Министерство науки и высшего образования РФ

Федеральное государственное автономное

образовательное учреждение высшего образования

**«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Институт космических и информационных технологий

институт

Межинститутская базовая кафедра

«Прикладная физика и космические технологии»

кафедра

УТВЕРЖДАЮ Заведующий кафедрой

\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

подпись инициалы, фамилия

« \_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_20 \_\_\_ г.

**МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ**

«Разработка методики интеграции экспертных систем в сетевые пользовательские приложения»

тема

09.04.01.03 Информатика и вычислительная техника

код и наименование направления

09.04.01 «Информационные системы космических аппаратов и центров управления полетами»

код и наименование магистерской программы

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Руководитель |  |  |  |  |  | В.А. Углев |
|  |  | подпись, дата |  | должность, ученая степень |  | инициалы, фамилия |
| Выпускник |  |  |  |  |  | Н.А. Болсуновский |
|  |  | подпись, дата |  |  |  | инициалы, фамилия |
| Рецензент |  |  |  |  |  |  |
|  |  | подпись, дата |  | должность, ученая степень |  | инициалы, фамилия |
| Нормоконтролер |  |  |  |  |  | В.Е. Чеботарев |
|  |  | подпись, дата |  |  |  | инициалы, фамилия |

Красноярск 2023

Министерство науки и высшего образования РФ

Федеральное государственное автономное

образовательное учреждение высшего образования

**«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Институт космических и информационных технологий

институт

Межинститутская базовая кафедра

«Прикладная физика и космические технологии»

кафедра

УТВЕРЖДАЮ Заведующий кафедрой

\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

подпись инициалы, фамилия

« \_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_20 \_\_\_ г.

**ЗАДАНИЕ**

**НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ**

**в форме** магистерской диссертации

Красноярск 2023

Студенту\_Болсуновскому Николаю Александровичу\_

фамилия, имя, отчество

Группа\_КИ20-01-3М\_ Направление (специальность) 09.04.01

номер код

Информатика и вычислительная техника\_

наименование

Тема выпускной квалификационной работы Разработка методики интеграции экспертных систем в сетевые пользовательские приложения

Утверждена приказом по университету № \_\_\_\_\_\_\_ от\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель ВКР: В.А. Углев, к.т.н

инициалы, фамилия, должность, ученое звание и место работы

Исходные данные для ВКР: в рамках ВКР разрабатывается методика интеграции экспертных систем в сетевые пользовательские приложения.

Перечень графического материала: презентация PowerPoint, раздаточный материал.

Руководитель ВКР \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.А. Углев

Подпись, инициалы и фамилия

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Н.А. Болсуновский

подпись, инициалы и фамилия студента

« \_\_\_ » \_\_\_\_\_\_\_\_\_\_ 20\_\_ г.

**РЕФЕРАТ**

Выпускная квалификационная работа по теме «Разработка методики интеграции экспертных систем в пользовательские проекты» содержит 69 страниц текстового документа, 37 использованных источников.

*Искусственный интеллект, экспертные системы, нечеткая логика, продукционные правила, FLM\_Builder.*

Целью диссертационного исследования является разработка методики интеграции экспертных систем в сетевые пользовательские приложения.

Задачи диссертационного исследования:

1 выполнить анализ предметной области и патентные исследования по теме методики интеграции экспертных систем в сетевые пользовательские приложения;

2 разработать методику интеграции экспертных систем в сетевые пользовательские приложения;

3 провести эксперимент.

В результате ВКР была разработана новая методика интеграции экспертных систем в сетевые пользовательские приложения. В результате эксперимента предложенной методики, проведенного в рамках учебного процесса магистратуры СФУ были получены результаты, которые подтверждают удобство и быстроту интеграции экспертных систем в сетевые пользовательские приложения

|  |
| --- |
|  |

**Содержание**

[Введение 4](#_Toc12258539)

[1 Глава 1 8](#_Toc12258540)

[1.1 ЭС и проблемы их интеграции с пользовательскими приложениями 8](#_Toc12258542)

[1.1.1 Об экспертных системах (ЭС) 10](#_Toc12258543)

[1.1.2 Классификация ЭС 13](#_Toc12258544)

[1.1.3 Проблемы интеграции ЭС в пользовательские приложения 17](#_Toc12258545)

[1.2 Продукционные ЭС и система FLM\_Builder 20](#_Toc12258546)

[1.3 Постановка задачи 20](#_Toc12258546)

[1.4 Выводы по главе 20](#_Toc12258546)

[2 Глава 2. 22](#_Toc12258547)

[2.1 Проект модульной организации продукционной ЭС 22](#_Toc12258548)

[2.2 Организация модели ЭС 24](#_Toc12258549)

[2.3 Механизмы просчёта 28](#_Toc12258553)

[2.4 Методы интеграции ЭС 30](#_Toc12258553)

[2.5 Выводы по главе 34](#_Toc12258553)

[3 Глава 3 42](#_Toc12258557)

[3.1 Модуль FLM\_Modul.py 42](#_Toc12258558)

[3.2 Сетевой конструктор 44](#_Toc12258558)

[3.3 Вычислительный эксперимент 47](#_Toc12258558)

[3.4 Выводы по главе 54](#_Toc12258558)

[Заключение 63](#_Toc12258566)

[Список сокращений 64](#_Toc12258567)

[Список использованных источников 65](#_Toc12258568)

**ВВЕДЕНИЕ**

*Актуальность работы.*

Экспертные системы являются актуальной темой исследований и разработок в настоящее время. В современном мире, где данные становятся все более объемными и сложными, экспертные системы могут помочь в обработке и анализе информации, а также принимать решения на основе этой информации, таких как медицина, финансы, производство и другие. Эти программы используют знания экспертов в определенной области, чтобы помочь пользователям принимать решения и решать сложные задачи. Экспертные системы представляют собой инновационный инструмент для автоматизации процессов принятия решений, что позволяет сократить время и улучшить качество принимаемых решений. Они могут быть использованы для решения различных задач, от простых до сложных, и могут помочь неспециалистам в принятии решений на основе знаний и опыта.

Однако, анализ существующих экспертных систем и способов интеграции их в сетевые пользовательские приложения показал необходимость разработки новой методики интеграции экспертных систем в сетевые пользовательские приложения, что бы пользователь, не обладающий достаточными знаниями в программировании смог интегрировать экспертную систему в своё приложение.

*Объектом* исследования является экспертные системы, интегрируемые в сетевые пользовательские приложения.

*Предметом* исследования являются способы интеграции ЭС в сетевые пользовательские проекты.

*Гипотеза* исследования заключается в том, что возможно разработать методику интеграции экспертных систем в сетевые пользовательские приложения, которая будет более эффективной и быстродействующей, чем существующие методы интеграции.

*Целью* диссертационного исследования является разработка методики интеграции сетевых экспертных систем в пользовательские приложения.

*Задачи* диссертационного исследования:

* выполнить анализ предметной области, и патентные исследования по теме интеграции экспертных систем в сетевые пользовательские приложения;
* разработать методику интеграции экспертных систем в сетевые пользовательские приложения;
* провести апробацию предложенной методики в учебном процессе магистратуры СФУ.

При написании диссертации использовались *методы* нечеткой логики, экспертных систем, инженерии знаний, объектно-ориентированного программирования, а также *средства* высокоуровневого программирования.

*Критерием* результата является время, затраченное на интеграцию экспертной системы.

*Научная новизна* диссертационного исследования состоит в предложенной методике интеграции экспертных систем в сетевые пользовательские приложения.

*Практическая значимость* данного диссертационного исследования заключается в возможности расширения функционала пользовательского приложения при помощи интегрированной экспертной системы, что позволяет существенно сократить время на разработку по сравнению со стандартными методами программирования. Это было продемонстрировано на примере учебного процесса магистрантов СФУ.

*Перечень разделов ВКР:*

– введение;

– обзор существующих исследований в области интеграции экспертных систем в сетевые пользовательские приложения;

– разработка методики интеграции экспертных систем в сетевые пользовательские приложения;

– проведение апробации предложенной методики в учебном процессе магистратуры СФУ;

– заключение.

1. **Глава 1**

**1.1 ЭС и проблемы их интеграции с пользовательскими приложениями**

* + 1. **Об экспертных системах (ЭС)**

Экспертные системы – это прикладные программы области искусственного интеллекта, в которых база знаний представляет собой формализованные эмпирические знания высококвалифицированных специалистов в какой – либо узкой предметной области. Экспертные системы предназначены для того, чтобы прийти на замену экспертов при решении профессионально сложных задач в силу недостаточного количества специалистов, недостаточной оперативности при принятии решений экспертами или из-за присутствия негативных факторов, которые могут влиять на работу специалистов.

