для сокращения затрат) предлагается соз-

дание надстроек над оболочкой – дополнительных программных средств, которые позволяют описывать базы знаний (БЗ) без знания языка оболочки и предоставляют пользователю ЭС удобный интерфейс. При этом надстройки не требуют дополнительных ресурсов и не изменяют логику работы оболочки. Оболочка ЭС совместно с надстройками образуют средство построения ЭС (СРЭС), которое обеспечит поддержку этапов жизненного цикла ЭС от формализации до тестирования при разработке ЭС в произвольной предметной области для круга определенных задач (от выбора задач зависит структура БЗ).

Архитектура средства построения экспертных систем

Любая типовая ЭС должна содержать обязательные блоки: БЗ, решатель, подсистему объяснений, интерфейс пользователя и интерфейс когнитолога. Исходя из этого, структуру СРЭС можно представить в следующем виде (рис. 1):

Рис. 1. Структура СРЭС

- интерфейс когнитолога, генерирующий БЗ;
- ядро системы, состоящее из стандартной оболочки, используемой в качестве машины вывода (решателя), БЗ, входных и выходных данных, представленных в виде текстовых файлов, содержащих код на языке оболочки и конечные рекомендации;
- интерфейс пользователя, генерирующий входные данные в виде фактов, объектов, констант на языке оболочки, отображающий состояние внешней среды системы и представляющий пользователю конечные рекомендации с пояснениями.

Эксперт в выбранной предметной области с помощью ПО «Интерфейс эксперта» может выполнять следующие функции:

- формирование БЗ;

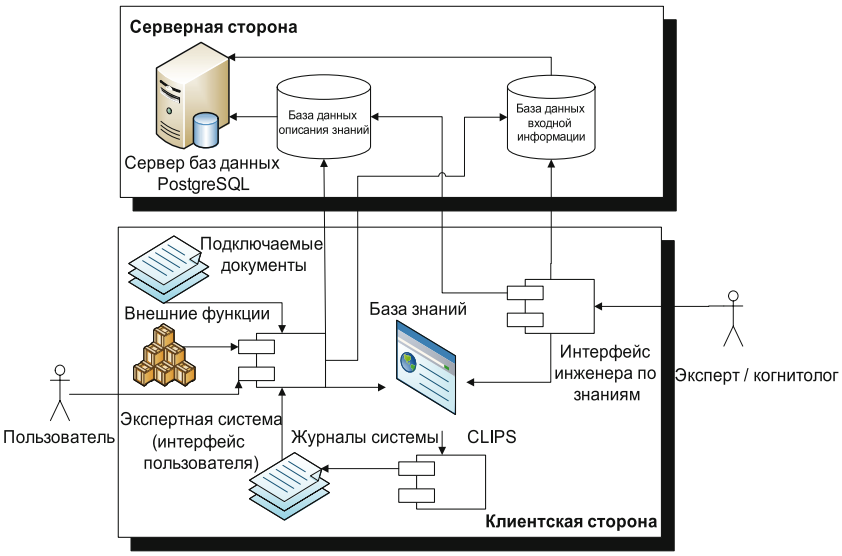

Рис. 2. СРЭС на базе клиент-серверной архитектуры

- описание предметной области через термины и задачи;
- создание объяснений принимаемых решений в рамках предметной области.

Пользователь, применяющий ПО «Интерфейс пользователя», может:

- отправлять запрос на решение поставленной задачи;
- уточнять описание конкретной ситуации (сбор дополнительной информации);
- получать рекомендации по решению поставленной в запросе задачи;
- получать объяснения по поводу представляемых рекомендаций.

Возможны различные варианты реализации СРЭС. Рассмотрим вариант на базе архитектуры «клиент-сервер» (рис. 2). Предлагаемое решение представлено в виде двух отдельных программных модулей: интерфейса эксперта и интерфейса пользователя – на основе оболочки ЭС CLIPS (кроссплатформенная с открытыми кодами система на базе продукционной модели представления знаний) с использованием базы данных PostgreSQL для хранения описания предметной области.

