БУ ВО

«СУРГУТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ХАНТЫ-МАНСИЙСКОГО АВТОНОМНОГО ОКРУГА - ЮГРЫ»

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Кафедра информатики и вычислительной техники

УТВЕРЖДАЮ:

Заведующий кафедрой ИВТ

к.т.н.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Д.А. Федоров

" \_\_ " \_\_\_\_\_\_\_\_\_\_\_ 2022г.

**КУРСОВОЙ ПРОЕКТ**

по дисциплине «**Интеллектуальные системы и технологии**»

бакалавра по направлению

09.03.02 - Информационные системы и технологии

на тему ***Разработка экспертной системы на основе выбранной модели представления знаний******(Проблемы с компьютером)***

Выполнил: бакалавр группы: \_\_\_\_\_\_\_

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

(фамилия, имя, отчество)

по специальности 09.03.02 - Информационные системы и технологии

Руководитель:

ст. преподаватель Шайторова И.А. \_\_\_\_\_\_\_\_\_\_

подпись

Сургут – 2022

БУ ВО

«СУРГУТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ХАНТЫ-МАНСИЙСКОГО АВТОНОМНОГО ОКРУГА - ЮГРЫ»

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Кафедра информатики и вычислительной техники

**ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ**

по направлению 09.03.02 - Информационные системы и технологии

дисциплина «Интеллектуальные системы и технологии»

(№ группы, ФИО)

**Тема** *Разработка экспертной системы на основе выбранной модели представления знаний (Проблемы с компьютером)*

**Целевая установка**

*Разработка экспертной системы определения болезни щитовидной железы для дальнейшего прогнозирования лечения врачом.*

**Исходные данные** Литература, сеть Internet, выборки данных из научно-исследовательских работ.

Начало проектирования 01 февраля 2022

Конец проектирования 30 апреля 2022

**Содержание работы**

1. Общая характеристика предметной области, определение ресурсов на разработку экспертной системы (ЭС).
2. Концептуализация предметной области (структурная модель предметной области, функциональная (поведенческая) модель предметной области).
3. Выбор метода формализации знаний и инструментальных методов разработки ЭС.
4. Проектирование базы знаний (формализация базы знаний, описание базы знаний на языке представления знаний (ЯПЗ), описание технологии загрузки и актуализации базы знаний).
5. Обзор полученных результатов. Вывод. Список литературы.
6. Подготовить отчет. Защитить Курсовой проект.

**Отчетный материал**

1. Пояснительная записка на 20-30 листах.

2. Графики и схемы

**Литература**

**Подписи:**

Задание получил студент(ка) Тунян Эдмон Гарниковчи /ФИО/

Руководитель проекта \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/И.А. Шайторова/

Зав. кафедрой ИВТ, к.т.н\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_/Д.А. Федоров/

**СПИСОК СОКРАЩЕНИЙ**

|  |  |  |
| --- | --- | --- |
| **ПЭВМ** | **–** | персональная электронно-вычислительная машина |
| **РП** | **–** | рабочей памяти |
| **БД** | **–** | базой данных |
| **БЗ** | **–** | базы знаний |
| **ЭС** | **–** | экспертная система |
| **ПК** | **–** | персональный компьютер |
| **ПР** | **–** | проводка |
| **В** | **–** | вывод |
| **ОС** | **–** | операционная система |
| **ПСЖ** | **–** | проблема с железом |
| **ОЗУ** | **–** | оперативно запоминающее устройство |
| **ВК** | **–** | видеокарта |
| **ЦП** | **–** | центральный процессор |
| **СЭ** | **–** | синий экран |
| **СОС** | **–** | статус операционной системы |

**РЕФЕРАТ**

Курсовой проект содержит 51 страниц, 12 рисунков, 20 таблиц, 20 используемых источника, 1 приложения.

**Разработка экспертной системы определения проблем работы ПЭВМ для дальнейшего прогнозирования мастером**

*Ключевые слова*: продукционная модель представления знаний, сетевая модель представления знаний, фреймовая модель представления знаний, модели представления знаний, проблемы с компьютером, проблемы с операционной системой, проблемы с железом.

*Цель работы* – разработка экспертной системы определения проблем работы ПЭВМ для прогнозирования мастером устранения неполадок.

