АНАЛИЗ ПОДХОДОВ К ПОСТРОЕНИЮ ЭКСПЕРТНЫХ ИНФОРМАЦИОННЫХ СИСТЕМ ДЛЯ ПОДДЕРЖКИ ОПЕРАТИВНОГО ПЕРСОНАЛА АТОМНЫХ ЭЛЕКТРОСТАНЦИЙ

Р.В. Левицкий, *Московский технический университет связи и информатики, rlpostbox@yandex.ru.*

УДК 004.891.2

Аннотация. Рассматриваются основные принципы экспертных систем; задачи, которые могут быть решены с их помощью; принципы сбора и формализации знаний; основные модели представления данных. Производится обзор современных инструментальных средств для практической разработки подобных систем. Определяется подход к разработке опытного прототипа экспертной системы для поддержки оперативного персонала АЭС.

Ключевые слова: искусственный интеллект; экспертная система; модель; логический вывод; продукционная модель; фрейм; семантические сети; база знаний; прототип.

ANALYSIS OF APPROACHES TO THE CONSTRUCTION OF EXPERT INFORMATION SYSTEMS TO SUPPORT THE OPERATIONAL STAFF OF NUCLEAR POWER PLANTS

Roman Levitskiy, *Moscow technical university of communications and informatics.*

Annotation. The basic principles of expert systems are considered. Tasks that can be solved with their help. The principles of collecting and formalizing knowledge. Basic data presentation models. A review of modern tools for the practical development of such systems is made. An approach to the development of an experimental prototype of an expert system to support the operating personnel of a nuclear power plant is determined.

Keywords: artificial intelligence; expert system; model; inference; production model; frame; semantic networks; knowledge base; python; prototype.

Введение

Очень часто при промышленной эксплуатации стратегически важных объектов, таких как АЭС, оперативному персоналу необходимо, действуя в ограниченных временных рамках, анализировать большие объемы данных и следовать многочисленным служебным нормативным документам. На сегодняшний день для поддержки оперативного персонала возможно использовать вспомогательные экспертные системы (ЭС). Их история насчитывает более 80 лет, но тем не менее в практике они встречаются довольно редко из-за специфичности и высокой цены.

Цель данной работы заключается в изучении и описании задач, решаемых экспертными системами, основных подходов, используемых в них, а также в определении наиболее подходящего прототипа экспертной системы для поддержки оперативного персонала.

Постановка задачи исследования

Проводимое исследование нацелено на обобщение данных по ряду современных общедоступных экспертных систем. Для реализации указанной задачи предлагается рассмотреть следующие вопросы [1]:

- цели и задачи экспертных систем;
- требования к инструментам для реализации экспертных систем;
- возможности для создания опытного прототипа.

Обобщение указанных данных позволит найти оптимальный способ разработки архитектуры прототипа экспертной системы.

Задачи и отличительные признаки систем искусственного интеллекта

Экспертные системы (далее также – «ЭС») относятся к разделу информатики, изучающему искусственный интеллект (далее также – «ИИ») – Научные направления информатики (рис. 1). Это направление также связано с такими науками как психология и нейрофизиология, которые исследуют и моделируют интеллектуальные процессы мозга [2, 3].

Рисунок 1

Перед искусственным интеллектом ставят цели:

- исследовать и моделировать интеллектуальные процессы мозга;
- решать интеллектуальные задачи более эффективно, чем это делает мозг.

Согласно классификации американских ученых-психологов Никерсона, Перкинсона и Смита к интеллектуальным задачам относятся:

- способность классифицировать образы;
- способность к адаптивному изменению поведения (к обучению);

- способность к дедуктивному мышлению (делать выводы из имеющихся посылок);
- способность к индуктивному мышлению (к обобщению с возможностью порождать новые знания);
- способность разрабатывать и использовать концептуальные модели;
- способность к пониманию.

Искусственный интеллект с точки зрения информатики исследуется как преобразователь процессов обработки информации, использующий знания, введенные в него.