На серверной стороне располагаются базы данных описания предметной области и входной информации. Остальные компоненты системы находятся на клиентской стороне. В зависимости от типа пользователя возможны различные варианты установки системы. Если пользователь – когнитолог или эксперт, который будет создавать или тестировать экспертную систему, то необходимо размещение всех компонентов. Если пользователь – оператор экспертной системы, не обладающий правами на модификацию системы, то на клиентской стороне интерфейс инженера по знаниям не устанавливается.

СРЭС позволяет экспертам совместно с конгитологом создавать описания предметных областей, которые сохраняются в базах данных. На их основе средствами СРЭС генерируются БЗ на языке COOL, понятном для оболочки ЭС CLIPS. Интерфейс пользователя ЭС преобразует входные

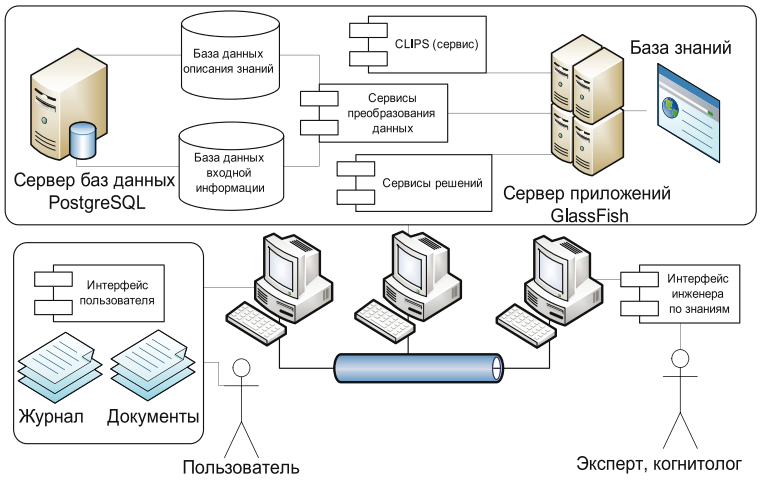

Рис. 3. СРЭС на базе SOA

данные (факты и значения атрибутов классов), поступающие из разнородных источников (от пользователя, из базы данных, от внешних процедур), в формат CLIPS и, наоборот, выходные данные (решений, объяснений, документов, внешних функций), поступающие от оболочки ЭС, в формат, понятный и удобный пользователю.

Реализация предложенной структуры возможна и на базе сервис-ориентированной архитектуры (SOA, рис. 3).

Такая реализация требует переработки внешних функций, функций интерфейса по преобразованию форматов и CLIPS в сервисы (так как исходный код открыт и реализован на языке программирования Си, то доработка не вызывает затруднений). Интерфейс пользователя становится потребителем этих сервисов, которые располагаются на сервере приложений. Интерфейс когнитолога претерпевает минимальные изменения, так как он работает только с базами данных.

Выбор между этими реализациями СРЭС зависит от того, на базе какой архитектуры должна функционировать разрабатываемая ЭС, будет ли она интегрироваться с другими системами и с какими именно.

Описание работы программного комплекса

Логика работы пользовательской части СРЭС проста – пользователь, следуя указаниям системы, ставит вопрос (выбирает задачу, рис. 4) и вводит данные (рис. 5), затем получает ответ – решение (рис. 6), объяснение и документы. При этом он может редактировать данные, решать задачи заново с измененными входными данными, запускать внешние функции или сервисы (зависит от архитектуры), обеспечивающие выполнение предложенного системой решения.

Для работы с интерфейсом когнитолога, т. е. для создания БЗ, необходимы минимальные знания о продукционной модели представления знаний. Работу по созданию БЗ можно разделить на следующие этапы:

- непосредственное описание предметной области через термины, которые могут быть в системе нескольких типов (класс, атрибут/слот, значение, константа и факт);
- описание задач предметной области (совокупность правил, связанных между собой, приводящих к решению задачи и содержащих объяснения этих решений);

