Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»

Кафедра математической
кибернетики и компьютерных наук

РАЗРАБОТКА ЭКСПЕРТНОЙ СИСТЕМЫ «ВЫБОР ЯЗЫКА ПРОГРАММИРОВАНИЯ»

АВТОРЕФЕРАТ ДИПЛОМНОЙ РАБОТЫ

студентки 6 курса 611 группы
специальности 010501 – Прикладная математика и информатика
факультета компьютерных наук и информационных технологий
Цветковой Елены Александровны

Научный руководитель
к. ф.-м. н., доцент _____ А.С. Иванов

Заведующий кафедрой
к.ф.-м.н. _____ С. В. Миронов

Саратов 2016

ВВЕДЕНИЕ. Искусственный интеллект (ИИ) – это совокупность научных дисциплин, изучающих методы решения задач интеллектуального (творческого) характера с использованием ЭВМ. Область ИИ имеет более чем сорокалетнюю историю развития. Начиная с истоков, в ней рассматривался ряд весьма сложных задач, которые, как и другие, и до сих пор являются предметом исследований: машинный перевод (автоматический перевод с одного естественного языка на другой), автоматические доказательства теорем, распознавание изображений и анализ сцен, алгоритмы, планирование действий роботов и стратегии игр.

В начале восьмидесятых годов в исследованиях по искусственному интеллекту создалось отдельное направление, которое приобрело название "экспертные системы" (ЭС). Они ориентированы на решение широкого круга задач в неформализованных областях, то есть на приложения, которые до недавних времен считались малодоступными для вычислительной техники. Экспертные системы позволяют специалистам, которые не имеют навыков программирования, создавать практически значимые приложения, что резко расширяет сферу использования вычислительной техники. Такие системы при решении практических задач позволяют получать результаты, можно сравнить, а иногда и которые превосходят те, которые может получить эксперт-человек. [1]

Существует класс программ, которые называются оболочкой экспертной системы, созданный с целью позволить непрограммистам воспользоваться результатами работы программистов, решавших подобные проблемы. Таким образом под оболочками (shells) понимают "пустые" версии существующих экспертных систем, т.е. готовые экспертные системы без базы знаний. Примером такой оболочки можно привести программу EMYCIN, которая позволяет использовать архитектуру системы MYCIN в приложении к другим областям медицины (программа MYCIN была направленна только на заболевания крови). На основе EMYCIN были разработаны экспертные системы как для медицины (например, система PUFF для диагностики

легочных заболеваний), так и для других областей знаний, например программа структурного анализа SACON.

Целью дипломной работы является разработка и создание экспертной системы с продукционной базой знаний по выбору языка программирования. При планировании программного проекта имеется огромный выбор языков программирования, в лабиринтах которых легко заблудиться. Выбор языка зависит от многих факторов. Если это личный проект или хобби, можно выбрать знакомый язык. Если выбор зависит от имеющихся ресурсов, результат может быть весьма неочевидным. Можно также потратить много времени на разработку повторно используемых компонентов. В связи с этим была разработана программа, которая способна помочь в выборе наиболее подходящего языка.

В дипломной работе рассматриваются языки программирования по основным значимым параметрам, на основании чего разрабатывается база знаний с продукционной моделью и создается оболочка экспертной системы, которая поддерживает данную модель.

Работа состоит из введения, трех глав, заключения, списка использованных источников и трех приложений. Содержит 1 таблицу, 4 рисунков и 12 источников. Во введении кратко рассмотрено такое направление искусственного интеллекта, как экспертная система, описана актуальность работы, сформулирована цель. В первой главе приведена теоретические основы экспертных систем, описана их структура и этапы разработки. Во второй главе рассмотрены средства построения экспертных систем с примерами коммерческих оболочек. В третьей главе описан ход разработки базы знаний и оболочки экспертной системы, приведен результата тестирования программы. В заключении сформулированы результаты работы. В приложениях приведены набор правил предметной области, листинг базы знаний и полный код программы.

1 Экспертные системы. Структура экспертной системы. Экспертные системы – это направление исследований в области искусственного интеллекта по созданию вычислительных систем, умеющих принимать решения, схожие с решениями экспертов в заданной предметной области.