Обычно экспертные системы рассматриваются с точки зрения их применения в двух аспектах: для решения конкретных задач, где они могут быть использованы и в какой области деятельности.

Просчет в современных экспертных системах в основном ведется внутри конструктора, в котором они были созданы и интегрировать их в пользовательские проекты либо не возможно, либо интеграция представляет из себя сложный и трудозатратный процесс.

Общая структура ЭС представлена на рисунке 1. Необходимо иметь ввиду, что ЭС, построенные для реальных задач имеют гораздо сложную структуру, чем схема, представленная на рисунке, но представленные блоки в любом случае будут присутствовать в любой ЭС.   
**Схема работы с ЭС:**

- заполнение входных данных;

- отправка запроса к ЭС

- получение ответа от решателя из состава ЭС, который используя базу знаний

- получение разъяснений к полученному ответу от подсистемы объяснений.

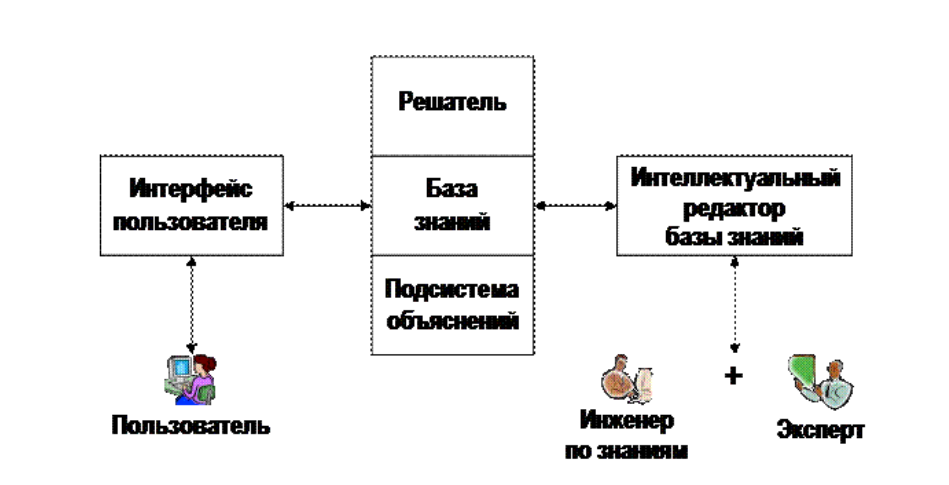


Рис. 1. – Общая структура ЭС

Определим термины, использующиеся в процессе разработки ЭС:

Пользователь – человек, нуждающийся в помощи и поддержки со стороны ЭС, в процессе выполнения своих служебных обязанностей. Обычно, это человек не обладающий достаточной компетенцией.

Инженер по знаниям – человек, выступающий промежуточным звеном между базой знаний и экспертом. Он формирует базу знаний, на основании знаний, полученных от эксперта.

Пользовательский интерфейс – программное обеспечение, которое использует пользователь в процессе работы для заполнения входных данных, а так же получения ответа от ЭС.

База знаний – это центральная часть экспертной системы, которая содержит информацию о предметной области и записана на машинный носитель в форме, понятной эксперту и пользователю. Эта информация обычно записана на языке, приближенном к естественному, но также существует и внутреннее "машинное" представление базы знаний.

Решатель – программа, которая имитирует ход мыслей эксперта, используя знания их базы знаний. Так же может называться «машина вывода» или «блок логического вывода».

Подсистема объяснений ответа – программа, которая помогает пользователю понять как ЭС пришла к этому ответу, она может выдать умозаключение, принятое на каждом шаге формирования ответа.

Редактор базы знаний – программа, позволяющую инженеру по знаниям формировать базу знаний.

Выше была описана структура простой ЭС, однако большие промышленные ЭС имеют гораздо более сложную структура, помимо существующих блоков они могут включать в себя базы данных, а так же различные средства обмена данными со сторонними программами и библиотеками.

Данные (знания), содержащиеся в базе знаний можно разделить на несколько категорий:

– Концептуальное знание – это такие знания, представленные в словах человеческой речи, научных терминах и свойствах объектов, которые стоят за этими терминами. Так же сюда входят зависимости, связи между этими терминами, которые тоже выражаются в человеческих словах.

– Фактуальные знания – такие знание, описывающие качественные и количественные характеристики различных объектов. Любое знание содержит информацию, которая может быть представлена в виде базы знания.

– Алгоритмическое знание – такое знание может быть представлено в виде функции или программы.

|  |  |  |  |
| --- | --- | --- | --- |
| № | Показатель | Эксперт | ЭС |
| 1 | Стабильность | Эксперт со временем утрачивает свою компетенцию, особенно после перерыва в работе. | ЭС сохраняет результаты вывода, которые можно спустя время пересмотреть. |
| 2 | Эффективность | Передача знаний от человека к человеку - долгий и дорогой процесс. | Передать искусственную информацию – это всего лишь процесс копирования. |
| 4 | Воспроизводимость | Эксперт от случая к случаю, может принимать различные решения – это все результат воздействия внешних факторов, в том числе эмоциональных. | На ЭС не влияют внешние факторы, результат выполнения ЭС всегда одинаков, при одинаковых входных данных. |
| 5 | Обучение | Эксперт адаптируется, если происходят изменения среды. | ЭС необходимо модифицировать, при изменениях в экспертной области. |
| 6 | Возможность получения экспертных знаний из различных источников. | Знания от одного эксперта. | ЭС может содержать скомбинированные знания, полученные от нескольких экспертов. |
| 7 | Объяснение ответа | Эксперт может быть неспособен объяснить почему он пришел к этому выводу. | ЭС способна объяснить все свои умозаключения, которые привели ее к этому ответу. |

Обычно экспертные системы рассматриваются с точки зрения их применения в двух аспектах: для решения конкретных задач, где они могут быть использованы и в какой области деятельности.

Классификация экспертных систем производят по четырем признакам (рис. 2):

– по связи с реальным временем,

– по типу решаемой задачи,

– по типу ЭВМ,

– по степени интеграции

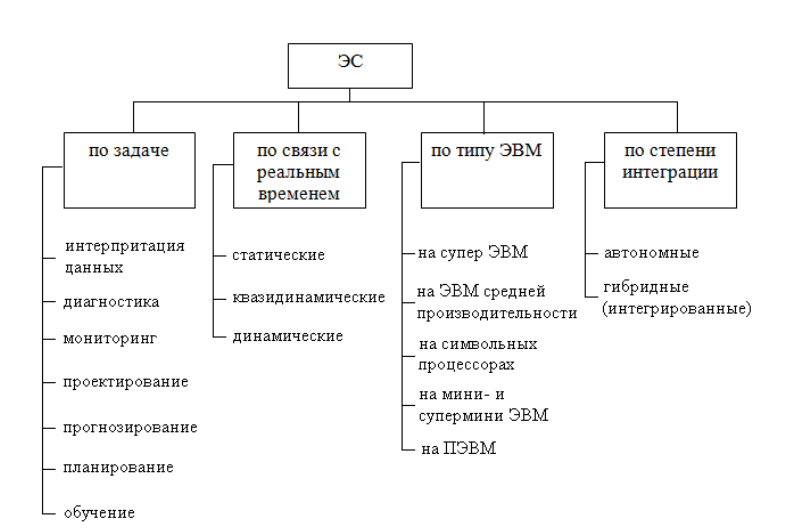


Рис. 2. – классификация экспертных систем по областям применения

***Классификация по решаемой задаче***

По решаемой задаче экспертные системы подразделяются на семь видов:

–Интерпретация данных – одна из традиционных задач для экспертных систем. Под интерпретацией понимается определение смысла данных, результаты которого должны быть согласованными и корректными. Обычно предусматривается многовариантный анализ данных.

Пример: HАSP/SIАP– интерпретирующая система, которая определяет местоположение и типы судов в Тихом океане по данным акустических систем слежения.

– Экспертные системы диагностики. Под диагностикой понимается обнаружение неисправности в некоторой системе, т.е. отклонение от нормы. Такая трактовка позволяет с единых теоретических позиций рассматривать и неисправность оборудования в технических системах, и заболевания живых организмов, и всевозможные природные аномалии.

Пример: Simptоmius – онлайн сервис диагностики заболеваний на базе ЭС. Пациенты должны указать свои симптомы, а Simptоmius выведет список возможных диагнозов, подходящих под указанные симптомы.

– Экспертные системы мониторинга. Основной задачей данных ЭС является непрерывная интерпретация данных в режиме реального времени и сигнализация о выходе тех или иных параметров за допустимые пределы.

Пример: RЕАCTОR–помощь диспетчерам атомного реактора.