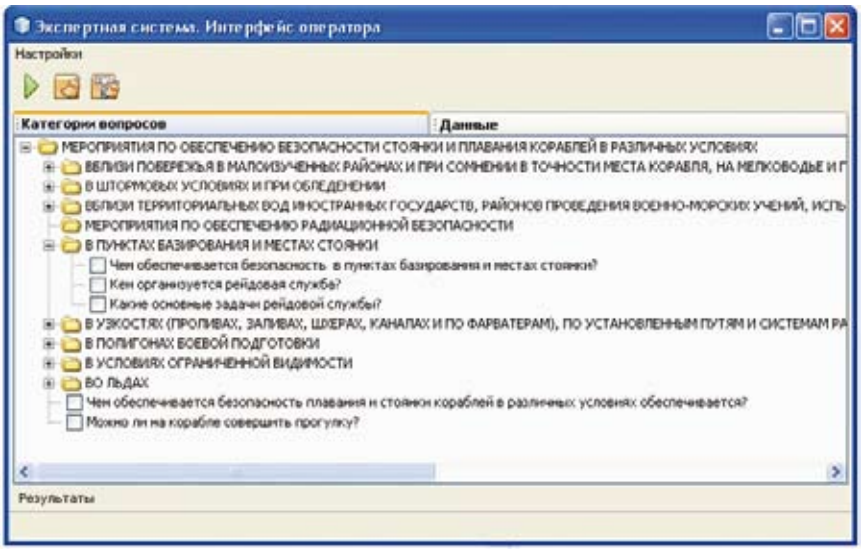

Рис. 4. Выбор вопроса в экспертной системе

- проверка правильности задания правил базы знаний (тестирование).

Задача предметной области в системе описывается через набор следующих элементов:

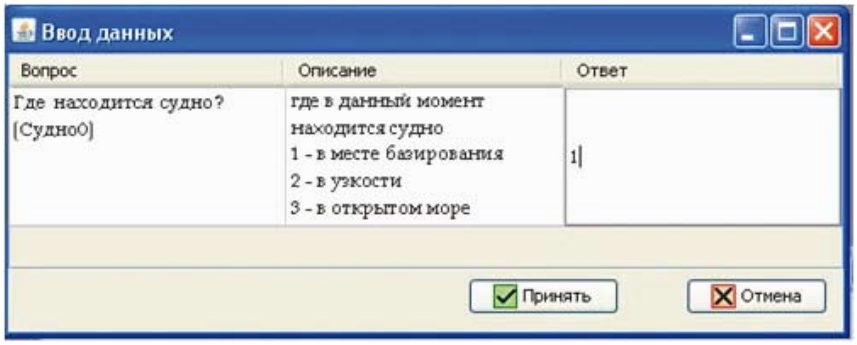

Рис. 5. Ввод данных в экспертную систему пользователем

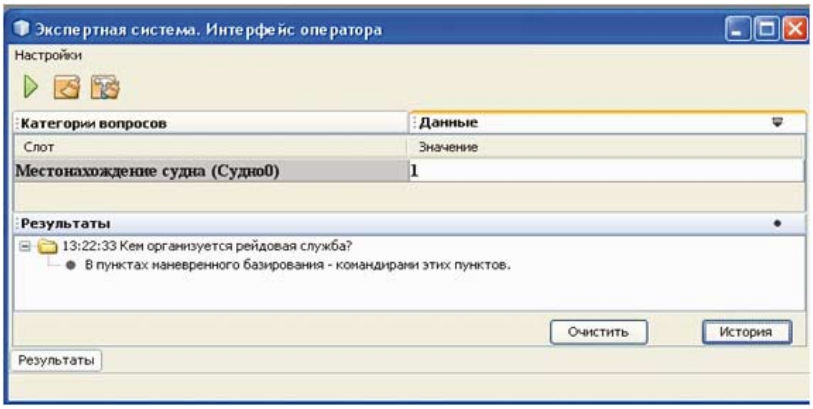

Рис. 6. Вывод результирующего решения (в краткой форме)

- название задачи – идентификатор задачи в интерфейсе когнитолога (параметр не доступен пользователю ЭС);

- вопрос, на который отвечает задача, – идентификатор-запрос, который предоставляется пользователю ЭС;

- описание входных данных, необходимых для решения задачи, с указанием источника получения этих данных;

- набор правил, позволяющих получить решение (правила могут быть взаимосвязаны (цепочка правил формируется с помощью фактов) или независимы). Каждое правило может содержать: имя, условие выполнения (условие накладывается на входные данные и на генерируемые в процессе выполнения факты), решение, объяснение, прикрепленный документ, прикрепленную внешнюю функцию, перечень генерируемых и удаляемых фактов, приоритет, степень уверенности в правильности ответа (первые 4 элемента обязательные);

- набор подзадач, если задача составная.