При разработке экспертной системы принято разделять ее на три основных модуля:

- База знаний;
- Машина логического вывода;
- Интерфейс с пользователем.

Принято считать машину вывода и интерфейс одним большим модулем, обычно называемым оболочкой экспертной системы или просто оболочкой.

Этапы разработки экспертной системы. В ходе работ по созданию ЭС сложилась определенная технология их разработки [5], включающая шесть последующих шагов: идентификацию, концептуализацию, формализацию, выполнение, тестирование, опытную эксплуатацию.

Процесс разработки ЭС не сводится к строгой последовательности рассмотренных выше шагов. В ходе создания ЭС приходится неоднократно возвращаться на более ранние этапы и пересматривать принятые там решения. [1]

2 Средства построения экспертных систем. В настоящее время существует большое количество средств для построения экспертных систем.

Из средств, которые в настоящее время нашли применение, можно создать классификацию:

1) Символьные языки, ориентированные на создание экспертных систем и систем искусственного интеллекта (LISP, SMALLTALK).

2) Языки инженерных знаний (языки высокого уровня, направленные на построение экспертных систем: PROLOG, OPS-5).

3) Системы автоматической разработки экспертных систем, направленные на знания: ART, TIMM.

4) Оболочки экспертных систем: EMYCIN, ЭКСПЕРТ.

3 Разработка экспертной системы. Предметная область и ее формализация. Описание предметной области

Предметной областью было выбрано определение необходимого языки программирования для написания программ.

Из всех существующих языков программирования как более распространённые, известные, со сложившимся обширным сообществом и большим количеством библиотек для данной работы были выбраны:

- Java;
- C;
- C#;
- C++;
- Delphi;
- Haskell;
- Erlang;
- Common Lisp;
- Perl;
- PHP;
- Ruby;
- Python.

В качестве критериев, важных при выборе языка программирования, для создаваемой базы знаний экспертной системы следующие характеристики были выделены:

- парадигма языка программирования;
- вид типизации;
- стандартизация;
- управление памятью;
- переносимость кода;
- скорость разработки;
- скорость исполнения. [8]

Сравнительные характеристики выбранных языков программирования представлены в таблице 1. [8][9][10]

Таблица 1 – Сравнительные характеристики языков программирования

Язык программирования	**Парадигма**	**Типизация**	**Управление памятью**
Java	Императивный, ООП	Статическая	Сборка мусора
C	Императивный	Статическая	Ручное управление
C++	Императивный, ООП	Статическая	Ручное управление
C#	Императивный, ООП	Статическая	Сборка мусора, ручное управление
Delphi	Императивный, ООП	Статическая	Ручное управление
Erlang	Функциональный, рефлексивный, декларативный	Динамическая	Сборка мусора
Haskell	Функциональный, декларативный	Статическая	Сборка мусора
Common Lisp	Императивный, ООП, функциональный, рефлексивный, декларативный	Динамическая	Сборка мусора
Perl	Императивный, ООП, функциональный	Динамическая	Сборка мусора
Ruby	Императивный, ООП, рефлексивный, декларативный	Динамическая	Сборка мусора
PHP	Императивный, ООП, рефлексивный, декларативный	Динамическая	Сборка мусора
Python	Императивный, ООП, рефлексивный, декларативный	Динамическая	Сборка мусора

Продолжение таблицы 1

Язык программирования	**Стандартизация**	**Переносимость кода**	**Скорость разработки ПО**	**Скорость исполнения программы**
Java	Нет	Виртуальной машиной	Быстрая	Средняя
C	ISO, ANSI	Перекомпиляцией	Медленная	Быстрая
C++	ISO	Перекомпиляцией	Медленная	Быстрая
C#	ISO, ECMA	Перекомпиляцией	Быстрая	Средняя
Delphi	Нет	Перекомпиляцией	Медленная	Быстрая

Erlang	Нет	Виртуальной машиной	Быстрая	Средняя
Haskell	Нет	Виртуальной машиной	Быстрая	Средняя
Common Lisp	ANSI	Виртуальной машиной	Средняя	Быстрая
Perl	Нет	Виртуальной машиной	Средняя	Средняя
Ruby	Нет	Виртуальной машиной	Быстрая	Медленная
PHP	Нет	Виртуальной машиной	Средняя	Медленная
Python	Нет	Виртуальной машиной	Быстрая	Медленная

Обоснование механизма вывода решения. Выбор языка программирования осуществляется на основе следующих критериев:

- парадигма языка программирования;
- вид типизация;
- стандартизация;
- управление памятью;
- переносимость кода;
- скорость разработки;
- скорость исполнения.