– Экспертные системы проектирования. Задача этих ЭС состоит в подготовке спецификаций на создание «объектов» с заранее определёнными свойствами. Под спецификацией понимается весь набор необходимых документов – чертёж, пояснительная записка и так далее. Для организации эффективного проектирования и, в ещё большей степени, перепроектирования необходимо формировать не только сами проектные решения, но и мотивы их принятия. Таким образом, в задачах проектирования тесно связываются два основных процесса, выполняемых в рамках соответствующей экспертной системы: процесс вывода и процесс объяснения.

Пример: XCОN (или R1) – проектирование конфигураций ЭВМ VАX.

– Экспертные системы прогнозирования. Эти системы логически выводят вероятные следствия из - за данных ситуаций. В прогнозирующей системе обычно используется параметрическая динамическая модель, в которой значения параметров «подгоняются» под заданную ситуацию. Выводимые из этой модели следствия составляют основу для прогнозов с вероятностными оценками.

Пример: ЕCОN– прогнозы в экономке.

– Экспертные системы планирования. Эти системы находят планы действий, относящихся к объектам, способным выполнять некоторые функции. В таких экспертных системах используются модели поведения реальных объектов с тем, чтобы логически вывести последствия планируемой деятельности.

Пример: STRIPS — планирование поведения робота.

– Экспертные системы обучения. Эти системы диагностируют ошибки при изучении какой – либо дисциплины с помощью ЭВМ и подсказывают правильные решения. Они собирают знания о гипотетическом «ученике» и его характерных ошибках, затем в ходе своей работы они способны диагностировать слабости в знаниях обучаемых и находить необходимые средства для их устранения.

Пример: PRОUST–обучение языку Паскаль.

– Фотоаппараты и видеокамеры используют нечеткую логику, чтобы реализовать опыт фотографа в управлении этими устройствами. Например, компании Fisher Sanyo производят нечеткие логические видеокамеры, в которых применяется нечеткая фокусировка и стабилизация изображения.

– Компания Matsushita выпускает стиральную машину, в которой используются датчики и микропроцессоры с нечеткими алгоритмами управления. Датчики определяют цвет и вид одежды, степень загрязнения, а нечеткий микропроцессор выбирает наиболее подходящую программу стирки из 600 доступных комбинаций температуры воды, количества стирального порошка и времени стирки.

– В автомобилестроение. Компания Mitsubishi выпустила первый в мире автомобиль, где управление каждой системой основано на нечеткой логике. Эта же компания производит «нечеткий» кондиционер, который управляет изменением температуры и влажности в помещении согласно человеческому восприятию степени комфорта.

– Компания Nissan разработала «нечеткую» автоматическую трансмиссию и«нечеткую» противоскользящую тормозную систему и реализовала их в одно!

из своих автомобилей повышенной комфортности.

В общем случае все системы, основанные на знаниях, можно подразделить на системы, которые решают задачи анализа, и на системы, которые решают задачи синтеза. Главное отличие задач анализа от задач синтеза заключается в том, что если в задачах анализа множество решений может быть перечислено, и включено в систему, то в задачах синтеза множество решений потенциально строится из решений компонентов или под проблем. Задача анализа – это интерпретация данных, диагностика; к задачам синтеза относятся проектирование, планирование. Комбинированные задачи: обучение, мониторинг, прогнозирование.

***Классификация по связи с реальным временем***

По связи с реальным временем экспертные системы подразделяются на статические, квазидинамические и динамические.

***Статические ЭС*** используются в тех предметных областях, в которых база знаний и интерпретируемые данные не меняются во времени.

***Квазидинамические ЭС*** используются в тех ситуациях, когда происходят изменения с некоторым фиксированным интервалом времени.

***Динамические ЭС*** работают взаимосвязано с датчиками объектов в режиме реального времени. Такие ЭС работают непрерывно, интерпретируя поступающие в нее данные.

***Классификация по типу ЭВМ***

Существуют следующие экспертные системы по типу ЭВМ:

– ЭС для выполнения уникальных стратегически важных задач на суперЭВМ

(CОNVЕX, Эльбрус, CRАY др.);

– ЭС на ЭВМ средней производительности (типа ЕС ЭВМ, mаinfrаmе);

– ЭС на рабочих станциях и символьных процессорах (Аpоllо, Silicоn Grаphics, SUN);

– ЭС на мини и супер мини ЭВМ (VАX, micrо –VАX и др.);

– ЭС на настольных компьютерах (ВМ РС, МАС II и др.).

***Классификация по степени интеграции с другими программами***

По степени интеграции с другими программами экспертные системы можно разделить на автономные и гибридные.

***Автономные ЭС*** работают в режиме консультирования пользователя для уникальных задач, для решения которых не нужно использовать традиционные методы обработки данных (моделирование, расчеты и др.).

***Гибридные ЭС*** представляют из себя программный пакет‚ который собирает вместе пакеты прикладных модулей (математическую статистику, линейное программирование или системы управления базами данных) и средства обработки знаний. Это может быть интеллектуальная надстройка над пакетами прикладных программ или вшитая среда для решения нестандартной задачи с элементами экспертных знаний.

* + 1. **Классификация на основе критерия модульности: анализ и систематизация подходов к разбиению систем на модули**

ЭС подразделяются на:

- настольные приложения создания и просчёта экспертных систем,

- модульные приложения создания и просчёта экспертных систем, интегрируемые в настольные пользовательские приложения,

- модульные сетевые приложения создания и просчёта ЭС, интегрируемые в сетевые приложения или сетевые языки программирования.

Существующие конструкторы ЭС:

**VisiRule** – конструктор экспертных систем, представленный в виде настольного приложения (рис. 3). VisiRule предоставляет графический интерфейс для создания диаграмм, которые затем преобразуются в экспертную систему. По – этому VisiRule позволяет создавать экспертные системы без знания программирования.

VisiRule поддерживает такие типы правил, как IF-THEN, правила с несколькими условиями.

Просчет ЭС, созданной в VisiRule, возможно провести как в самом конструкторе, так и в онлайн сервисе visiruleexamples.com

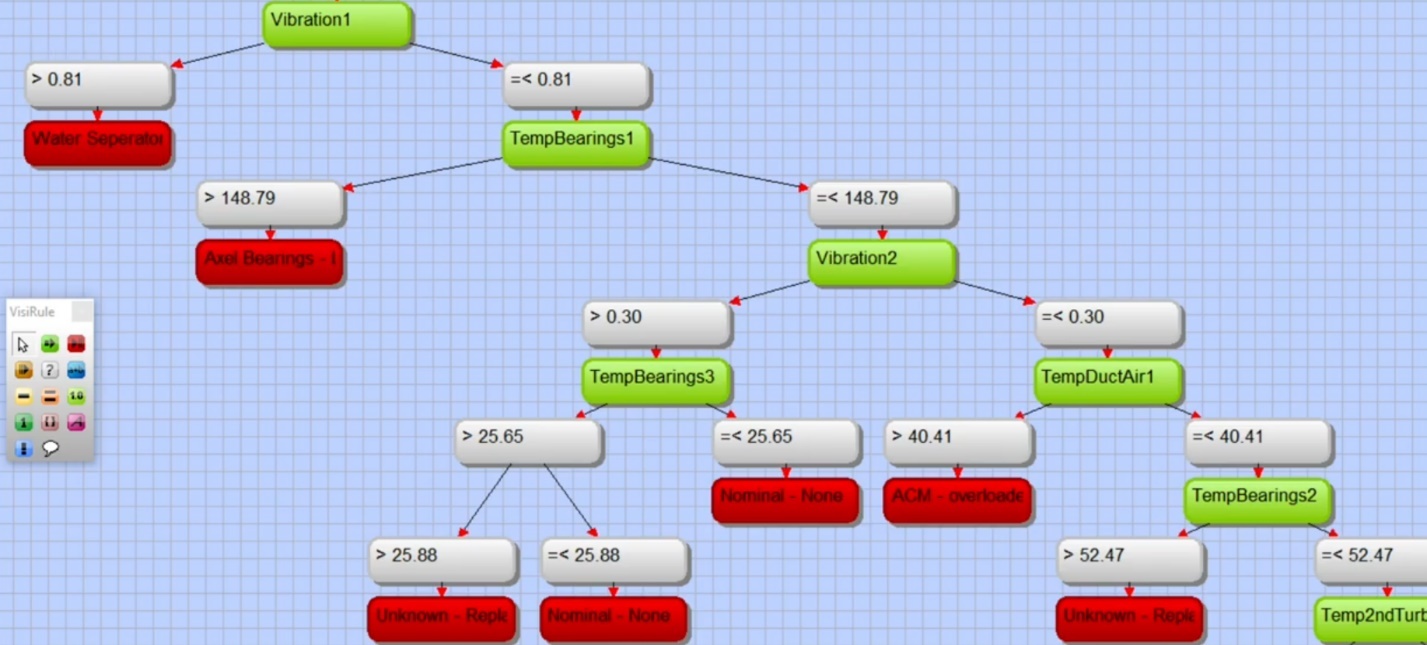


Рис. 3. – Внешний вид ЭС VisiRule

Как и любое программное обеспечение, VisiRule имеет свои недостатки. Такие как,

- Невозможность интеграции ЭС, в пользовательские приложения;

- Сложность в использовании, особенно для тех, кто не имеет опыта в создании или поддержке экспертных систем.