Разработка искусственного интеллекта происходит в двух направлениях [1]:

1) к первому относят разработку на основе компьютерной парадигмы, которая заключается в создании компьютерных программ, способных решать интеллектуальные задачи. Подход основан на двух основных идеях: первая заключается в том, что мозг – это аналог машины Тьюринга, только более сложный, вторая – в том, что все интеллектуальные задачи поддаются компьютерному моделированию.
2) второй подход придерживается нейронной парадигмы, суть которой заключается в имитации нейронных сетей мозга, в которых будут решаться интеллектуальные задачи. В основе указанной парадигмы лежит идея о необходимости уделить основное внимание взаимодействию нейронов. Из-за своего сложного строения они сильно упрощаются и во внимание принимаются только синапсы, аксоны и пороги срабатывания. Наиболее значимых успехов в данном направлении удалось добиться в задачах, связанных с распознаванием образов, где образ – это класс изображений, а распознавание – это процесс отнесения изображения к определенному классу.

Сложности на пути создания искусственного разума

Приведенные ранее идеи компьютерной парадигмы спустя некоторое время с момента их создания были поставлены под сомнение. Раньше было принято считать, что мозг – это просто большой компьютер, но со временем обнаружились серьезные функциональные различия между ними. У человека имеется две системы хранения и обработки информации: образная и символьно-логическая. Образная система у человека врожденная, а символьно-логическая возникает в процессе обучения. У компьютера имеется только символьная система, а все остальное – лишь ее интерпретация для различных типов данных.

Идея о том, что можно смоделировать все интеллектуальные процессы человека, оказалась несостоятельной. Программы, решающие интеллектуальные задачи, не моделируют процесс их решения человеком, их работа строится на алгоритмах и компьютерных структурах. Проблема создания «идеального» искусственного разума далека от своего решения.

Гораздо больших успехов ИИ достиг в моделировании отдельных интеллектуальных способностей и связанных с ними интеллектуальных задач. В компьютерной парадигме произошел отказ от стремления моделировать интеллектуальные процессы, а основное внимание было сосредоточено на результате. Стало важно, чтобы программа решала задачу, а не то, что она это делает не так, как человек.

Эффективное решение интеллектуальных задач

Тем не менее, несмотря на все ранее сказанное, у человека и машины есть общий класс задач, которые они решают одинаково – это алгоритмические задачи.

Первоначально в области искусственного интеллекта при решении интеллектуальных задач сложился подход, называемый лабиринтной моделью или поиском в пространстве рис. 2 (представление пространства состояний). Этот подход представляет процесс решения задачи как последовательный переход из начального состояния в целевое через ряд промежуточных состояний. Формально под пространством состояний подразумевается ориентированный граф, вершины которого – это состояния, а ребра – возможные переходы из одного состояния в другое. Поиск в пространстве состояний формально можно представить, как алгоритмическую задачу, ее решение заключается в полном переборе всех возможных переходов [4].

