В середине 70-хх годов в ходе изучения искусственного интеллекта было сформировано новое, самостоятельное направление, получившее название экспертные системы.

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.

Классификация экспертных систем производят по четырем признакам (рис. 1):

– по связи с реальным временем,

– по типу решаемой задачи,

– по типу ЭВМ,

– по степени интеграции

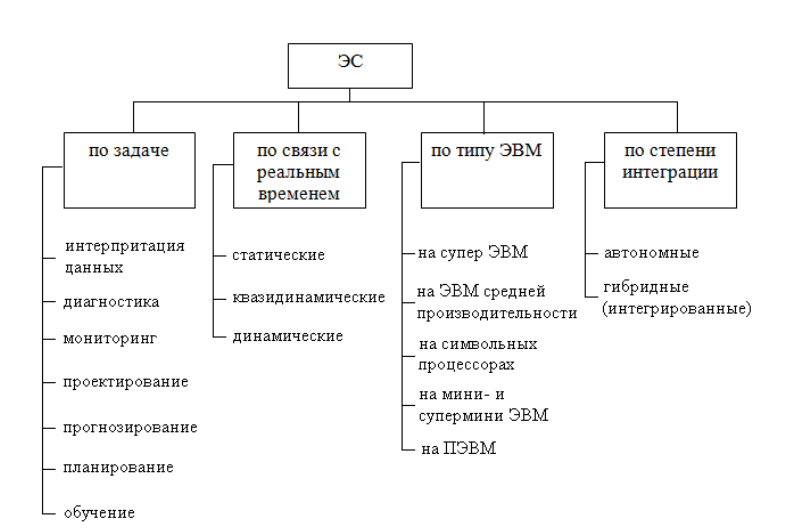


Рис. 1. – классификация экспертных систем по областям применения

***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 заданных 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.

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.

**Помимо приведенных примеров, ЭС используются в:**

– 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.

– 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.

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

По связи с реальным времени 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 супер ЭВМ

(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 супермини 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.

***Структура ЭС***

Общая структура ЭС представлена на рисунке 2. Необходимо иметь ввиду, что ЭС, построенные для реальных задач имеют гораздо сложную структуру, чем представлена на рисунке, но блоки, представленные на рисунке в любом случае будут присутствовать в любой ЭС. Процесс работы с ЭС можно описать следующим образом: пользователь заполняет поля входных данных, после чего посылает запрос с исходными данными в ЭС, после чего решатель из состава ЭС, используя базу знаний – выдает ответ, объясняя, как он пришел к этому ответу, используя систему объяснений.

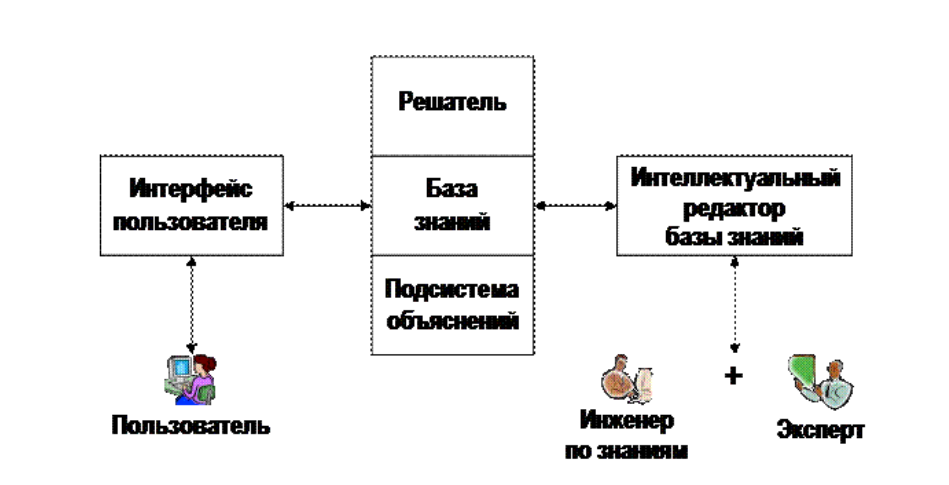


Рис. 2. – Общая структура ЭС

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. |

**Преимущества нечетких систем:**

– возможность оперировать нечеткими входными данными: например, непрерывно изменяющиеся во времени значения (динамические задачи), значения, которые невозможно задать однозначно (результаты статистических опросов, рекламные компании и т.д.);

– возможность нечеткой формализации критериев оценки и сравнения: оперирование критериями “большинство”, “возможно”, “преимущественно” и т.д;

– возможность проведения качественных оценок как входных данных, так и выходных результатов: вы оперируете не только значениями данных, но и их “степенью достоверности” и ее распределением;

– возможность проведения быстрого моделирования сложных динамических систем и их сравнительный анализ с заданной степенью точности: оперируя принципами поведения системы, описанными fuzzy – методами, вы во – первых, не тратите много времени на выяснение точных значений переменных и составление описывающих уравнений, во – вторых, можете оценить разные варианты выходных значений.