- VisiRule достаточно дорог для людей, потому что рассчитан на большие компании.

- VisiRule имеет ограниченную поддержку, особенно если вы столкнулись с проблемами, которые не могут быть решены через документацию или сообщество пользователей.

В целом, VisiRule является мощным инструментом для создания экспертных систем, но как и любое программное обеспечение – имеет свои ограничения и недостатки, которые могут быть важны при выборе инструмента для создания экспертных систем

**Criterium Decision Plus (CDP)**

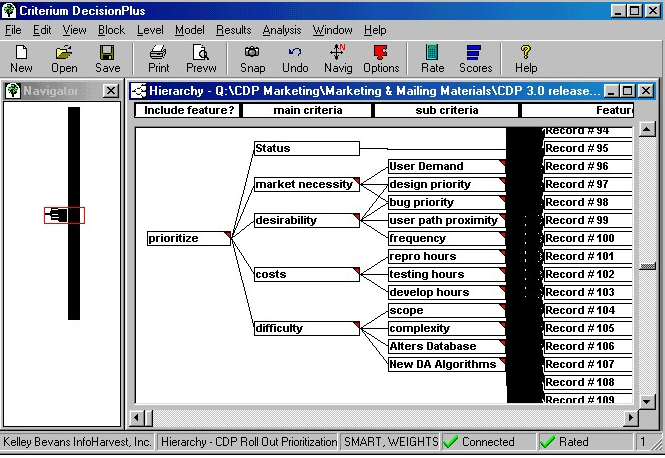
****

Рис. 4. – ВнешнийвидЭС Criterium Decision Plus

Criterium Decision Plus – настольное приложение создания и просчёта экспертных систем, которое использует метод анализа иерархий (МАИ) для оценки и сравнения различных альтернативных вариантов. МАИ - это методология, разработанная Томасом Саати, который позволяет разбить сложную проблему на более простые части и оценить их относительное значение.

CDP позволяет пользователям создавать иерархические модели, которые включают в себя критерии, подкритерии и альтернативы. Пользователи могут оценить каждый критерий и подкритерий относительно друг друга, чтобы определить их важность для принятия решения. Затем они могут оценить каждую альтернативу относительно каждого критерия и подкритерия, чтобы определить, какая альтернатива наиболее подходит для решения проблемы.

CDP также предоставляет пользователю возможность проводить сценарный анализ, чтобы оценить, как изменение весов критериев и подкритериев может повлиять на результаты принятия решений. Кроме того, CDP может генерировать отчеты и графики, которые помогают пользователям лучше понимать результаты моделирования и принимать более обоснованные решения.

CDP может быть полезен для принятия решений в различных областях, однако имеет ряд недостатков, таким как его зависимость от качества входных данных. Если данные, используемые для создания модели принятия решений, не точны или неполны, то результаты модели могут быть неточными или недостоверными. Так же он сложен в настройке самого приложения и настройке параметров, что может потребовать дополнительного времени и усилий со стороны пользователя. Данный конструктор может не подходить для всех типов принятия решений и может быть ограничен в своих возможностях для определенных сценариев. Поэтому перед использованием этого инструмента необходимо тщательно оценить его преимущества и недостатки в контексте конкретной задачи.

**ES** – **BuilderWeb**

ES – Builder Web – это простой образовательный инструмент, разработанный для студентов с целью развития навыков работы с экспертными системами, представленный в виде онлайн конструктора создания и просчёта экспертных систем.

Проект представляет собой дальнейшее развитие настольной версии приложения ES – Builder 3.0 для Windows в улучшенный веб – интерфейс с использованием фреймворка разработки AJAX.

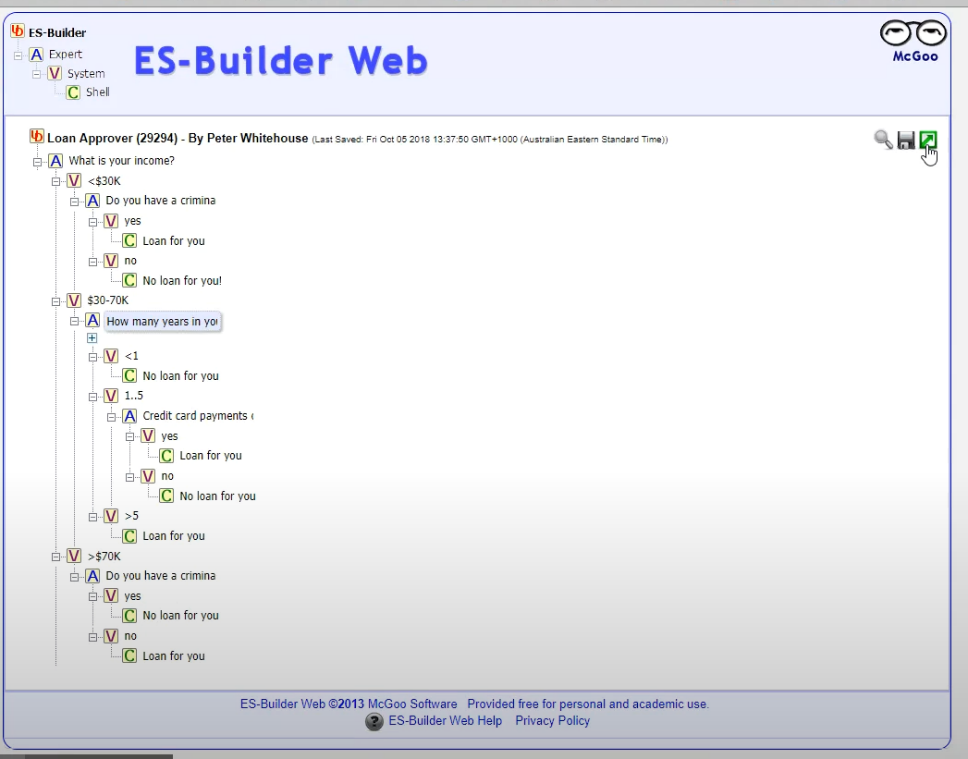


Рис. 5. – Внешний вид ЭС ES – Builder Web

ES – Builder Web это достаточно удобный конструктор для обучения студентов работе с ЭС, но сложную ЭС в нем создать не получится.

**Prolog**

Пролог (англ. Prolog) – язык и система логического программирования, основанные на языке предикатов математической логики дизъюнктов Хорна, представляющей собой подмножество логики предикатов первого порядка.

Язык сосредоточен вокруг небольшого набора основных механизмов, включая сопоставление с образцом, древовидного представления структур данных и автоматического перебора с возвратами. Хорошо подходит для решения задач, где рассматриваются объекты (в частности структурированные объекты) и отношения между ними. Пролог, благодаря своим особенностям, используется в области искусственного интеллекта, компьютерной лингвистики и нечислового программирования в целом. В некоторых случаях реализация символьных вычислений на других стандартных языках вызывает необходимость создавать большое количество кода, сложного в понимании, в то время как реализация тех же алгоритмов на языке Пролог даёт простую программу, легко помещающуюся на одной странице.

Prolog является декларативным языком программирования: логика программы выражается в терминах отношений, представленных в виде фактов и правил. Для того чтобы инициировать вычисления, выполняется специальный запрос к базе знаний, на которые система логического программирования генерирует ответы «истина» и «ложь». Для обобщённых запросов с переменными в качестве аргументов созданная система Пролог выводит конкретные данные в подтверждение истинности обобщённых сведений и правил вывода .Иначе говоря, предикат можно определить как функцию, отображающую множество произвольной природы в множество булевых значений {ложно, истинно}. Задача пролог - программы заключается в том, чтобы доказать, является ли заданное целевое утверждение следствием из имеющихся фактов и правил.