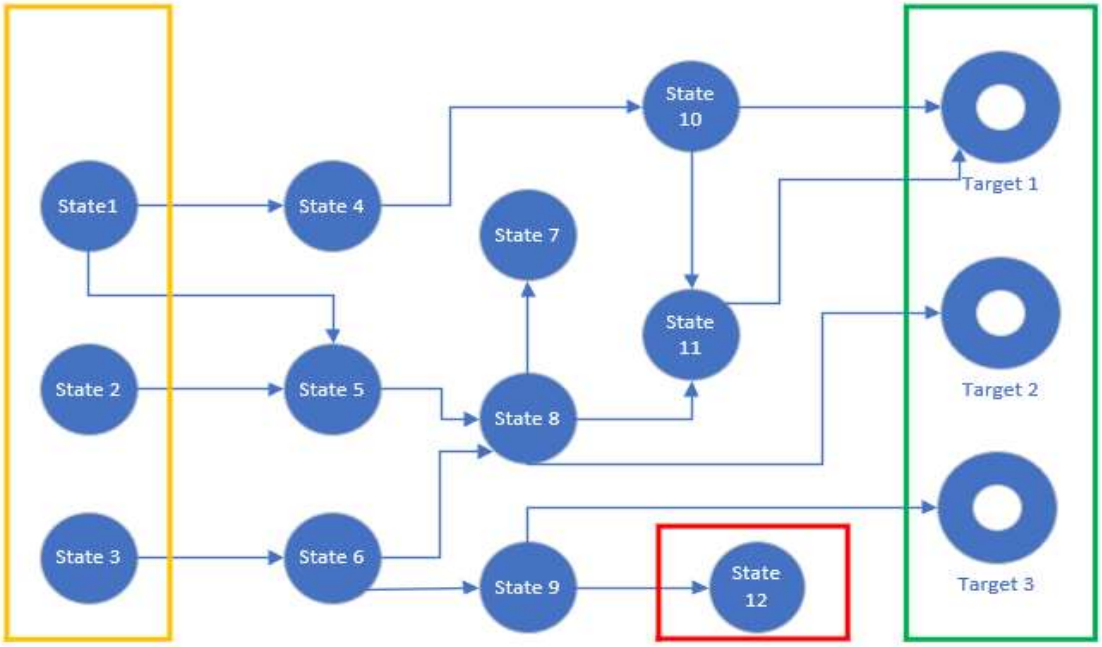

Рисунок 2

Существует два основных алгоритма полного перебора: поиск в глубину и поиск в ширину. Но для сложных задач эти методы не всегда применимы и приходится прибегать к специальным методам сокращения объема перебора – эвристическим методам. Задачи, решаемые с помощью эвристик, являются интеллектуальными. Эвристики зависят от контекста задачи. Главная проблема

при создании эвристических интеллектуальных программ – это поиск формализации, который бы позволил автоматизировать получение обработанных данных, пригодных для эвристик.

Один из распространенных методов получения эвристик состоит в построении оценочной функции, которая грубо оценивает каждое состояние с точки зрения достижимости цели. Такие функции широко используются, например, в шахматных программах [5].

Представление знаний в экспертных системах

Этапы разработки экспертных систем

Системы искусственного интеллекта или интеллектуальные системы – это системы, основанные на знаниях. В их разработке принято выделять следующие этапы:

- идентификация проблемы;
- извлечение знаний;
- структурирование знаний;
- решение задачи;
- реализация (создание прототипа);
- тестирование.

Знания принято разделять на глубинные и поверхностные, где под глубинными понимаются фундаментальные знания, относящиеся к неким научным дисциплинам, а под поверхностными понимают эмпирические ассоциации и причинно-следственные связи. Экспертные системы используют преимущественно поверхностные знания.

Способы получения знаний и их структурирование

В процессе получения знаний задействованы эксперты и инженеры по знаниям. В задачу инженеров входит создание условий и методик, позволяющих получить от экспертов формализованные знания. Инженеры по знаниям могут прибегать к таким средствам как:

- просьба эксперта прочитать лекцию;
- анкетирование;
- интервью;
- свободный диалог;
- круглый стол;
- мозговой штурм;
- экспертные игры.

Полученные знания нуждаются в обработке. Этот этап называется структурированием знаний, при данном подходе работа разбивается на восемь слоев анализа:

- стратегический;

- организационный;
- концептуальный;
- функциональный;
- пространственный;
- временной;
- каузальный;
- экономический.

Основными слоями при структурировании данных являются концептуальный, функциональный и каузальный анализ.

Модели представления знаний в экспертных системах

Экспертная система делает вывод, опираясь на знания, которые в нее были вложены на этапе формирования. Сами знания имеют определенную структуру, в виде которой они хранятся в базе знаний [6].

В экспертных системах принято различать четыре модели представления данных [2]:

- логическая модель;
- продукционная модель;
- модель семантической сети;
- модель фрейма.

Дадим краткое описание каждой из этих моделей.