Для удобства и наглядности представления связей между правилами, относящимися к решению одной задачи, они представляются в виде дерева решений (рис. 7).

Взаимосвязь и порядок выполнения правил обеспечиваются, во-первых, приоритетами правил (каждому уровню соответствует свой приоритет, самый верхний уровень имеет самый высокий приоритет), во-вторых, генерацией текущих фактов. При выполнении каждого правила генерируется факт, связанный с этим правилом, который сохраняется в системе до конца решения поставленной задачи. Наличие в системе фактов, соответствующих вышестоящим правилам в цепочке от текущего правила до вершины дерева решений, является обязательным, генерируемым автоматически системой дополнительным условием в правиле. Если между вершиной дерева и текущим правилом есть параллельные участки, то при формировании дополнительного условия они объединяются логическим оператором «ИЛИ».

Дополнительное условие наличия фактов добавляется к условию правила и связывается логическим оператором «И». Также для избегания процессов зацикливания в условие каждого правила автоматически добавляется проверка отсутствия в системе факта, генерируемого текущим правилом.

Результатом выполнения правила может быть рекомендация по решению задачи, вызов процедуры и обязательно факт. На каждой ветви должно быть хотя бы одно правило, генерирующее рекомендацию. Если их несколько, то итоговая рекомендация является суммой всех рекомендаций правил на текущей ветви дерева решений (рекомендации соединяются, начиная с верхнего уровня).

Основное условие правила задается отдельно и представляется пользователю в виде графа, узлами которого являются объекты БЗ (слоты классов; константы; значения, вводимые пользователем; факты), а дугами – логические операторы (рис. 8). Созданное условие может использоваться в нескольких правилах, условие может входить в другое условие в виде подусловия.

Для описания терминов предметной области на языке COOL используются операторы *defclass, defglobal* (задают структуру классов, определяют слоты и глобальные переменные), для задач – *defrule*.

(defclass Название класса; объявление класса
(is-aUSER); объявление класса наследника класса USER
(roleconcrete); роль класса - конкретный
(slot Название_слота (type Tun))
)
*(defglobal ? *Название_глобальной_переменной* = Значение); объявление глобальной переменной и задание ее значения*

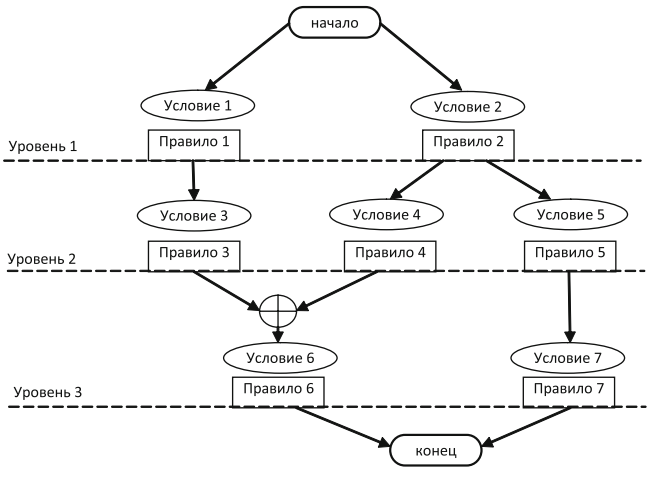

Рис. 7. Пример схемы дерева решений

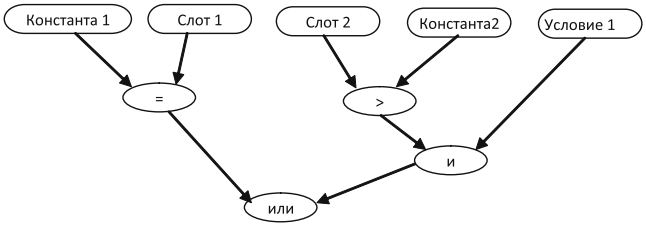

Рис. 8. Пример схемы графа условия

Структура правила, позволяющая представить в ЭС вышеописанные механизмы, может быть следующей:

(defrule Название_правила_P

(declare (salience Приоритет)); установка приоритета правила согласно уровню в дереве решений