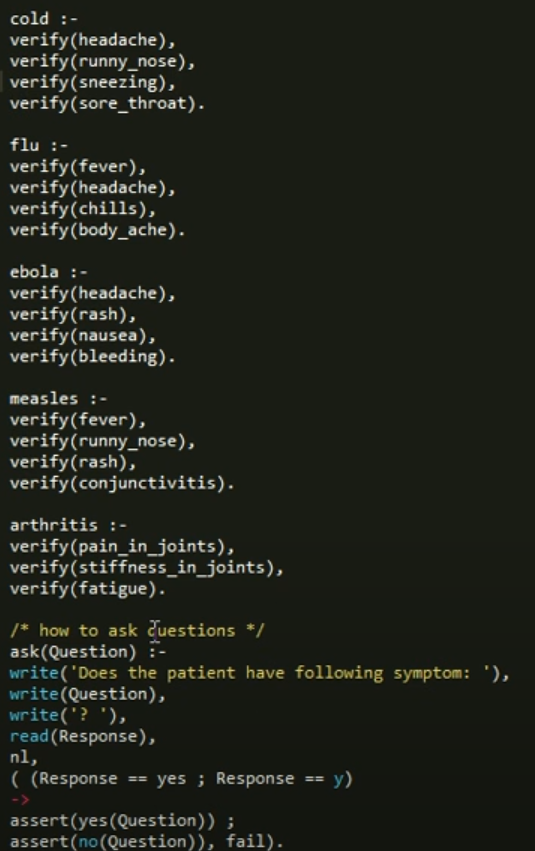


Рис. 6. – Внешний вид ЭС написанной на Prolog

Prolog явно плох в обработке чисел. Это выражается в том, что он производит множество вычислений над числами с плавающей точкой, в значительной степени бессмысленных, в ходе чего накапливаются ошибки, которые никто не может явно понять или обработать.

Это область, в которой реализация Prolog приложений не может конкурировать, так как сегодня все эти вычисления сопряжены со слишком большими накладными расходами. Так же в Prolog относительно обширная терминология.

Чтобы выполнить Prolog код в сетевом приложении:

* Установить SWI - Prolog на ваш сервер.
* Создайте серверный API, который будет обрабатывать запросы на выполнение Prolog исходников. Для этого вы можете использовать любой серверный язык программирования, например, Node.js или Python.
* Используйте библиотеку swipl.js, чтобы вызывать SWI-Prolog из вашего серверного API. Эта библиотека позволяет выполнить Prolog исходник и получить результат в формате JSON.
* Создайте клиентскую часть вашего приложения на React, которая будет отправлять запросы на серверный API и отображать результаты.
* Реализуйте механизм безопасности, чтобы предотвратить выполнение вредоносного кода на вашем сервере. Например, можно использовать модуль child\_process в Node.js для запуска SWI-Prolog в отдельном процессе.

Prolog - это язык программирования, который был разработан в 1970-х годах для решения задач искусственного интеллекта и логического программирования. В то время он был одним из наиболее популярных языков для создания экспертных систем и других приложений, основанных на логическом выводе.

Однако, с развитием интернета и веб-технологий, Prolog потерял свою популярность и значимость. Он не предназначен для того, чтобы ЭС, созданные в нем в веб - приложений и не имеет поддержки для работы с базами данных, что делает его менее привлекательным для современного программирования.

Сегодня большинство веб - приложений создаются с использованием других языков программирования, таких как JavaScript, Python,. Эти языки имеют мощные библиотеки и фреймворки для работы с веб-технологиями и базами данных, что делает их более подходящими для создания современных веб - приложений.

Тем не менее, Prolog все еще используется в некоторых областях, таких как искусственный интеллект, логическое программирование, анализ данных и т.д. Он также может быть полезен для решения специфических задач, которые требуют логического вывода и рассуждения.

* + 1. **Проблемы интеграции ЭС в пользовательские приложения**

Современные конструкторы ЭС построены так, что ЭС, созданные в них достаточно трудно внедрить в сторонние пользовательские сетевые приложения, и рассчитаны на просчет созданной ЭС внутри самого конструктора.

Возьмем следующие критерии, для классификации рассмотренных ЭС:

*WISIVIG редактор* – это возможность прикладных программ или веб – интерфейсов, которая позволяет отображать содержимое в процессе редактирования так, чтобы оно выглядело максимально близко к конечной продукции, такой как печатный документ, веб – страница или презентация. Название этой аббревиатуры WYSIWYG происходит от «What You See Is What You Get», что означает "что видишь, то и получишь».

*Модульность* – это свойство программного обеспечения, которое означает, что оно состоит из отдельных компонентов, называемых модулями. Каждый модуль выполняет определенную функцию и может быть разработан и тестирован независимо от других модулей. Это позволяет упростить разработку, тестирование и сопровождение программного обеспечения, а также повторно использовать код в разных проектах. Модульность также облегчает совместную работу разработчиков, поскольку каждый может работать над своим модулем, не вмешиваясь в работу других. Кроме того, модульность позволяет улучшить надежность и безопасность программного обеспечения, поскольку ошибки и уязвимости могут быть легче выявлены и исправлены в отдельных модулях. В данном случае под модульностью надо понимать разделение системы построения ЭС на *модуль конструктора ЭС и модуль просчета ЭС.*

*Поддержка нечеткой логики (НЛ):*

Нечеткая логика (фаззи-логика) - это метод решения проблем, которые не могут быть точно определены или измерены. В отличие от традиционной бинарной логики, которая работает с двумя значениями (истина/ложь), нечеткая логика работает с нечеткими или приблизительными значениями. Она используется в системах управления, прогнозированиях, робототехнике, искусственном интеллекте и других областях.

Основная идея нечеткой логики заключается в том, что каждый элемент имеет степень принадлежности к определенному классу. Например, если мы говорим о цвете, то красный цвет может иметь степень принадлежности 0,8, а оранжевый - 0,3. Эти значения могут быть использованы для принятия решений на основе нечетких данных.

Для реализации нечеткой логики используются нечеткие множества, нечеткие правила и нечеткие выводы. Нечеткие множества представляют собой наборы элементов с разными степенями принадлежности к классу. Нечеткие правила определяют связи между элементами и классами. Нечеткий вывод позволяет получить результат на основе нечетких правил и данных.

Преимущества нечеткой логики включают возможность работать с нечеткими и неопределенными данными, упрощение моделирования сложных систем, повышение точности прогнозирования и улучшение производительности системы. Нечеткая логика также позволяет учитывать не только точечные значения, но и их изменения во времени и пространстве. Это делает ее полезной для решения задач в области управления, где необходимо учитывать множество факторов и условий.

В целом, нечеткая логика представляет собой мощный инструмент для решения проблем, связанных с неопределенностью и нечеткостью данных. Она может быть использована в различных областях, включая науку, технологии и бизнес.

*Поддержка сети* в данном случае означает возможность конструирования и/или просчета созданной ЭС в онлайн сервисе.

*Поддержка сетевых языков* означает возможность встраивания модуля просчета в отдельный пользовательский проект, написанный на одном из сетевых языков программирования JavaScript или Python.

Поддержка сетевых языков программирования, таких как JavaScript и Python, означает, что разработчики могут встраивать модули для работы с сетью и интернетом в свои пользовательские проекты, написанные на этих языках. Это позволяет создавать более сложные и функциональные приложения, которые могут взаимодействовать с другими приложениями и сервисами через интернет.

JavaScript и Python являются одними из наиболее популярных языков программирования в мире, и они широко используются для создания веб-приложений, мобильных приложений, игр и других программных продуктов. Оба языка имеют встроенную поддержку для работы с сетью и интернетом, что делает их очень удобными для создания приложений, которые требуют взаимодействия с другими приложениями и сервисами через интернет.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ЭС | WISIVIG-редактор | Модульность | Поддержка НЛ | Поддержка сети | Поддержка сетевых языков | Отсутствие необходимости развертывать на сервере среду просчёта |
| VisiRule | + | - | - | + | - | + |
| CDP | + | - | - | - | - | + |
| ES-BuilderWeb | + | - | + | + | - | + |
| Prolog | - | + | + | - | - | + |
| FLM BUILDER | -/+ | + | + | - | - | - |

* 1. **Продукционные ЭС и система FLM\_Builder (метод нечёткой логики)**

*Продукционные экспертные системы* – это системы, основанные на правилах, которые организованы в виде IF-THEN структур, называемых продукционными правилами. Эти правила описывают условия, которые необходимо выполнить, чтобы применить правило, а также действия, которые необходимо выполнить, если правило выполняется. Интерпретатор управляет активизацией правил в зависимости от имеющихся фактов, что позволяет системе автоматически принимать решения и повышать эффективность работы экспертов. Продукционные экспертные системы широко используются в различных областях, таких как медицина, финансы, промышленность и т.д.

В продукционных системах знания представлены в виде множества правил, на основе которых формируются выводы для различных ситуаций. В зависимости от метода логического вывода, системы могут быть с прямым или обратным логическим выводом. Общая стратегия решения задач заключается в разбиении задач на фрагменты, которые можно легче доказать. Системы с прямым логическим выводом начинают работу с известных начальных фактов и используют правила для создания выводов или выполнения определенных действий.

Системы, использующие обратный логический вывод, оперируют гипотезами, начиная свою работу с цели, которую пользователь пытается доказать. Затем они продолжают поиск правил, которые могут подтвердить правдивость этой гипотезы. Продукционные правила, на которых основаны такие системы, широко применяются благодаря своим особенностям.