Логическая модель использует язык логики предикатов и включает в себя их имена, функции, логические связки, кванторы и правила построения логических формул.

Продукционная модель хранит выражения вида «если УСЛОВИЕ, то ДЕЙСТВИЕ» или «если УСЛОВИЕ, то ДЕЙСТВИЕ иначе ДЕЙСТВИЕ» [9].

Семантические сети выражают отношения типа «класс-суперкласс», «элемент-класс» и являются по своей природе иерархическими структурами, поддерживающими неограниченную вложенность внутренних состояний.

Фреймовую модель можно представить, как реляционную таблицу данных с привязанными к ней функциями. Фрейм состоит из слотов и демонов. Слоты – атрибуты объекта, имеющие тип данных. Демоны – функции, автоматически вызывающиеся для исполнения при определенных условиях.

Возможна ситуация, когда представление знаний выбрано сразу после идентификации проблемы. В этом случае структура базы знаний появляется раньше других компонентов и это позволяет совместить этапы получения и систематизации знаний с помощью программных средств. Такой подход принято называть прямым приобретением знаний. Примером системы, обеспечивающей возможность прямого приобретения знаний, является система *SIMER,* разработанная в Институте программных систем РАН, располагающейся в Переславле-Залесском [3].

Современные экспертные системы

Классическая архитектура экспертной системы представлена на рис. 3. Здесь она представлена в упрощенном виде (табл. 1 – **типы инструментальных средств для разработки** экспертных систем) [7].

Рисунок 3

Таблица 1.

№ п/п	Наименование	Тип	Описание
1	*LISP*	Язык общего программирования	Язык, ориентированный на работу с символьной информацией. Разработан под руководством Дж. Маккарти в 1960 г.
2	*PROLOG*	Язык логического программирования	Язык, ориентированный на логическое представление знаний. Разработан в 1970-е гг.
3	*OPS5*	Специальный язык разработки ЭС	Синтаксис приближен к *LISP*. Разработан Чарльзом Форге в 1970-х гг.
4	*FLAVORS*	Специальный язык разработки ЭС	Объектное расширение языка *LISP*. Разработан Говардом Кэнноном в *MIT* в 1970-х гг.
5	*CLIPS (*нач. 80-х*)*	Специальный язык разработки ЭС	Си подобный язык. Разрабатывался Чарльз Форджи создателем *OPS*5. Создан в 1985 гг.
6	*LOOPS*	Оболочка ЭС	Объектный язык программирования.
7	*EMYCIN*	Оболочка ЭС	Каркас экспертной системы *MYCIN* из которой извлекли все содержание.

№ п/п	Наименование	Тип	Описание
8	*ART*Enterprise*	Среда разработки ЭС	Среда разработки прикладных программ широкого применения, объединяющая в себе правила, объектно-ориентированную систему. Разработана в 1980-х гг.
9	*KEE*	Среда разработки ЭС	Основанный на представлении знаний в виде фреймов, написан на языке *LISP*. Произведена *IntelliCorp,* выпущен в 1983 г.
10	*G2*	Среда разработки ЭС	Объектно-ориентированная среда для разработки и сопровождения приложений реального времени, использующих базы данных. Разработана фирмой *Gensym* в 1993 г.
11	*CLIPS*	Среда разработки ЭС	Общедоступный программный инструмент для построения экспертных систем. Разработан Чарльзом Форджи в 1985 г. Написан на языке Си.
12	*FuzzyCLIPS*	Среда разработки ЭС	Расширение оболочки экспертной системы *CLIPS*, поддерживающей применение нечеткой логики. Разработан группой интегрированного мышления Института информационных технологий Национального исследовательского совета Канады и широко распространялся в течение ряда лет.
13	Малая Экспертная Система 2.0	Среда разработки ЭС	Она предназначена для проведения консультации с пользователем в какой-либо прикладной области (на которую настроена загруженная база знаний) с целью определения вероятностей возможных исходов и использует для этого оценку правдоподобности некоторых предпосылок, получаемую от пользователя. Важным достоинством данной программы является возможность создания и применения собственной базы знаний.