**Использование нечеткого управления рекомендуется:**

– для очень сложных процессов, когда не существует простой математической модели;

– для нелинейных процессов высоких порядков;

– если должна производиться обработка (лингвистически сформулированных) экспертных знаний;

**Использование нечеткого управления не рекомендуется, если:**

– приемлемый результат может быть получен с помощью общей теории управления

– уже существует формализованная и адекватная математическая модель;

– проблема не разрешима.

**Существующие конструкторы ЭС**

**FuziCalc** фирмы FuziWare – это первая в мире электронная таблица, позволяющая работать как с точными числовыми значениями, так и с приблизительными, "нечеткими" величинами.

Если в процессе вычислений вы использовали нечеткие величины, результат также будет иметь вид функции распределения. Однако в любом случае результат будет получен и он будет точнее и достовернее, чем при использовании любых других доступных методов.

**iThink** – это программное обеспечение, которое позволяет создавать экспертные системы без необходимости знания языков программирования (рис. 3). С помощью iThink можно создавать блок-схемы, которые описывают логику принятия решений в экспертной системе, задавать правила и факты, а также проводить тестирование и отладку созданной системы. – предоставит Вам принципиально новые возможности, которые выходят далеко за рамки разработки стандартных форм документов.

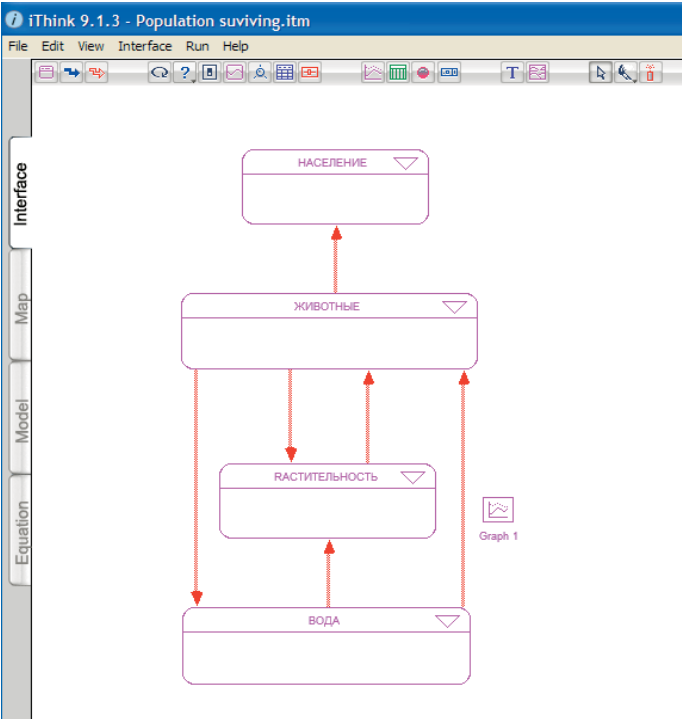


Рис.3 - Схема имитационной модели в режиме Interface (Фрейма)

В начале 90-х пакет iThink стал признанным стандартом структурного моделирования на Западе. Он широко используется в интеллектуальных центрах корпораций, банках, правительственных структурах и проектно –исследовательских учреждениях.