Системы, основанные на продукционных правилах, широко применяются благодаря своим особенностям, которые включают в себя:

– Модульность. Благодаря модульной организации, представление знаний в экспертных системах становится более простым и удобным, а также упрощается процесс расширения системы, поэтапно наращивая ее возможности.

– Подсистема объяснений. Продукционные экспертные системы обладают возможностью создания легко понятных средств объяснения, благодаря правилам, которые используются в их работе. Эти средства могут отслеживать последовательность активированных правил и восстанавливать ход рассуждений, которые привели к определенному выводу.

– Принципы, на которых основаны продукционные экспертные системы, имеют аналогию с познавательным процессом человека, как показали исследования Ньюэлла и Саймона. Это означает, что правила являются естественным способом моделирования процесса решения задач человеком. Поэтому при выявлении экспертных знаний нет необходимости объяснять экспертам сложную структуру представления знаний, потому что используется простое представление в виде правил IF – THEN.

*Нечеткая логика* – это раздел математики и искусственного интеллекта, который занимается разработкой методов и алгоритмов для работы с нечеткими данными и нечеткими понятиями. Нечеткость возникает в тех случаях, когда необходимо работать с данными, которые не могут быть однозначно определены или описаны. Например, это может быть описание качества продукта или оценка эксперта.

Основными понятиями нечеткой логики являются нечеткие множества и нечеткие правила. Нечеткое множество - это множество, элементы которого имеют различную степень принадлежности к этому множеству. Например, множество "высокие температуры" может содержать элементы с различными степенями принадлежности, например, 0.8 для температуры 30 градусов и 0.2 для температуры 25 градусов.

Нечеткая логика является важным инструментом для работы с нечеткими данными и понятиями. Она позволяет получать выводы на основе нечетких правил и множеств, что может быть полезно в различных областях. Благодаря своей универсальности и гибкости, нечеткая логика продолжает привлекать внимание исследователей и практиков.

*Система FLM\_Builder (*Fuzzy Logic Model Builder*)* – это конструктор экспертных систем, созданный в 2005 году для обучения студентов информационных специальностей в Красноярском государственном техническом университете (сейчас Сибирский федеральный университет). Она основывалась на композиционном выводе и методе нечеткой логики для обработки входных данных. Модуль работы с моделями был написан на языке программирования Object Pascal, а интерфейсная часть была реализована в виде оконного приложения в среде Delphi. Создание экспертных систем происходило в режиме конструктора и включало формирование архитектуры процесса рассуждений, описание этапов принятия решений и заполнение базы знаний. Результат сохранялся в виде текстового файла с расширением \*.flm.

С помощью приложения FLM\_Builder пользователь мог интегрировать модель ЭС в произвольные приложения в среде Delphi. Для этого ему нужно было скопировать файл flm в каталог нового проекта и разместить там же FLM\_modul.pas. Затем модель загружалась и вызывалась функция ее просчета, где вектор входных значений передавался в виде аргументов. Работы по интеграции flm – модели ограничивались всего 5 –7 строками типового кода. Так как изначально все студенты обучались основам программирования в Delphi, то сложностей в организации простейших интерфейсов при выполнении практических работ по данной теме у них не возникало. Пример кода для интеграции flm – модели приведен ниже (модель MyESModel.flm имеет три фактора на входе и два ответа на выходе).

В процессе использования программы FLM\_Builder и ее интеграционных модулей были внесены существенные изменения в основные подходы и технологии разработки систем. Настольные уступили место сервисному подходу через интернет – формы. В 2008 году была предпринята первая попытка реализации такого функционала, но она не была реализована в сервисной архитектуре и требовала полной переработки. В программе FLM\_Builder есть недостатки, такие как ограничение числа слоев логического вывода, невозможность указать частичную связность при переходе между слоями, отсутствие возможности вызова пользовательских функций в процессе просчета и отсутствие протоколирования промежуточных значений. Также возникают сложности при интеграции модели ЭС в код языка Python и JavaScript. В связи с этим возникла необходимость перепроектирования FLM\_Builder как сервиса.

Код для интеграции экспертной системы, созданной в конструкторе экспертных систем FLM\_Builder в пользовательское настольное приложение Delphi представлен на рисунке 6

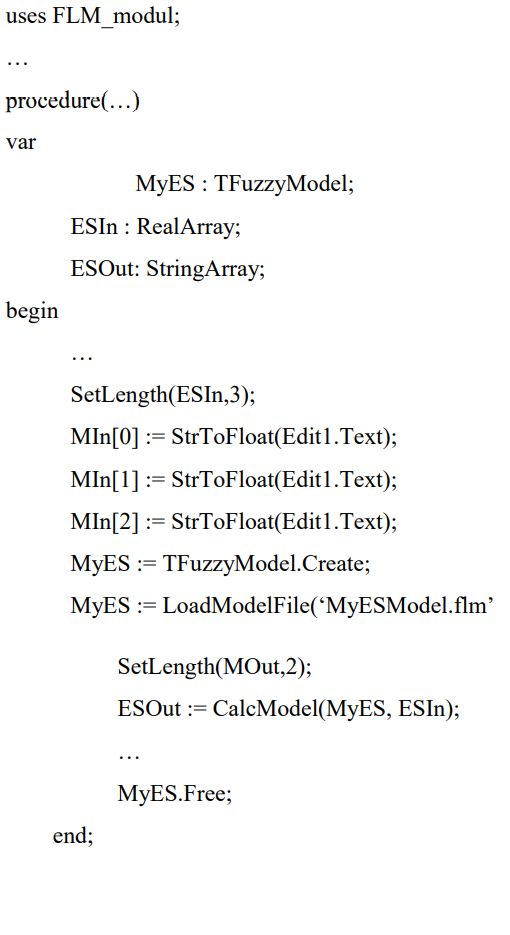


Рис. 6 – Код интеграции ЭС FLM\_Builder в проект Delphi

Из-за расширения номенклатуры языков программирования и специальностей студентов, стало необходимо интегрировать FLM-модели в код приложений за пределами учебного процесса, в том числе и в проекты на языке С++

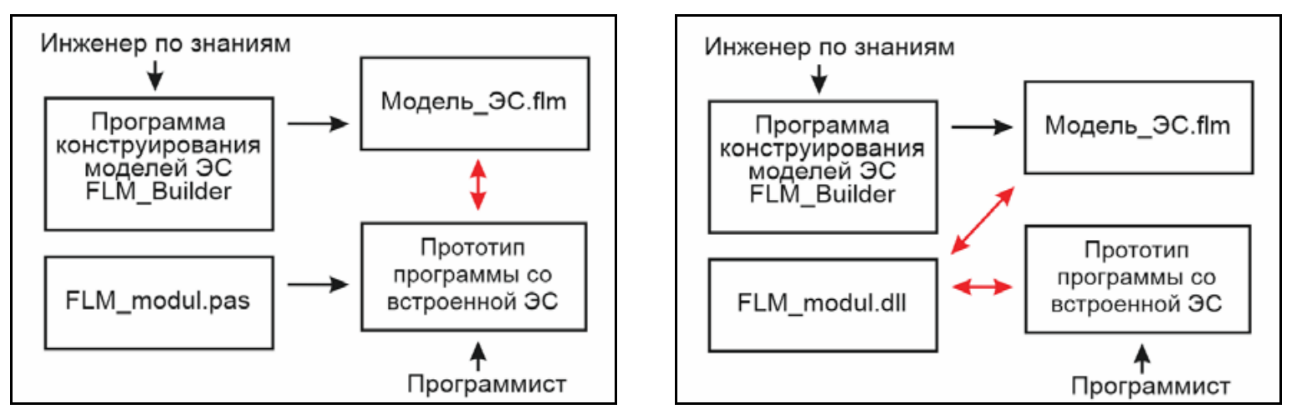


Рис. 7– Схема интеграции flm-файлов в пользовательские проекты на

базе модуля FLM\_modul.pas (слева) и его dll версии (справа)

В последние годы наблюдается значительный сдвиг в предпочтениях пользователей и разработчиков программного обеспечения. Технологический прогресс и развитие веб – технологий привели к возникновению новой эры в мире приложений – эра веб – приложений. Веб – приложения стали популярными и заняли первое место в выборе пользователей, вытеснив десктопные приложения. Преимущества, которые предлагают веб-приложения:

– *Доступность*. Одним из главных преимуществ веб-приложений является их доступность. Веб-приложения могут быть запущены на любом устройстве с веб – браузером, будь то компьютер, планшет или смартфон. Это означает, что пользователи могут получить доступ к своим данным и функциональности веб-приложений в любое время и в любом месте, где есть доступ к интернету.