Дадим краткое описание основных элементов архитектуры ЭС.

Интерфейс пользователя может быть организован через командную строку и графический интерфейс, который передает вопросы и текущее состояние пользователю и принимает от него команды.

База знаний – это ключевой компонент ЭС. Все выводы строятся на основании хранящихся там формализованных определенным образом правил [8].

Данные – это формализованная информация, на которую опирается модель представления данных при построении своих выводов.

Модель представления данных или «Решатель» – элемент, выполняющий главную роль по построению логических выводов, хранит внутри себя эвристические правила и алгоритмы решения конфликтных ситуаций при множественных результатах вывода.

Интеллектуальный редактор – это элемент или отдельная программа, способная редактировать, добавлять, удалять знания или выводить отчет о знаниях, хранящихся в ЭС.

В качестве примера приведем совмещенный формат представления знаний и правил в среде разработки ЭС «Малая Экспертная Система 2.0» [9].

Пример базы знаний:
"Определение домашнего питомца по признакам."

Вопросы:
Морда вытянутая?
Крылья есть?
На поглаживания по спине отвечает довольным урчанием?
Живет в аквариуме (или другом резервуаре с водой)?
Есть лапы?
При встрече с хозяином виляет хвостом?

Собака, 0.2, 1, 0.7, 0.5, 2, 0, 0.5, 3, 0.01, 0.5, 4, 0, 0.5, 6, 0.9, 0.05
Кошка, 0.2, 1, 0.1, 0.5, 2, 0, 0.5, 3, 0.95, 0, 4, 0, 0.5
Попугай, 0.1, 2,1, 0.3, 4, 0, 0.5
Рыбка, 0.1, 2, 0, 0.5, 4, 1, 0.1, 5, 0, 0.5
Тритон, 0.1, 2, 0, 0.5, 4, 1, 0.2, 5, 1, 0.5

База знаний представляет собой текстовый файл (который в дальнейшем может быть зашифрован), включающий три секции со следующей структурой [15]:

- описание базы знаний, имя автора, комментарий и т.п.;
- свидетельство № 0 (любой текст не более 1000 символов, заканчивающийся переносом строки);
- описание правила вывода.

Последняя секция требует более подробного рассмотрения [10].

В ней перечисляются правила вывода, каждое из которых задается в отдельной строке, перечисление заканчивается с концом файла. В начале описания правила вывода задается исход, вероятность которого меняется в соответствии с данным правилом. Это текст, включающий любые символы, кроме запятых. После запятой указывается априорная вероятность данного исхода (*P*), т.е. вероятность исхода в случае отсутствия дополнительной информации. После этого через запятую идет ряд повторяющихся полей из трех элементов: первый элемент (*i*) – это номер соответствующего вопроса (симптома, свидетельства); следующие два элемента (*Ру=Р*(*Е Н*) и *Рn=Р(Е*/*неН*)) – вероятности получения ответа «Да» на этот вопрос, если возможный исход верен и неверен соответственно. Эти данные указываются для каждого вопроса, связанного с данным исходом.

Классификация экспертных систем

Развитие экспертных систем прошло долгий путь от формирования основных принципов и подходов, которые были описаны ранее, до современных программных комплексов, позволяющих сократить трудоемкость и сделать подобные системы более доступными.

В тоже время укрупнение программных комплексов влечет за собой стандартизацию, которая может накладывать ограничения. Также особенностью, связанной с большими системами, является их высокая цена – доступно не так много бесплатных инструментальных средств. Это связано со сложностью разработки подобных систем, их специфичностью и высоким порогом вхождения как для разработчиков, так и пользователей.

В настоящее время инструментальные средства проектирования экспертных систем можно разделить на следующие типы [1]:

- языки программирования общего назначения;
- языки программирования, предназначенные для экспертных систем;
- оболочки экспертных систем;
- многофункциональные интегрированные среды, поддерживающие разработку экспертных систем.