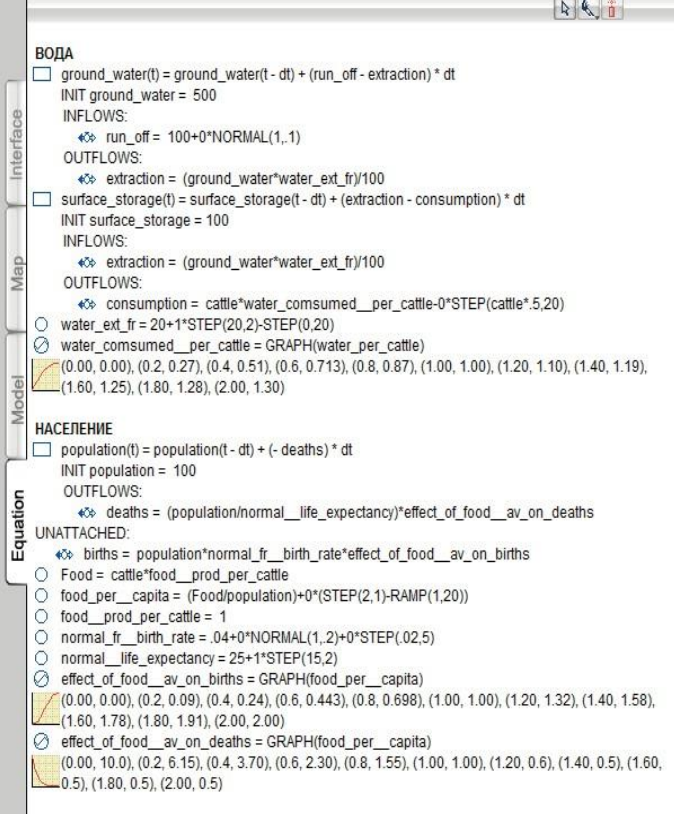


Рис.4 - Режим Уравнения модели для подмоделей ВОДА и НАСЕЛЕНИЕ

С помощью iThink решались разнообразные задачи, начиная от анализа причин разрушения дамбы в Юго – Восточной Азии в 1989г. и кончая обслуживанием и распределением пациентов, поступающих в приемный покой клиники. Однако в наибольшей степени ему органичны так называемые “потоковые” задачи. Они охватывают весьма широкую группу ситуаций, встречающихся в повседневной жизни предпринимателей, менеджеров и экспертов в области бизнес-планирования. Дело в том, что большинство развивающихся во времени явлений можно представить как потоковые процессы.

Пакет ориентирован на широкую группу пользователей – от руководителей, решающих сложные управленческие проблемы, до специалистов в области ценных бумаг, консультационных компаний, индивидуальных предпринимателей и исследователей. Однако, для новичков пакет может оказаться достаточно сложным в освоении, т.к. имеет много функций, которые будут непонятны для неопытного пользователя. iThink может быть неэффективным для больших и сложных моделей, так как имеет проблемы с производительностью при обработке больших объемов данных.

**Knowledge Craft** – это программное обеспечение, которое предоставляет инструменты для создания и управления экспертными системами. Он включает в себя редакторы правил и фактов, инструменты для тестирования и отладки системы, а также возможность интеграции с другими приложениями. Knowledge Craft также поддерживает различные форматы данных, что позволяет импортировать и экспортировать знания в разных форматах.

Как и любое программное обеспечение, пакет Knowledge Craft имеет свои недостатки. Один из них – это его высокая стоимость, которая может быть слишком высокой для некоторых пользователей. Кроме того, Knowledge Craft может быть сложным в использовании для новичков в области экспертных систем, так как он предоставляет множество возможностей и инструментов, которые могут быть непонятными для начинающих. Еще одним недостатком может быть ограниченность в выборе языков программирования, которые поддерживаются пакетом.

**FLEX** (Flexible Expert System) – это пакет экспертных систем, разработанный компанией Texas Instruments в 1980-х годах. Он был предназначен для создания экспертных систем на базе микропроцессоров и микроконтроллеров, и был широко использован в промышленности и научных исследованиях.

Хотя пакет экспертных систем FLEX был популярен в прошлом, он имеет некоторые недостатки и ограничения, которые могут сделать его менее привлекательным для использования в настоящее время.

Во-первых, язык программирования FLEX может быть сложным для понимания и использования для новичков. Это может затруднить процесс создания экспертных систем и требовать дополнительного времени и ресурсов для обучения.

Кроме того, FLEX не имеет широкой поддержки и сообщества разработчиков, что может ограничить доступность ресурсов и инструментов для разработки и отладки экспертных систем.

Также стоит отметить, что FLEX не обновлялся уже довольно давно, что означает, что он может быть менее совместимым с современными технологиями и системами.

Наконец, FLEX может быть менее гибким и универсальным, чем некоторые более современные пакеты экспертных систем, что может ограничить его применение в некоторых отраслях и проектах.

**Fuzzy Logic Model Builder** (FLMBuider) была разработана в 2005 году для обучения студентов Красноярского государственного технического университета (с 2007 г. Сибирского федерального университета) специальностей информационного профиля, не специализирующихся на разработке программного обеспечения. В её основе был положен принцип композиционного вывода, входы обрабатывались методом нечеткой логики (НЛ) (использован критерий минимума).