Основные результаты:

– решена задача 1, т.е. построена продукционная модель представления знаний в предметной области «Проблемы с ПЭВМ»;

– решена задача 2, т.е. построена сетевая модель представления знаний в выбранной предметной области «Проблемы с ПЭВМ»;

– решена задача 3, т.е. построена фреймовая модель представления знаний в предметной области «Проблемы с ПЭВМ»;

– решена задача 4, т.е. разработана экспертная система (ЭС) на основе выбранной модели представления знаний «Проблемы с ПЭВМ».

ОГЛАВЛЕНИЕ

[Глава 1. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ 5](#_Toc101443871)

[1. Продуктовая модель 5](#_Toc101443872)

[2. Фреймовая модель 7](#_Toc101443873)

[3. Семантическая модель 11](#_Toc101443874)

[4. Экспертная система 13](#_Toc101443875)

[4.1 Структура экспертной системы 13](#_Toc101443876)

[4.2 Состав участников разработки экспертной системы 14](#_Toc101443877)

[4.3 Классификация экспертных систем 15](#_Toc101443878)

[Глава 2. ПРАКТИЧЕСКАЯ ЧАСТЬ 19](#_Toc101443879)

[1. Продукционная модель 19](#_Toc101443880)

[1.1 Описание процесса решения 19](#_Toc101443881)

[1.2 Решение 20](#_Toc101443882)

[2. Фреймовая модель 22](#_Toc101443883)

[2.1 Описание процесса решения 22](#_Toc101443884)

[2.2 Решение 23](#_Toc101443885)

[3. Семантическая модель 31](#_Toc101443886)

[3.1 Описание процесса решения 31](#_Toc101443887)

[3.2 Решение 31](#_Toc101443888)

[4. Экспертная система 35](#_Toc101443889)

[4.1 Переменные экспертной системы: 35](#_Toc101443890)

[4.2 Правила экспертной системы: 35](#_Toc101443891)

[4.3 Стек технологий для разработки экспертной систем 37](#_Toc101443892)

[4.4 Демонстрация работы экспертной системы 38](#_Toc101443893)

[ЗАКЛЮЧЕНИЕ 40](#_Toc101443894)

[ЛИТЕРАТУРА 41](#_Toc101443895)

[ПРИЛОЖЕНИЕ 44](#_Toc101443896)

Глава 1. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

# Продуктовая модель

Продукционная модель представления знаний — это модель, которая базируется на определённых правилах и даёт возможность выразить знание в форме предложений типа «Если (условие A), то (действие B)» ( ).

*Под условием* понимается образец, по которому осуществляется поиск в базе знаний, а *действием* — действие, выполняемые при успешном исходе поиска (они могут быть промежуточными, выступающими далее как условия, и терминальными или целевыми, завершающими работу системы).

Традиционная продукционная модель знаний включает в себя следующие базовые компоненты:

1. набор правил (или продукций), представляющих базу знаний продукционной системы;
2. рабочую память, в которой хранятся исходные факты, а также факты, выведенные из исходных фактов при помощи механизма логического вывода;
3. сам механизм логического вывода, позволяющий из имеющихся фактов, согласно имеющимся правилам вывода, выводить новые факты.

Существуют для типа выполнения систем продукции:

1. прямой
2. обратный

*Прямой вывод (прямая цепочка рассуждений)* называется также выводом, управляемым данными, или нисходящим. В таких системах поиск идет от исходных данных (фактов) к заключениям.

*Обратный вывод* (*обратная цепочка рассуждений*) называется также выводом, управляемым целями, или восходящим. В таких системах выдвигается некоторая гипотеза В, а затем идет поиск промежуточных фактов A, подтверждающих эту гипотезу.

Достоинства продукционной модели

1. Использование принципа модульности.
2. Простота поддержки, модификации правил.
3. Простота механизма логического вывода.

Недостатки продукционной модели

1. Неясность взаимных отношений правил.
2. Долгий неэффективный поиск и проверка на применимость правил.
3. Сложность оценки целостного образа знаний.

# Фреймовая модель

Теория фреймов была предложена в 1974 году Марвином Минским.

*Фрейм* – это структура данных, в которой в определенном порядке представлены сведения о свойствах объекта. Под фреймом понимается однажды определенная единица представления знаний, которую можно изменять лишь в деталях согласно текущей ситуации.