На сегодняшний день доступны различные виды инструментов для разработки экспертных систем, правда некоторые из них уже можно считать устаревшими, так как они были разработаны более двадцати лет назад [11].

Выбор базы знаний для экспертной системы

Первоначально термин «база знаний» использовался для описания одной из двух подсистем экспертной системы. Система, основанная на знаниях, состоит из базы знаний, представляющей факты о мире и способов рассуждать об этих фактах, чтобы вывести новые факты или выявить несоответствия (рис. 3 – упрощенная базовая архитектура экспертных систем).

Термин «база знаний» был придуман, чтобы отличить эту форму хранилища знаний от более распространенного и широко используемого термина «база данных». В течение 1970-х гг. практически все крупные информационные системы управления хранили свои данные в иерархических или реляционных базах данных того или иного типа. На этом этапе истории информационных технологий различие между базой данных и базой знаний было четким и недвусмысленным [6].

База данных имела следующие свойства:

Плоские данные: данные обычно представлялись в табличном формате со строками или числами в каждом поле.

Несколько пользователей: обычная база данных, необходимая для поддержки одновременного входа в одни и те же данные нескольких пользователей или систем.

Транзакции. Важным требованием к базе данных было поддержание целостности и согласованности данных, к которым обращаются одновременно работающие пользователи. Это так называемые свойства *ACID*: атомарность, согласованность, изоляция и долговечность.

Большие долгоживущие данные: корпоративная база данных должна поддерживать не только тысячи, но и сотни тысяч или более строк данных. Такая база данных обычно необходима для сохранения после конкретных применений какой-либо отдельной программы; когда нужно было хранить данные в течение многих лет и десятилетий, а не в течение всего срока службы программы [12].

Первые системы, основанные на знаниях, имели потребности в данных, которые были противоположны этим требованиям к базам данных. Экспертная система требует структурированных данных. Не только таблицы с числами и строками, но и указатели на другие объекты, которые, в свою очередь, имеют дополнительные указатели. Идеальным представлением базы знаний является объектная модель (часто называемая онтологией в литературе по искусственному интеллекту) с классами, подклассами и экземплярами.

Ранние экспертные системы также мало нуждались в множестве пользователей или сложности, связанной с требованием транзакционных свойств данных. Данные ранних экспертных систем использовались для получения конкретного ответа, такого как медицинский диагноз, конструкция молекулы или реакция на чрезвычайную ситуацию. Как только решение проблемы было известно, не было критической потребности хранить большие объемы данных обратно в постоянное хранилище памяти. Более точным утверждением было бы то, что, учитывая доступные технологии, исследователи пошли на компромисс и отказались от этих возможностей, потому что они поняли, что они выходят за рамки того, чего можно было ожидать, и они могут разработать полезные решения нетривиальных проблем без них. Даже с самого начала более проницательные исследователи осознавали потенциальные преимущества возможности хранить, анализировать и повторно использовать знания [13].

Требования к объему базы знаний также были другими по сравнению с обычной базой данных. База знаний, необходимая для того, чтобы знать факты о мире. Например, чтобы представить утверждение, что «Все люди смертны». База данных обычно не может отражать эти общие знания, но вместо этого должна хранить информацию о тысячах таблиц, которые представляют информацию о конкретных людях. Представление о том, что все люди смертны, и возможность рассуждать о каждом конкретном человеке, что они смертны – это работа базы знаний. Представление, что Джордж, Мэри, Сэм, Дженна, Майк и сотни тысяч других клиентов – это люди с определенным возрастом, полом, адресом и т.д. – это работа для базы данных.