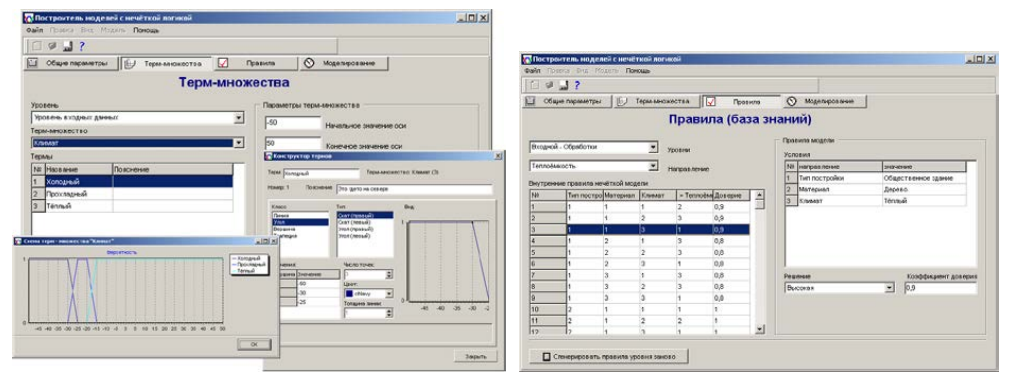


Рис. 5. Примеры экранных форм второго и третьего этапов описания

модели (задача подбора типа утеплителя)

Модуль работы с моделями был реализован в виде модуля на языке программирования Object Pascal (FLM\_modul.pas), а интерфейсная часть в виде оконного приложения в среде Delphi. Создание ЭС происходило в режиме конструктора и производилось за 3 этапа (рис. 2):

– формирование архитектуры процесса рассуждений (формы дерева принятия решений);

– описание этапов принятия решений (включая конструирование характеристических функций для фазификации);

– заполнение базы знаний (набора продукционных правил). Результирующий файл с расширением \*.flm сохранялся в виде текстового файла (нотация конфигурационных фалов ini).

**1.2 Технологии внедрения ЭС**

***FLM BUILDER***

Созданную в приложении FLM\_Builder модель ЭС пользователь может интегрировать в произвольные приложения в среде Delphi: для этого ему необходимо скопировать файл flm в каталог нового проекта и туда же разместить FLM\_modul.pas, после чего загрузить модель и обратиться к функции её просчёта (вектор входных значений подается в виде аргументов, см. левую часть рис. 3).

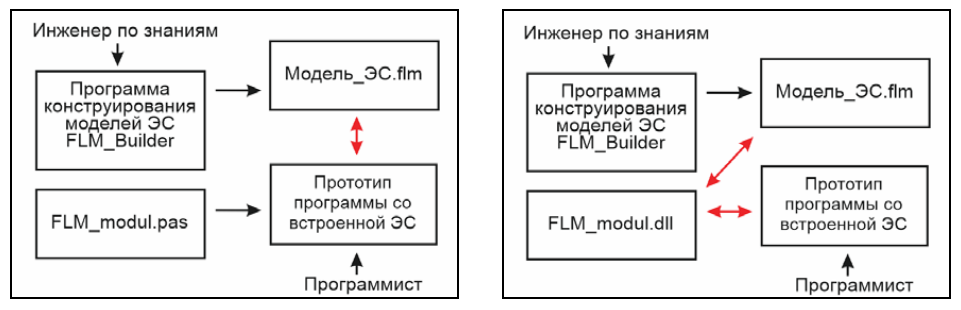


Рис. 6. Схема интеграции flm-файлов в пользовательские проекты на базе модуля FLM\_modul.pas (слева) и его dll версии (справа)

Таким образом, весь объём работ по интеграции flm-модели ограничивался 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, необходимого для интеграции flm – модели приведен ниже (три фактора на входе, два ответа на выходе для модели MyESModel.flm

uses FLM\_modul;

…

procedure(…)

var

MyES : TFuzzyModel;

ESIn : RealArray;

ESOut: StringArray;

begin

…

SetLength(ESIn,3);

MIn[0] := StrToFloat(Edit1.Text);

MIn[1] := StrToFloat(Edit1.Text);

MIn[2] := StrToFloat(Edit1.Text);

MyES := TFuzzyModel.Create;

MyES := LoadModelFile(‘MyESModel.flm’);

27

SetLength(MOut,2);

ESOut := CalcModel(MyES, ESIn);

…

MyES.Free;

end;

**1.3 Специфика современных web приложений**

Раньше создавались большие настольные системы, которые необходимо было развертывать на каждом ПК отдельно, создавались они преимущественно на языках C и C++, сейчас же в век интернет – технологий, популярность приобрели web – ориентированные языка – это JavaScript, Python

**Преимущества веб-приложений**

Веб-приложения – это приложения, которые работают через интернет в веб-браузере и не требуют установки на компьютер или мобильное устройство пользователя. Преимущества веб-приложений над другими типами приложений включают:

– Универсальность: веб-приложения могут работать на любой операционной системе и на любом устройстве с доступом к интернету, что делает их удобными для использования.