– *Обновления и установка*. Веб-приложения обладают преимуществом в обновлениях и установке. Пользователям больше не нужно устанавливать и обновлять приложения на своих устройствах. Разработчики могут выпускать обновления веб – приложений непосредственно на сервере, и пользователи автоматически получают последнее обновление при запуске веб – приложения.

В современном информационном обществе, где данные и знания становятся все более доступными, появляется необходимость в эффективном использовании экспертных знаний для принятия решений. Одним из способов достижения этой цели является интеграция экспертных систем в веб-приложения. В этой статье мы рассмотрим, почему такая интеграция становится все более важной и как она может принести значительные преимущества, такие как:

– *Расширение возможностей веб-приложений*. С развитием технологий веб – разработки, веб – приложения становятся все более мощными и функциональными. Однако, в большинстве случаев, эти приложения ограничены в своей способности принимать сложные решения на основе экспертных знаний. Интеграция экспертных систем позволяет расширить возможности веб – приложений и предоставить пользователям более интеллектуальные и информированные решения.

– *Улучшение пользовательского опыта*. Интеграция экспертных систем в веб-приложения позволяет создавать персонализированные и контекстно – зависимые рекомендации и решения для пользователей.

**1.3 Постановка задачи**

*Целью* диссертационного исследования является разработка методики интеграции сетевых экспертных систем в пользовательские приложения.

*Задачи* диссертационного исследования:

* выполнить анализ предметной области, и патентные исследования по теме интеграции экспертных систем в сетевые пользовательские приложения;
* разработать методику интеграции экспертных систем в сетевые пользовательские приложения;
* провести апробацию предложенной методики в учебном процессе магистратуры СФУ.

В процессе разработки методики интеграции экспертных систем в сетевые пользовательские приложения за основу был взят конструктор экспертных систем FLM\_Builder.

**1.4 Выводы по главе**

В главе 1 были рассмотрены экспертные системы и проблемы их интеграции в пользовательские приложения. Была проведена классификация ЭС и рассмотрены основные проблемы, связанные с интеграцией ЭС в пользовательские приложения.

Были рассмотрены актуальные конструкторы ЭС на сегодняшний день, их достоинства и недостатки.

Также в главе были рассмотрены продукционные ЭС и система FLM\_Builder, основанная на методе нечеткой логики. Продукционные ЭС являются наиболее распространенным типом ЭС и могут использоваться для решения различных задач. Система FLM\_Builder позволяет создавать продукционные ЭС с помощью нечеткой логики, что позволяет учитывать неопределенность и нечеткость входных данных.

В главе была поставлена задача, связанная с разработкой методики интеграции ЭС в сетевые пользовательские приложения. В дальнейшем исследовании будет рассмотрено решение этой задачи с помощью системы FLM\_Builder и метода нечеткой логики.

Таким образом, глава 1 магистерской диссертации представляет собой введение в тему исследования, рассматривает основные понятия и проблемы, связанные с экспертными системами и их интеграцией в пользовательские приложения, а также определяет направление дальнейшего исследования.

1. **Глава 2**
   1. **Проект модульной организации продукционной ЭС**

Продукционные экспертные системы (ПЭС) являются мощным инструментом для автоматизации принятия решений в различных областях. Однако, с ростом сложности задач и объема знаний, возникает необходимость в организации ПЭС таким образом, чтобы они были более эффективными и гибкими. В этой главе мы рассмотрим проект модульной организации продукционной экспертной системы и почему он является важным для решения сложных задач.

*Основные принципы модульной организации ПЭС*. Модульная организация ПЭС предполагает разделение системы на отдельные модули, каждый из которых отвечает за определенный аспект решения задачи. Каждый модуль содержит набор правил (продукций), которые определяют логику принятия решений. Это позволяет легко добавлять, изменять или удалять модули в зависимости от требований задачи, что делает систему более гибкой и адаптивной.

*Преимущества модульной организации ПЭС*. Проект модульной организации ПЭС имеет ряд преимуществ, которые делают его эффективным в решении сложных задач. Проект модульной организации ПЭС имеет ряд преимуществ, которые делают его эффективным в решении сложных задач. Вот некоторые из них:

*Гибкость и масштабируемость:* Модульная организация позволяет легко добавлять, изменять или удалять модули в системе. Это означает, что ПЭС может быть адаптирована под различные требования задачи без необходимости полной перестройки системы. Кроме того, модульная организация обеспечивает возможность масштабирования системы путем добавления новых модулей при необходимости.

*Удобство разработки и тестирования:* Разделение системы на модули упрощает процесс разработки и тестирования ПЭС. Каждый модуль может быть разработан и протестирован независимо от других модулей, что упрощает отладку и обнаружение ошибок. Это также позволяет распределить работу между разработчиками и ускорить процесс разработки.

*Повторное использование знаний:* Модульная организация позволяет повторно использовать модули в различных ПЭС или в разных частях одной системы. Это экономит время и ресурсы, так как необходимо разрабатывать и поддерживать только один модуль, который может быть использован в различных контекстах.

*Легкость интеграции*. Каждый модуль может быть разработан и интегрирован независимо от других модулей. Это позволяет разработчикам сосредоточиться на конкретной функциональности и обеспечивает более простую интеграцию модулей в единую систему. Кроме того, при интеграции модулей возможно использование стандартизированных интерфейсов, что упрощает взаимодействие между модулями.

*Улучшенная отказоустойчивость*. Модульная организация ПЭС позволяет локализовать проблемы и сбои в системе. Если один модуль перестает работать, это не приведет к полному отказу системы, а только к некоторым ограничениям функциональности. Остальные модули могут продолжать работать нормально, что обеспечивает более высокую отказоустойчивость и надежность системы.

*Улучшенная безопасность*. Модульная организация также способствует повышению безопасности ПЭС. Каждый модуль может быть разработан с учетом определенных безопасностных требований, и в случае возникновения угрозы безопасности, можно сосредоточиться на защите и обновлении только соответствующих модулей, минимизируя потенциальные риски для всей системы.

На рисунке 8 представлена модульная организация ПЭС.

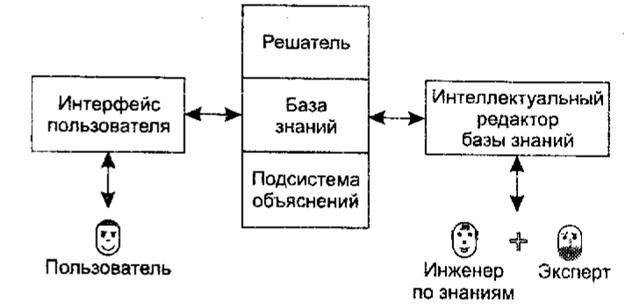


Рис. 8– Структура организации модульной ПЭС.

Основные узлы в ПЭС:

1. Пользователь – человек, который получает сведения из ЭС и может закладывать факты и правила в ЭС.

2. Инженер по знаниям – специалист по искусственному интеллекту, проектирующий и создающий экспертную систему. Обычно инженер по знаниям выступает в роли посредника между экспертом и базой знаний.

3. Эксперт – специалист, приглашённое или нанимаемое лицо для выдачи квалифицированного заключения или суждения по вопросу, которому создается экспертная система.

4. Интерфейс пользователя – это программа, которая дает возможность пользователю взаимодействовать с экспертной системой (ввод, вывод информации).

5. Решатель – (дедуктивная машина, машина вывода, блок логического вывода) – программа, моделирующая ход рассуждений эксперта на основании знаний, имеющихся в базе знаний.

6. База знаний – система, которая содержит знания о предметной области в виде продукционных правил.

7. Подсистема объяснений – подсистема ПЭС, которая служит для вывода заключений, представление комментариев и объяснение мотивов.

8. Интеллектуальный редактор базы знаний – служит для управления правилами в базе знаний (добавление/ редактирование/ удаление).

* 1. **Организация модели ЭС**

Для сохранения модели ЭС был выбран язык разметки xml.

XML - это язык разметки подобный HTML. Расшифровывается как (англ. Extensible Markup Language - Расширяемый Язык Разметки) и является рекомендацией сообщества W3C в качестве языка разметки общего назначения. В отличии от остальных языков разметки, XML сам по себе не определён (это означает, что вы должны сами определять используемые теги).

Модель ЭС в xml разбита на следующие блоки:

1. Блок конфигурации SETUP. В этом блоке сохраняются общие данные об экспертной системе (рисунок 9), а именно:

– В теге NAME хранится название ЭС,

– В теге CAPTION хранится описание модели ЭС ,

– В теге COMMENTS хранятся комментарии к модели ЭС,

– В теге CREATEDATE хранится дата создания модели ЭС,

– В теге MODDIFDATE хранится дата последнего модифицирования модели ЭС,

– В теге USERNAME хранится имя автора модели ЭС,

– В теге MODDIFDATE хранится дата последнего модифицирования модели ЭС,

– В теге INPUTLEVELTERMSET хранится количество терм-множеств, расположенных на уровне входных данных модели ЭС,

– В теге PROCESSINGLEVELTERMSET количество терм-множеств на уровне обработки модели ЭС, OUTPUTLEVELTERMSET

– В теге OUTPUTLEVELTERMSET количество терм-множеств на уровне вывода модели ЭС.