Формализация базы знаний. На этапе формализации базы знаний осуществляется выбор метода представления знаний. В рамках выбранного формализма осуществляется проектирование логической структуры базы знаний.

В продукционной модели основной единицей знаний служит правило в виде: "если <условие>, то <заключение>", с помощью которого могут быть выражены причинно-следственные, пространственно-временные, функционально-поведенческие (ситуация – действие) отношения объектов. Правилами могут быть описаны и сами объекты: "объект – свойство" или "набор свойств – объект", хотя чаще всего описания объектов фигурируют только в

качестве переменных ("атрибут – значение") внутри правил. Продукционная модель в основном предназначена для описания последовательности различных ситуаций или действий и в меньшей степени для структурированного описания объектов.

Продукционная модель предполагает более гибкую организацию работы механизма вывода по сравнению с логической моделью. Так, в зависимости от направления вывода возможна как прямая аргументация, управляемая данными (от данных к цели), так и обратная, управляемая целями (от целей к данным). Прямой вывод используется в продукционных моделях при решении, например, задач интерпретации, когда по исходным данным нужно определить сущность некоторой ситуации или в задачах прогнозирования, когда из описания некоторой ситуации требуется вывести все следствия. Обратный вывод применяется, когда нужно проверить определенную гипотезу или небольшое множество гипотез на соответствие фактам, например, в задачах диагностики. [3]

Реализация экспертной системы. В качестве языка программирования был выбран C#, как довольно современный язык, который обладает рядом особенностей и преимуществ перед другими языками. [11].

База знаний для разработанной экспертной системы хранится в одном отдельном файле с форматом .xml. Число условий в одном правиле не ограничено. Вывод одного правила может быть условием для другого.

Чтобы в алгоритме вывода можно было оперировать фактами, значениями фактов, учитывать их связь в определенном правиле и делать выводы, соответствующие данному набору фактов, база знаний экспертной системы представляется в виде определенных структур: массив вопросов <*Questions*>, массив переменных <*Objects*>, список целей <*Targets*> и набор правил <*Rules*>.

Каждое правило имеет следующую структуру:

<Rule id="25">

<Condition name="Парадигма языка" value="ООП" />

<Condition name="Типизация" value="Динамическая" />

<Consequence name="Язык программирования" value="Common Lisp" />

<Reason text="по двадцать пятому правилу" />

</Rule>

Здесь Rule id – номер правила,

Condition name – это факты i-того правила, value1,2 – значение факта.

Consequence name - название вывода i-того правила, value3 – содержание или значение вывода.

Reason text - пояснение объясняющее правило.

Листинг базы знаний представлен в приложении Б.

Алгоритм формирования вывода. Для формирования вывода был применен алгоритм с прямой цепочкой рассуждений и поиском в глубину.

Машина логического вывода (интерпретатор правил) выполняет две функции: во-первых, просмотр правил из базы знаний, во-вторых, применения правил.

Этот механизм управляет процессом консультации, сохраняя для пользователя информацию о полученных заключениях, и запрашивает у него информацию.

Механизм вывода представляет собой программу и включает в себя два компонента: первый реализует собственно вывод, другой управляет этим процессом.

Действие компонента вывода основано на применении правила, называемого modus ponens: «если известно, что истинно утверждение А и существует правило вида «ЕСЛИ А, ТО В», тогда утверждение В также истинно».