– Кроссплатформенность: разработчики могут создавать веб-приложения, которые могут работать на различных устройствах и операционных системах без необходимости создавать отдельные версии приложения для каждой платформы.

– Удобство использования: веб-приложения не требуют установки и обновления, пользователи могут использовать их непосредственно в веб-браузере.

– Более простая разработка: создание веб-приложений проще, чем создание настольных или мобильных приложений, так как они не требуют специфических знаний или инструментов для разработки.

– Более простое распространение и обновление: веб-приложения можно обновлять без необходимости загрузки и установки новых версий приложения, что делает их более удобными для пользователей.

– Более низкие затраты: разработка веб-приложений может быть более экономически эффективной, поскольку нет необходимости создавать отдельные версии приложения для разных операционных систем и платформ.

– Большая доступность для пользователей: веб-приложения могут быть легко распространены и доступны для пользователей на всем мире без необходимости загрузки и установки дополнительных приложений.

– Большая масштабируемость: веб-приложения могут легко масштабироваться для обслуживания большого количества пользователей и данных.

– Более высокий уровень безопасности: веб-приложения могут быть более безопасными, так как они работают на удаленном сервере, что делает их менее уязвимыми для взлома или кражи данных, чем приложения, работающие локально на устройстве пользователя.

– Более высокая скорость разработки и выхода на рынок: создание веб-приложений может быть быстрее, чем создание приложений для определенной платформы, так как разработчики могут использовать уже существующие технологии и инфраструктуру.

– Возможность интеграции с другими системами: веб-приложения могут быть легко интегрированы с другими системами и сервисами через API, что делает их более гибкими и расширяемыми.

– Легкость сопровождения и поддержки: веб-приложения могут быть легко поддерживаемы и обновляемы, так как изменения могут быть внесены на серверной стороне, без необходимости установки обновлений на устройствах пользователей.

В целом, веб-приложения имеют множество преимуществ по сравнению с другими типами приложений. Они удобны в использовании, в ходе разработки вам не придется создавать отдельные приложения для разных операционных систем – они работают одинаково в любых браузерах: Internet Explorer, Opera, Safari, Google Chrome и т.п. легко масштабируемы, более безопасны и могут быть доступны на любом устройстве с доступом к интернету. Эти преимущества делают веб – приложения одним из наиболее перспективных и востребованных направлений в IT-индустрии.

**2.1**

ЭС, созданная в системе FLMBuilder 4b. сохраняется в нотации xml – документа, имеющего следующую структуру:

# Модель ЭС в нотации xml

<FLMMODEL>

# Блок данных конфигурации

[-](file:///C:\Users\qwerty\Documents\PythonProjectsBook\flmdata.xml) <SETUP>

  <NAME> Название модели</NAME>

  <CAPTION>Описание модели</CAPTION>

  <COMMENTS>Комментарий</COMMENTS>

  <CREATEDATE>Дата создания</CREATEDATE>

  <MODDIFDATE>Дата последнего редактирования</MODDIFDATE>

  <USERNAME>Автор модели</USERNAME>

  <INPUTLEVELTERMSET>Количество терм-множеств на уровне входных данных</INPUTLEVELTERMSET>

  <PROCESSINGLEVELTERMSET> Количество терм-множеств на уровне обработки </PROCESSINGLEVELTERMSET>

  <OUTPUTLEVELTERMSET> Количество терм-множеств на уровне вывода </OUTPUTLEVELTERMSET>

  </SETUP>

# Блок данных о уровне ввода данных

[-](file:///C:\Users\qwerty\Documents\PythonProjectsBook\flmdata.xml) <INPUTLEVEL>

# Блок данных о первом терм множестве уровня ввода данных

[-](file:///C:\Users\qwerty\Documents\PythonProjectsBook\flmdata.xml) <SET\_1>

  <NAME>Название переменной первого терм – множества уровня ввода</NAME>

  <CAPTION> Название первого терм – множества </CAPTION>

  <TERMCOUNT>Количество термов в множестве</TERMCOUNT>

  <TERM1>Название первого терма</TERM1>

  <TERM2> Название второго терма </TERM2>

  <TERMN> Название N терма </TERMN>

</SET\_1>

  </INPUTLEVEL>

# Блок данных о уровнях обработки данных

[-](file:///C:\Users\qwerty\Documents\PythonProjectsBook\flmdata.xml) <PROCESSINGLEVEL>