По мере того, как экспертные системы перешли от прототипов к системам, развернутым в корпоративных средах, требования к их хранилищам данных быстро начали пересекаться со стандартными требованиями к базам данных для нескольких распределенных пользователей с поддержкой транзакций. Первоначально спрос можно было наблюдать на двух разных, но конкурентных

рынках. Из сообществ *AI* и объектно-ориентированного программирования возникли объектно-ориентированные базы данных. Это были системы, разработанные с нуля для поддержки объектно-ориентированных возможностей, а также для поддержки стандартных служб баз данных. С другой стороны, крупные поставщики баз данных, такие как *Oracle,* добавили в свои продукты возможности, обеспечивающие поддержку требований базы знаний, таких как отношения и правила класса-подкласса (табл. 2 – **перечисление типов баз данных**).

Таблица 2.

Тип СУБД	Пример эффективного использования	Пример	Применение в ЭС
Реляционные	Нужна транзакционность; высокая нормализация; большая доля операций на вставку	*Oracle, PostreSQL, SQLite*	Применяются
Документно-ориентированные	Для хранения объектов в одной сущности, но с разной структурой; хранение структур на основе *JSON*	*MongoDB*	Не применяются
Ключ-значение	Задачи кэширования и брокеры сообщений	*Redis*	Не применяются
Графовые	Задачи подобные социальным сетям; системы оценок и рекомендаций	*Neo4j*	Активно применяются
Колоночные	Хранилища данных; выборки со сложными аналитическими вычислениями; количество строк в таблице превышает сотни миллионов.	*ClickHouse*	Не применяются
Объектно-ориентированные	Задачи представления, сохранения и передачи программных сущностей.	*DB4O*	Применяются

Рассмотрев типы современных баз данных и изучив их основные назначения в современных экспертных системах, делаем вывод, что для наших дальнейших работ перспективное направление имеет применение реляционных, графовых и объектно-ориентированных типов баз данных.

Дальнейшие действия по применению экспертных систем

Дальнейшие исследования в данной области знаний следует выстроить по принципу от простого к сложному. Так как нас в первую очередь интересует практическая польза от применения экспертных систем необходимо создать тестовый прототип на базе имеющихся инструментальных средств [14]. Для этих целей можно выбрать такую среду, как «Малая Экспертная Система 2.0» и подготовить на ней тестовую базу знаний, так как она достаточно проста в освоении и отладке. Однако к недостаткам можно отнести не сильно развитую архитектуру, пригодную для включения в ее состав более сложных программных комплексов. С этой точки зрения более перспективным выглядит применение среды *CLIPS*. Это бесплатный продукт с открытым исходным кодом, который написан на языке общего назначения (язык СИ). Данная среда может быть дополнена и скорректирована под задачи разрабатываемой архитектуры [15].

Использование *CLIPS* позволит создать удобную платформу для быстрого прототипирования экспертной системы в различных областях, однако для ее более удобного применения инженерами по знаниям необходимо отказаться от создания самой базы знаний непосредственно в данной среде. Более перспективным выглядит подход, при котором данные буду вестись в некой графовой базе данных с возможностью построения развитого отчета как в текстовом, так и в графическом виде. Отчет будет отражать структуру знаний в удобном для человека виде или текстовом формате. Также необходимо создать удобную графическую среду для ввода знаний. Все эти дополнения приведут к преобразованию среды *CLIPS* от простого пользовательского приложения к развитому «Клиент-Серверному» приложению с поддержкой протоколов взаимодействия с внешними системами [17].

Изучая современные платные решения в области экспертных систем следует обратить внимание на то, что почти все они хранят историю о выдаваемых заключениях. Это хорошая практика, которую также необходимо учесть в архитектуре будущего прототипа.

Язык программирования серверной части ЭС

Для разработки «Клиент-Серверного» приложения можно использовать практически любые современные языки программирования. Некоторые из наиболее популярных:

- ***JavaScript***
  - *Node.js*
- ***Java***
  - *Spring*
  - *Hibernate*
  - *JSF (JavaServer Faces)*
- ***Python***
  - *Django*
- ***C#***

- *ASP.NET MVC*
- *ASP.NET Core*

В работе используется язык *Python* и платформа *Django*. Эта платформа позволяет писать серверный код для динамических веб-страниц и веб-приложений, а также для программ командной строки. С помощью *Python* можно обойтись использованием одного языка программирования для разработки приложений вместо применения разных языков для работы над клиентской частью и серверной частью.