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

Управляющий компонент определяет порядок применения правил и выполняет четыре функции:

- сопоставление – образец правила сопоставляется с имеющимися фактами;
- выбор – если в конкретной ситуации может быть применено сразу несколько правил, то из них выбирается одно;
- срабатывание – если образец правила при сопоставлении совпал с какими-либо фактами из рабочей памяти, то правило срабатывает;
- действие – рабочая память подвергается изменению путём добавления в неё заключения сработавшего правила

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

В одном цикле может сработать только одно правило. Если несколько правил успешно сопоставлены с фактами, то такая ситуация называется конфликтной. Интерпретатор выполняет разрешение конфликтов используя стратегию глубины. Это воплощение стратегии новизны данных рассматриваемое по отношению к правилам, имеющим одинаковый класс выпуклости. Правила, выбранные в список заявок на основании данных, которые были включены в рабочую память сравнительно недавно, располагаются в этом списке раньше правил, при выборе которых использованы более старые данные. Таким образом, предпочтение отдается принципу поиска в глубину в пространстве состояний проблемы. [12]

При «поиске в глубину» происходит активация последнего правила, добавленного в конфликтное множество. Эта стратегия наиболее часто применяется в экспертных системах продукционного типа, т.к. она соответствует процессу рассуждений эксперта при решениии той или иной задачи. Поведение системы для пользователя выглядит довольно логичным: он видит, что система стремится свести каждую цель к подцели.

ЗАКЛЮЧЕНИЕ. Разработанная в ходе работы экспертная система является актуальной на сегодняшний день, так как:

- Формирует выводы, основываясь на знаниях которые хранятся отдельно от программного кода управляющего процессом вывода.
- Способна реконструировать методику решения задачи экспертом в соответствующей предметной области.
- Обладает способностью объяснять предлагаемое решение.
- Обеспечивает «дружественный», интуитивно понятный интерфейс.
- Имеет возможность изменения предметной области путем изменения только базы знаний без внесения поправок в код программы.
- Есть возможность писать все условия, значения и вопросы на русском языке.

Созданная в данной экспертной системе база знаний является полной и нацелена на помощь при принятии решений в выборе языка программирования для написания программ.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Статические и динамические экспертные системы: учеб. пособие / Э.В. Попов [и др.] М.: Финансы и статистика, 1996. 320 с.

2 Оболочки экспертных систем [Электронный ресурс]: сайт. URL: http://www.aiportal.ru/articles/expert-systems/shells.html (дата обращения 23.01.2016). Загл. с экрана. Яз. рус.

3 Долин, Г. Что такое ЭС / Г. Долин. М.: Компьютер пресс, 1992

4 Искусственный интеллект [Электронный ресурс]: сайт. URL: http://www.aiportal.ru (дата обращения 23.01.2016). Загл. с экрана. Яз. рус.

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

6 Рязанов, М.А. Анализ существующих средств разработки экспертных систем [Электронный ресурс]: текст научной статьи журнала «Известия Алтайского Государственного Университета». URL:

http://cyberleninka.ru/article/n/analiz-suschestvuyuschih-sredstv-razrabotki-ekspertnyh-sistem (дата обращения 23.01.2016). Загл. с экрана. Яз. рус.

7 Оболочки для создания экспертных систем. Экспертные системы. Объектно-событийное программирование [Электронный ресурс]: сайт. URL: http://bourabai.ru/alg/expert22.htm (дата обращения 23.01.2016). Загл. с экрана. Яз. рус.

8 Голицына, О.Л. Языки программирования. Учебное пособие. / О.Л. Голицына, Т.Л. Партыка, И.И. Попов. М.: Форум, 2008. 251 с.

9 Производительность языков программирования [Электронный ресурс]: сайт. URL: http://www.ibm.com (дата обращения 23.01.2016). Загл. с экрана. Яз. рус.

10 Кауфман, В.Ш. Языки программирования. Концепции и принципы. 2-е издание / В.Ш. Кауфман. М.: ДМК Пресс, 2011. 464 с.

11 Троелсен, Э. C# и платформа .NET. Библиотека программиста / Э. Троелсен. СПб.: Питер, 2006. 796 с.

12 Джексон, П. Введение в экспертные системы / П. Джексон. Вильямс, 2001. 393 с.