# Блок данных о первом терм множестве уровня обработки

[-](file:///C:\Users\qwerty\Documents\PythonProjectsBook\flmdata.xml) <SET\_1>

  <NAME> Название переменной первого терм – множества уровня обработки </NAME>

  <CAPTION> Название первого терм – множества </CAPTION>

  <TERMCOUNT> Количество термов в множестве </TERMCOUNT>

  <TERM1> Название первого терма </TERM1>

  <TERM2> Название второго терма </TERM2>

  <TERMN> Название N терма </TERMN>

  </SET\_1>

  </PROCESSINGLEVEL>

# Блок данных о уровне вывода

[-](file:///C:\Users\qwerty\Documents\PythonProjectsBook\flmdata.xml) <OUTPUTLEVEL>

# Блок данных о первом терм множестве уровня вывода

[-](file:///C:\Users\qwerty\Documents\PythonProjectsBook\flmdata.xml) <SET\_1>

  <NAME> Название переменной первого терм – множества уровня вывода </NAME>

  <CAPTION> Название первого терм – множества </CAPTION>

  <TERMCOUNT> Количество термов в множестве </TERMCOUNT>

  <TERM1> Название первого терма </TERM1>

  <TERM2> Название второго терма </TERM2>

  <TERMN> Название N терма </TERMN>

  </SET\_1>

  </OUTPUTLEVEL>

# Блок данных о правилах

[-](file:///C:\Users\qwerty\Documents\PythonProjectsBook\flmdata.xml) <RULES>

  <RULES\_LEVELS\_COUNT>Правила между сколькими уровнями</RULES\_LEVELS\_COUNT>

  <RULES\_LEVELS1\_COUNT>Сколько подуровней правил на уровне 1 </RULES\_LEVELS1\_COUNT>

  <RULES\_LEVELS2\_COUNT> Сколько подуровней правил на уровне 2 </RULES\_LEVELS2\_COUNT>

# Блок данных о первом уровне правил

[-](file:///C:\Users\qwerty\Documents\PythonProjectsBook\flmdata.xml) <RULES\_LEVEL\_1> # Правила уровня 1

[-](file:///C:\Users\qwerty\Documents\PythonProjectsBook\flmdata.xml) <RULES\_LEVEL\_1\_1> # Правила подуровня\_1 уровня\_1

  <RULES\_COUNT>Сколько правил на подуровне 1\_1</RULES\_COUNT>

# Пример правила: условие1\_условие2\_условие3..решение\_коэффициент уверенности

  <RULE\_1>1\_1\_1..2\_1</RULE\_1>

  <RULE\_2>1\_1\_2..3\_0.9</RULE\_2>

  <RULE\_3>1\_1\_3..1\_0.9</RULE\_3>

  <RULE\_4>1\_2\_1..3\_0.8</RULE\_4>

  <RULE\_5>1\_2\_2..3\_0.8</RULE\_5>

  </RULES\_LEVEL\_1\_1>

  </RULES\_LEVEL\_1>

# Блок данных о втором уровне правил

[-](file:///C:\Users\qwerty\Documents\PythonProjectsBook\flmdata.xml) <RULES\_LEVEL\_2>

[-](file:///C:\Users\qwerty\Documents\PythonProjectsBook\flmdata.xml) <RULES\_LEVEL\_2\_1>

  <RULES\_COUNT> Сколько правил на подуровне 2\_1</RULES\_COUNT>

  <RULE\_1>1\_1..8\_1</RULE\_1>

  <RULE\_2>1\_2..5\_0.8</RULE\_2>

  <RULE\_3>1\_3..6\_0.9</RULE\_3>

  </RULES\_LEVEL\_2\_1>

  </RULES\_LEVEL\_2>

  </RULES>

# Блок данных о связях между уровнями

[-](file:///C:\Users\qwerty\Documents\PythonProjectsBook\flmdata.xml) <CONNECTIONS> # Тут задаются связи между уровнями

  <LEVELS\_COUNT>Сколько уровней связей</LEVELS\_COUNT>

# Блок данных о связях между первым и вторым уровнями

[-](file:///C:\Users\qwerty\Documents\PythonProjectsBook\flmdata.xml) <LEVEL\_1>

  <CONNECTIONS\_COUNT> Сколько связей на уровне 1 </CONNECTIONS\_COUNT>

  <TERM\_1\_1\_1>1\_1</TERM\_1\_1\_1>

  <TERM\_1\_1\_2>1\_1</TERM\_1\_1\_2>

# Пример связи: 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 )

  <TERM\_1\_1\_3>1\_1</TERM\_1\_1\_3>

  </LEVEL\_1>

# Блок данных о связях между вторым и третьим уровнями

[-](file:///C:\Users\qwerty\Documents\PythonProjectsBook\flmdata.xml) <LEVEL\_2>

  <CONNECTIONS\_COUNT> Сколько связей на уровне 2 </CONNECTIONS\_COUNT>

  <TERM\_2\_1\_1>1</TERM\_2\_1\_1>

  <TERM\_2\_1\_2>1</TERM\_2\_1\_2>

# Пример связи: терм из уровня 2 (обработки)\_1(1 терм-множество)\_3(3 терм) соединен с терм – множеством 1 уровня 3 (вывода )