Два вида описания фрейма:

1 вид: рис 1 (имя фрейма и слоты)

Имя фрейма: ноутбук

2 вид: в виде таблицы (таблица 1)

Имя фрейма: человек

Таблица 1. Фрейм прототип-образец "Человек"

|  |  |  |  |
| --- | --- | --- | --- |
| **ЧЕЛОВЕК** | | | |
| **Имя слота** | **Значение слота** | **Способ получения значения** | **Демон** |
| Пол | Мужской или женский | Из внешних источников |  |
| Возраст | От 0 до 100 лет | Из внешних источников |  |

В общем случае структура данных фрейма может содержать более широкий набор информации, в который входят следующие атрибуты:

1. *Имя фрейма*. Оно служит для идентификации фрейма в системе и должно быть уникальным. Фрейм представляет собой совокупность слотов, число которых может быть произвольным.
2. *Имя слота*. Оно должно быть уникальным в пределах фрейма.
3. *Значение слота*. Оно должно соответствовать указанному типу данных и условию наследования. Значением слота могут быть числа или математические соотношения, тексты на естественном языке или программы, правила вывода или ссылки на другие слоты данного фрейма или других фреймов.
4. *Демон*. Демоном называется присоединенная процедура, автоматически запускаемая при выполнении некоторого условия.

* демон IF-NEEDED запускается, если в момент обращения к слоту его значение не было установлено
* демон IF-ADDED запускается при подстановке в слот значения
* демон IF-REMOVED – при стирании значения слота

1. *Тип данных*. Определяет тип значения слота. Употребляемые типы:

* *frame* — указатель на фрейм;
* *real* — вещественное число;
* *integer* — целое число;
* *boolean* — логический тип;
* *text* — фрагмент текста;
* *list* — список;
* *table* — таблица;
* *expression* — выражение;
* *lisp* — присоединенная процедура и т.д.

Пример фрейма – ноутбук:



Рисунок 1. Пример фрейма

***Фреймовая модель*** – модель представления знаний о свойствах объекта в виде структуры данных.

Применение фреймовой модели позволяет структурировать информацию в табличном или матричном виде. Фрейм также возможно рассматривать в виде *семантической сети*.

Фреймы образуют иерархию. Иерархия во фреймовых моделях порождает единую многоуровневую структуру, описывающую либо объект, если слоты описывают только свойства объекта, либо ситуацию или процесс, если отдельные слоты являются именами процедур, присоединенных к фрейму и вызываемых при его актуализации.