?o_i<- (object (Имя_ j_слота_i_объекта ?oi_ j)); присваивание локальным переменным значений слотов объекта или объектов, используемых в условии

(факт_n); наличие в системе факта или фактов, генерируемых правилами, находящимися выше по ветви дерева вывода

(not факт_P); проверка отсутствия в системе факта, связанного с текущим правилом

(test(условие))

=>

(send ?o_kput-Имя_слота_n_значение); изменение при необходимости значений слотов объектов

(printout Файловая_переменная

"<id_задачи >

<id_задачи>Число</id_задачи>

<вопрос> Строка </вопрос>

<описание_задачи>Число</описание_задачи>

<решение>Текст</решение>

<объяснение>Текст</объяснение>

<степень_уверенности>Число</степень_уверенности>

<документ>Полный путь к документу</документ>

<функция>

<название_функции> Имя функции </название_функции>

<описание_функции> Текст </описание_функции>

<библиотека> Полный путь к библиотеке </библиотека>

<входные_параметры>

<параметр>

<id_параметра> Число </id_параметра>

<тип_параметра> Тип </тип_параметра>

<значение_параметра> Число или текст </значение_параметра>

…

</параметр>

</входные_параметры>

<выходной_параметр>

<тип_параметра> Тип </тип_параметра>

<значение_параметра> Число или текст </значение_параметра>

</выходной_параметр>

</функция>

</id_задачи >

")

(assert (факт_P)); добавление в систему факта, связанного с этим правилом

)

Для представления текущей ситуации (входных данных для задачи) используются операторы *send* и *make-instance*, которые позволяют определить объекты и задать значения их слотов, отражающих данные решаемой задачи:

(make-instance Название_объекта of Название_класса); задание объекта определенного класса

(send [Название_объекта] put-Название_слота_объекта Значение); задание значения слота объекта.

Часть описания, избыточного для оболочки ЭС, но необходимого для интерфейса пользователя, хранится в базе данных и не кодируется в БЗ. Интерфейс пользователя напрямую обращается к базе данных.

Пользователю, эксперту и конгитологу нет необходимости разбираться с языком оболочки ЭС – генерация кода БЗ на основе описания, сохраненного в базе данных, производится автоматически интерфейсом когнитолога.

При использовании описанной структуры правой части правила результатом работы оболочки ЭС на основе сгенерированных БЗ и входных данных является файл в формате XML, в котором в соответствующих тегах отображены название задачи, решение задачи, объяснение решения, имя прикрепленного документа, название и необходимая информация для вызова внешней функции или сервиса (с зависимости от архитектуры), степень уверенности. Данные этого файла представляются пользователю в удобном виде.

Использование предлагаемой системы позволяет привлекать экспертов к непосредственному созданию ЭС, исключить из группы разработчиков программистов, значительно сократить время построения демонстрационных или исследовательских ЭС. СРЭС можно использовать для построения систем поддержки принятия решений, информационно-справочных систем, систем управления в различных предметных областях. В зависимости от реализации части «Интерфейс пользователя» и выбора архитектуры создаваемые системы могут работать в режиме реального времени.

Заключение

Разработан и протестирован макет предлагаемой СРЭС на базе архитектуры «клиент-сервер» с ограниченным набором пользовательских функций. Макет с расширенными функциональными возможностями на базе SOA находится в процессе разработки. Использование предлагаемой системы позволяет привлекать экспертов к непосредственному созданию экспертных систем, исключив из группы разработчиков программистов, благодаря чему значительно сократится время построения демонстрационных или исследовательских экспертных систем.

СПИСОК ЛИТЕРАТУРЫ

1. Джарратано Д., Райлт Г. Экспертные системы: принципы разработки и программирование. – М.: ИД Вильямс, 2007. – 1152 с.

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

3. Компас в мире сервис-ориентированной архитектуры (SOA): ценность для бизнеса, планирование и план развития предприятия / Н. Биберштейн [и др].– М.: Кудиц-пресс, 2007. – 256 с.

4. Ньюкомер Э. Веб-сервисы: XML, WSDL, SOAP и UDDI. – СПб.: Питер, 2003. – 256 с.

5. Хабибуллин И.Ш. Разработка web-служб средствами Java. – СПб.: БХВ-Петербург, 2003. – 409 с.