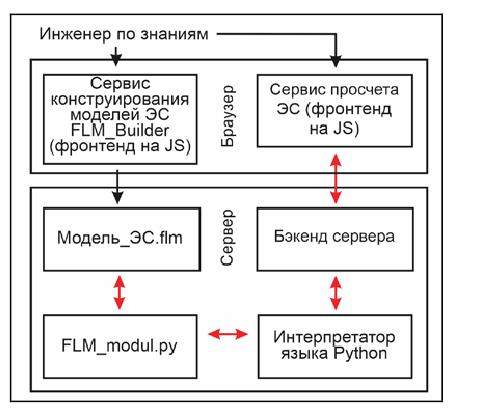
  <TERM\_2\_1\_3>1</TERM\_2\_1\_3>

  </LEVEL\_2>

  </CONNECTIONS>

  </FLMMODEL>

**2.2 Архитектура интеграции**

****

**2.3 Визуальное конструирование ЭС**

**Визуальный конструктор**

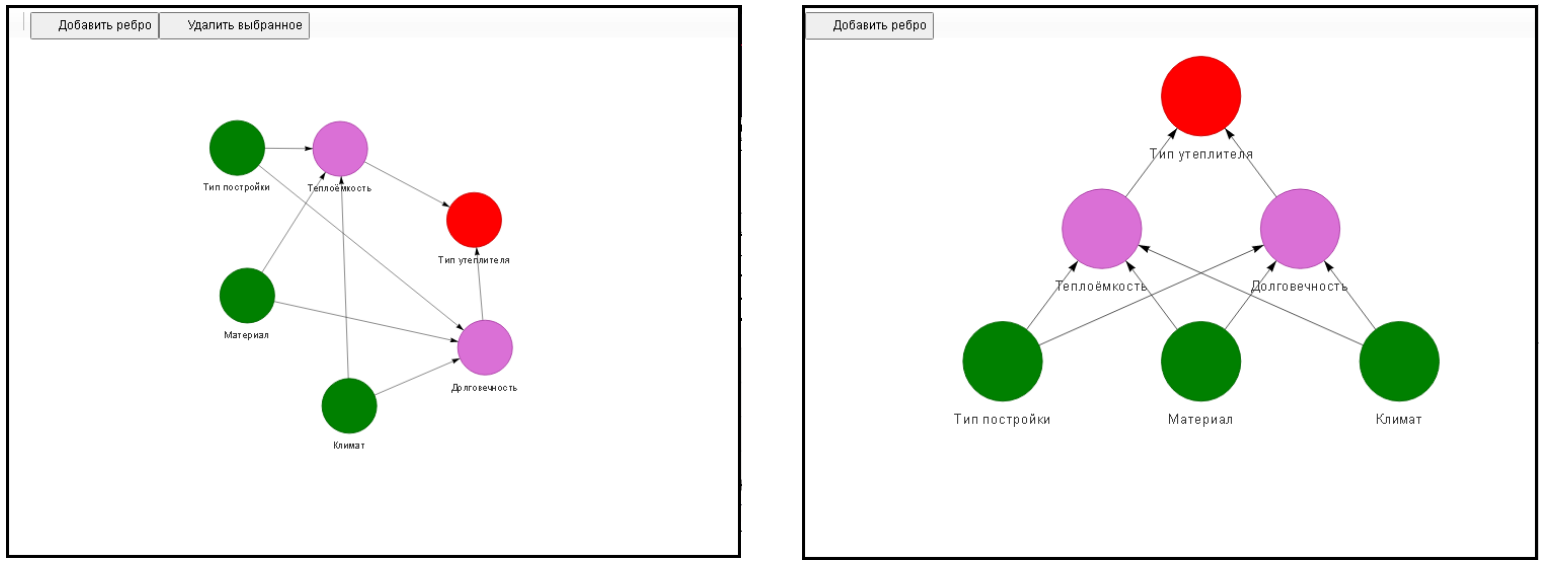
«Окно конструирования» – в этом окне происходит графическое конструирование структуры модели (дерева принятия решений). Двойным кликом левой кнопки мыши происходит создание нового терма – множества, для того чтобы соединить его с другим терм – множеством, нужно зайти вменю «Добавить ребро», после чего щелкнуть на терм – множестве, от которого необходимо провести связь и не отпуская кнопки мыши –протянуть связь к необходимому терм – множеству (рис. 1, слева). После создания ЭС, программа автоматически разбивает модель на несколько слоёв, где зеленое ТМ – это множества входа, фиолетовое – множества обработки, красное – множества вывода, бежевое – ТМ не имеющее связи с другими множествами (не включенные в процесс вывода). Каждый уровень обработки, а также уровень вывода, содержит ссылку на набор правил из БЗ, которые описывают стратегию логического вывода при просчёте модели. Так, первый уровень содержит упорядоченные терм – множества, проходя через которые входные данные фазифицируются, т. е. обрабатываются механизмом нечёткой логики. Другие, следующие за входным, слои могут содержать неупорядоченные терм – множества. Для их описания не требуется определять характеристические функции, так как в БЗ уже содержатся коэффициенты доверия, которые внёс эксперт при составлении модели и формировании базы знаний.