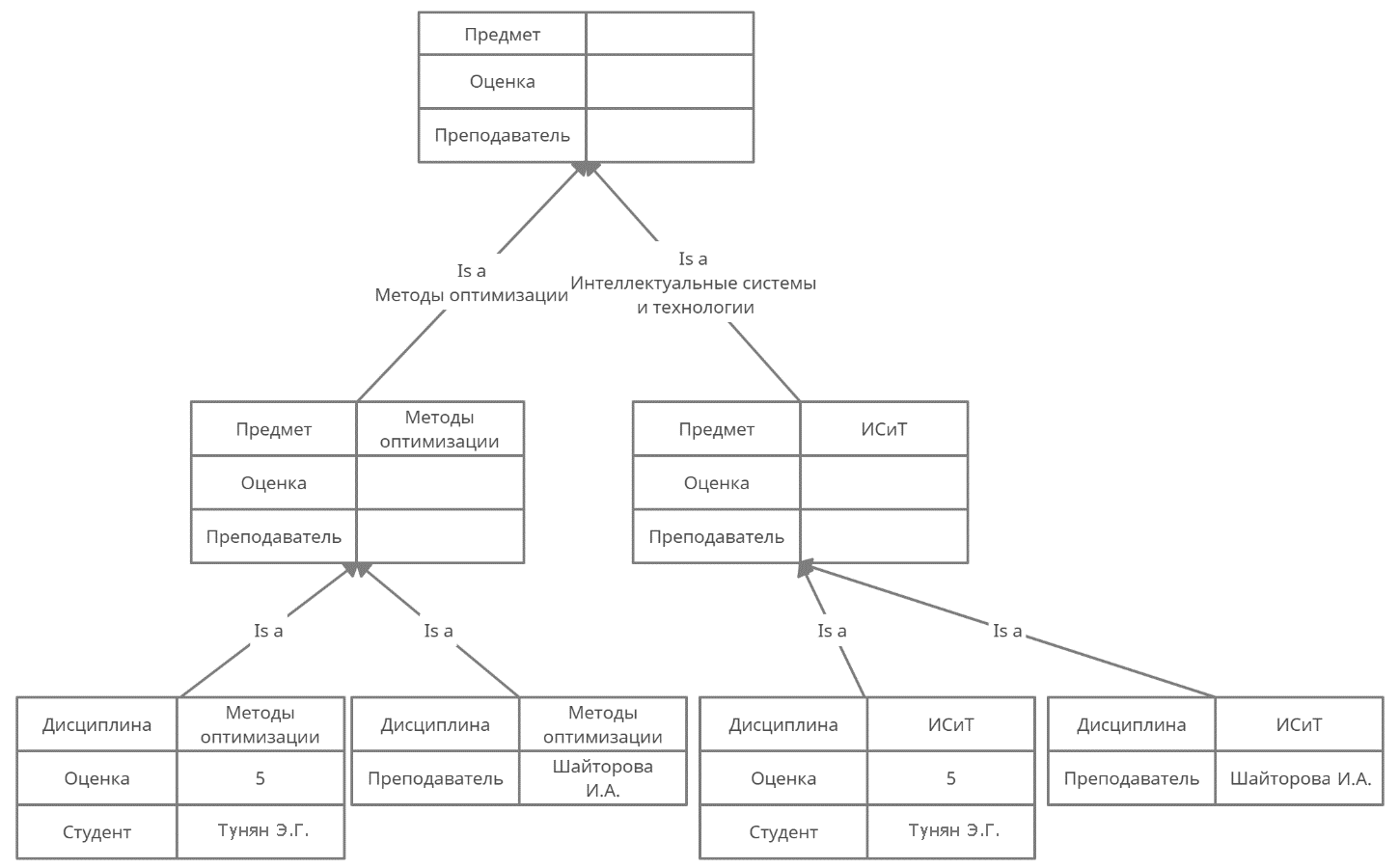


Рисунок 2. Пример фреймовой модели

Фреймы подразделяются на:

1. *Фрейм-экземпляр* - конкретная реализация фрейма, описывающая текущее состояние в предметной области.
2. *Фрейм-образец* – шаблон для описания объектов или допустимых ситуаций в предметной области.
3. *Фрейм-класс* – фрейм верхнего уровня для предоставления совокупности фреймов образцов.

*Достоинства фреймовой модели*

1. Эффективная реализация процедур вывода.
2. Обеспечивает хорошее соответствие реальной действительности
3. Позволяет комбинировать различные модели представления знаний, объединяя их достоинства и компенсируя их недостатки.

*Недостатки фреймовой модели*

1. Каждый фрейм может представлять собой сложный объект знаний. Удаление или добавление нового фрейма может быть затруднительным, поскольку должно предусматривать удаление всех других элементов, которые являются составными частями других фреймов.
2. Достаточно сложно осуществлять на фреймах представление временных процессов.
3. Отсутствует формальная теория вывода на фреймах. Поэтому на инженере по знаниям целиком лежит ответственность за корректность организации иерархии фреймов и их заполнения.

# Семантическая модель

*Семантическая модель* – это ориентированный граф, вершины которого – понятия, а дуги – отношения между ними. Термин «семантическая» означает «смысловая», а сама семантика – это наука, устанавливающая отношения между символами и объектами, которые они обозначают, то есть наука, определяющая смысл знаков. Модель на основе семантических сетей была предложена американским психологом Куиллианом.

В качестве понятий обычно выступают абстрактные или конкретные объекты, а отношения – это связи типа: «это» (АКО – A-Kind-Of, is или «элемент класса»), «имеет частью» (has part), «принадлежит», «любит» и т.д.

Можно предложить несколько классификаций семантических сетей, связанных с типами отношений между понятиями.

По количеству типов отношений:

* однородные (с единственным типом отношений);
* неоднородные (с различными типами отношений).

По типам отношений:

* бинарные (в которых отношения связывают два объекта);
* N-арные (в которых есть специальные отношения, связывающие более двух понятий).