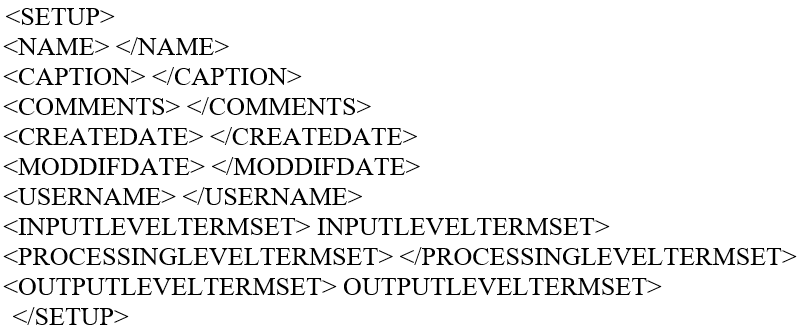


Рис. 9– Блок конфигурации SETUP модели ЭС в xml.

2. Блок INPUTLEVEL. В этом блоке содержится информация о терм –множествах уровня ввода данных (рисунок 10), а именно:

– В теге SET\_1 сохраняется информация о первом терм – множестве уровня ввода.

– В теге NAME сохраняется название переменной терм – множества уровня ввода.

– В теге CAPTION сохраняется название терм – множества уровня ввода.

– В теге TERMCOUNT сохраняется количество термов в терм – множестве уровня ввода.

– В теге TERM1 сохраняется название первого терма уровня ввода.

– В теге TERM2 сохраняется название второго терма уровня ввода.

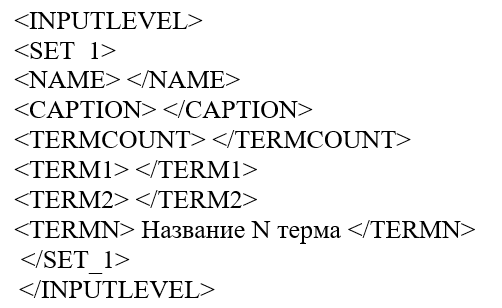


Рис. 10 – Блок INPUTLEVEL модели ЭС в xml.

3. Блок PROCESSINGLEVEL. В этом блоке содержится информация о терм –множествах уровня обработки данных (рисунок 11), а именно:

– В теге SET\_1 сохраняется информация о первом терм – множестве уровня обработки.

– В теге NAME сохраняется название переменной терм – множества уровня обработки.

– В теге CAPTION сохраняется название терм – множества уровня обработки.

– В теге TERMCOUNT сохраняется количество термов в терм – множестве уровня обработки.

– В теге TERM1 сохраняется название первого терма уровня обработки.

– В теге TERM2 сохраняется название второго терма уровня обработки.

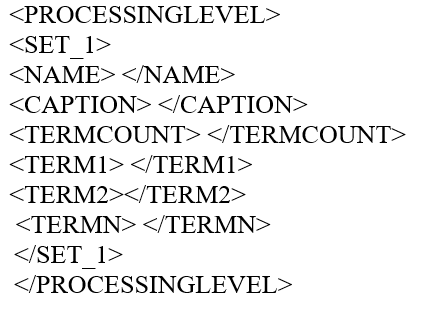


Рис. 11 – Блок PROCESSINGLEVEL модели ЭС в xml.

4. Блок OUTPUTLEVEL. В этом блоке содержится информация о терм –множествах уровня вывода данных (рисунок 12), а именно:

– В теге SET\_1 сохраняется информация о первом терм – множестве уровня вывода.

– В теге NAME сохраняется название переменной терм – множества уровня вывода.

– В теге CAPTION сохраняется название терм – множества уровня вывода.

– В теге TERMCOUNT сохраняется количество термов в терм – множестве уровня вывода.

– В теге TERM1 сохраняется название первого терма уровня вывода.

– В теге TERM2 сохраняется название второго терма уровня вывода.

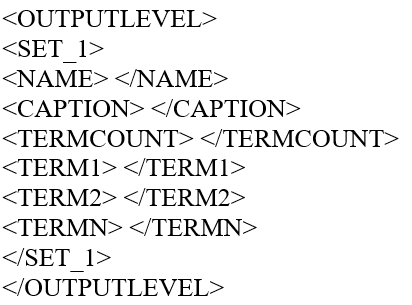


Рис. 12 – Блок OUTPUTLEVEL модели ЭС в xml.

5. Блок RULES. В этом блоке содержится информация о правилах (рисунок 13), а именно:

– В теге RULES\_LEVELS\_COUNT сохраняется информация о правилах между сколькими уровнями.

– В теге RULES\_LEVELS1\_COUNT сохраняется информация о подуровнях правил на уровне 1.

– В теге RULES\_LEVELS2\_COUNT сохраняется информация о подуровнях правил на уровне 2.

– В теге RULES\_LEVEL\_1 сохраняются правила уровня 1.

– В теге RULES\_LEVEL\_1\_1 сохраняются правила подуровня 1 уровня 1.

– В теге RULES\_COUNT сохраняется информация о правилах на подуровне 1\_1.

– В теге RULE\_1, RULE\_2 и т.д. сохраняется значения и коэффициент уверенности.

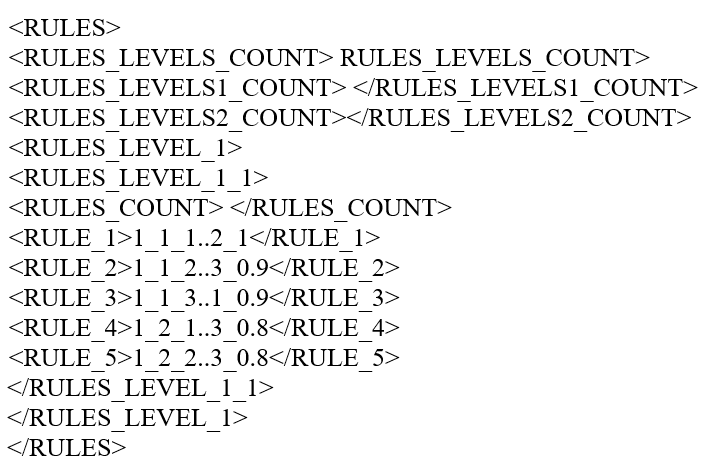


Рис. 13 – Блок RULES модели ЭС в xml.

6. Блок CONNECTIONS. В этом блоке содержится информация о связях между уровнями (рисунок 14), а именно:

– В теге LEVELS\_COUNT сохраняется информация о количестве уровней связей.

– В теге LEVEL\_1 сохраняется информация о связях между первым и вторым уровнями.

– В теге CONNECTIONS\_COUNT сохраняется информация о количестве связей на уровне 1.

– В теге TERM\_1\_1\_1 сохраняются связи входного ТМ 1. Пример: терм из уровня 1 (входной)\_1(1 терм-множество)\_3(3 терм) соединен с терм – множеством 1 и 2 уровня 2 (1 – есть связь/ 0 – связи нет )

– В теге TERM\_1\_1\_2 сохраняются связи входного ТМ 2.

– В теге LEVEL\_2 сохраняется информация о связях между вторым и третьим уровнями.

– В теге RULE\_1, RULE\_2 и т.д. сохраняется значения и коэффициент уверенности.

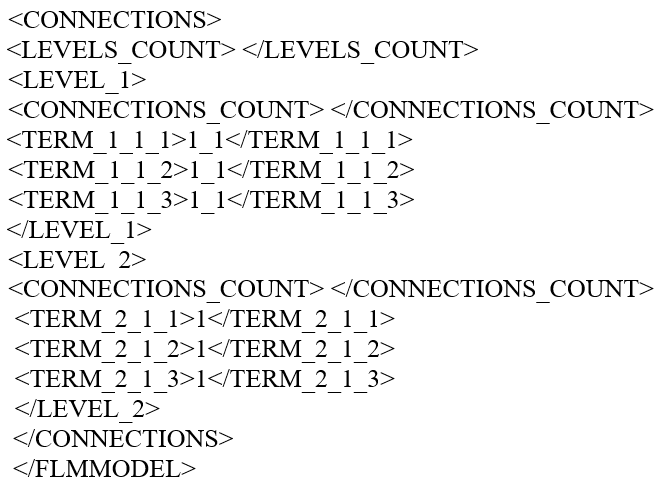


Рис. 14 – Блок CONNECTIONS модели ЭС в xml.