Рис. 7. Внешний вид «Окна конструирования»: исходное дерево принятия решений

(слева) и отсортировнное (справа)

**Окно заполнения Терм-множества**

«Окно настройки ТМ» – это окно вызывается при нажатии на любое из созданных ТМ в окне конструирования. В этом окне пользователю предлагается настроить внутреннюю структуру модели: модифицировать состав и содержание терм – множеств, настраивать функции принадлежности включённых в них термов. (рис. 8). Для формализации задачи, требуется исследовать предметную область и выявить те показатели, которые пойдут на вход модели или будут внутри неё. Эти показатели могут быть чёткими (однозначно определёнными), так и нечёткими (заданным неоднозначно в виде характеристически функций).

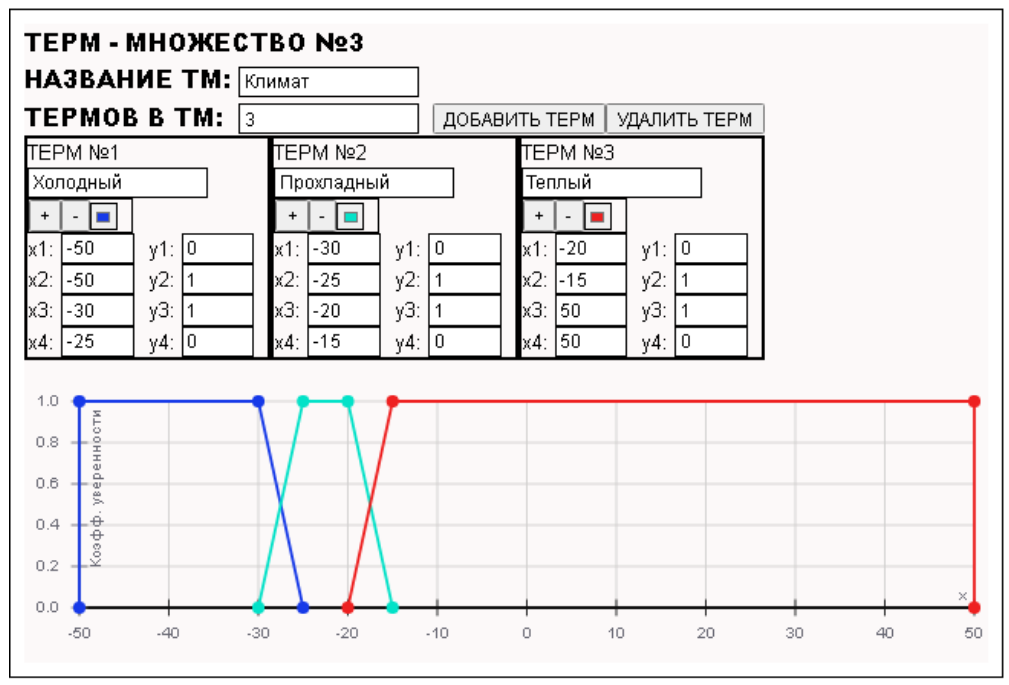


Рис. 8. Внешний вид «Окна настройки ТМ» для ТМ «Климат»

**Окно заполнения правил**

«Правила» – окно, вызываемое при нажатии кнопки «Правила», которая находится в окне настройки ТМ уровня обработки и уровня вывода, в этом окне пользователю (эксперту) предлагается сформировать базу знаний для создаваемой модели. Для каждой комбинации термов на слое формируется вывод, дополненный коэффициентом уверенности (рис. 9). Набор правил автоматически генерируется после создания всех необходимых связей в модели и инициализируется первым значением в терм – множестве с коэффициентом уверенности «1». Для изменения правил необходимо выбрать

нужный элемент из выпадающего списка, а также указать коэффициент уверенности в диапазоне от «0» до «1».



Рис. 9. Внешний вид окна «Правила»