В дальнейшем в работе автором планируется применять *CPython* – это диалект языка *Python,* реализованный на языке Си. Это позволит быстро и эффективно дополнять код *CLIPS* новыми функциями в относительно короткие сроки.

Часть ЭС, работающей на стороне сервера, планируется реализовать на *Django* – это открытый некоммерческий фреймворк *Python,* поддерживающий все современные возможности разработки. Выбор языка *Python* для серверной части обеспечивает проекту ряд преимуществ:

- рост эффективности разработки благодаря использованию одного языка для фронта и бэкенда и возможности переиспользования кода;
- возможность использовать *pip* – очень большой пакетный менеджер.

Языки программирования клиентской части

Планируется создание графического пользовательского приложения, поддерживающего взаимодействия с сервером ЭС. Для прототипа будет также выбран язык ***Python.*** Он обеспечивает реагирование интерфейса на действия пользователя, обрабатывает клики мышкой, нажатия клавиш, перемещения курсора. Также посылает запросы на сервер, загружает данные, позволяет вводить сообщения и т.д.

Среда разработки приложения

В качестве среды разработки будет использован редактор кода *PyCharm Community Edition*, поскольку продукт поддерживает разработку *Python,* и считается лучшей из бесплатных *IDE*.

Заключение

Изучение инструментов позволяет понять, что именно необходимо для создания архитектуры информационной системы, в которой все вложенные данные будут корректно обработаны. В дальнейшем необходимо проработать содержание тестового набора правил и фактов для загрузки в образец экспертной системы, а также архитектуру для взаимодействия с внешними системами.

Литература

1. Хултен Дж. Разработка интеллектуальных систем / пер. с анг. В. С. Яценкова. – М.: ДМК Пресс, 2019. – 284 с.

2. Джексон П. Введение в экспертные системы. Пер. с англ. – М.: Издательский дом «Вильямс», 2020.
3. Поспелов Д.А. Искусственный интеллект: фантазия или наука? – М.: Радио и связь, 2015.
4. Грешилов А.А. Математические методы принятия решений: учеб. пособие (с рассчетными программами на оптическом диске). – 2-е изд., испр. и доп. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2014. – 647 с.
5. Тейз А., Грибомон П., Луи Ж. и др. Логический подход к искусственному интеллекту: от классической логики к логическому программированию. Пер. с франц. – М.: Мир, 2017.
6. Гаврилова Т.А., Хорошевский В.Ф. Базы знаний интеллектуальных систем. – Спб: Питер, 2016.
7. URL http://bourabai.ru/alg/expert22.htm
8. Робинсон Ян, Вебер Джим, Эифрем Эмиль. Графовые базы данных: новые возможности для работы со связанными данными / пер. с англ. РН. Рагимова; науч. ред. А. Н. Кисилев. – 2-е изд. – М.: ДМК Пресс, 2016. – 256 с.
9. Дауни А.Б. Байесовские модели / пер. с анг. В. А. Яроцкого. – М.: ДМК Пресс, 2018. – 182 с.
10. Белоусов А.И. Дискретная математика: учебник для вузов. – 5-е изд. – Москва: Издательство МГТУ им. Н.Э. Баумана, 2015. – 743 с. (Математика в техническом университете; вып. 19).
11. Материалы компании, посвященные системе GENSYM
https://ignitetech.com/softwarelibrary/gensym
12. URL https://en.wikipedia.org/wiki/Expert_system
13. URL https://habr.com/ru/post/346236/
14. Нейлор К. Как построить свою экспертную систему. Пер. с англ., 2001. – 286 С.
15. URL http://clipsrules.net/AboutCLIPS.html
16. URL https://en.wikipedia.org/wiki/Inference_engine
17. URL https://en.wikipedia.org/wiki/Automated_theorem_proving