Наиболее часто в семантических сетях используются следующие отношения:

* элемент класса (роза – это цветок);
* атрибутивные связи /иметь свойство (память имеет свойство – объем);
* значение свойства (цвет имеет значение желтый);
* пример элемента класса (роза, например, чайная);
* связи типа «часть-целое» (велосипед включает руль);
* функциональные связи (определяемые обычно глаголами «производит», «влияет» и др.);
* количественные (больше, меньше, равно и др.);
* пространственные (далеко от, близко от, за, под, над и др.);
* временные (раньше, позже, в течение и др.);
* логические связи (и, или, не) и др.
* Минимальный состав отношений в семантической сети таков:
* элемент класса или АКО;
* атрибутивные связи /иметь свойство;
* значение свойства.

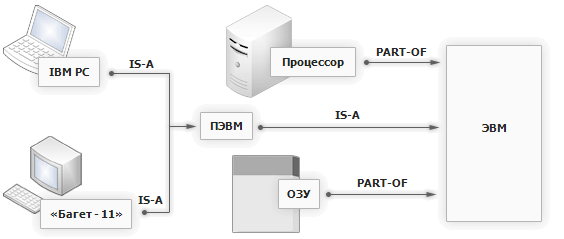


Рисунок 3. Пример семантической модели

*Достоинства семантической модели*

1. Простота и прозрачность описания.
2. Открытость, позволяющая выполнять дополнение и модифицирование
3. Близость структуры сети, представляющей систему знаний, семантической структуре фраз на естественном языке

*Недостатки семантической модели*

1. Отсутствие формального аппарата установления противоречивости описания
2. Сетевая модель не содержит ясного представления о структуре предметной области, поэтому формирование и модификация такой модели затруднительны

# Экспертная система

*Экспертная система*— компьютерная система, способная частично заменить специалиста-эксперта в разрешении проблемной ситуации.

Как правило, экспертные системы создаются для решения практических задач в некоторых узкоспециализированных областях, где большую роль играют знания опытных специалистов. Экспертные системы были первыми разработками, которые смогли привлечь большое внимание к результатам исследований в области искусственного интеллекта.

Экспертные системы имеют одно большое отличие от других систем искусственного интеллекта: они не предназначены для решения каких-то универсальных задач, как например нейронные сети или генетические алгоритмы. Экспертные системы предназначены для качественного решения задач в определенной разработчиками области, в редких случаях – областях.

Типичная статическая ЭС состоит из следующих основных компонентов решателя (интерпретатора):

1. рабочей памяти (РП), называемой также базой данных (БД);
2. базы знаний (БЗ);
3. компонентов приобретения знаний;
4. объяснительного компонента;
5. диалогового компонента.
   1. Структура экспертной системы

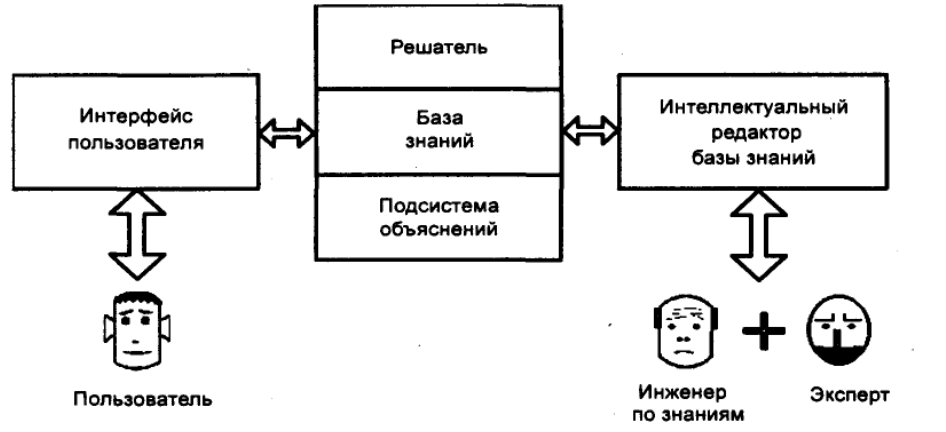
*База данных (рабочая память)* предназначена для хранения исходных и промежуточных данных решаемой в текущий момент задачи. Этот термин совпадает по названию, но не по смыслу с термином, используемым в информационно-поисковых системах (ИПС) и системах управления базами данных (СУБД) для обозначения всех данных (в первую очередь долгосрочных), хранимых в системе.

*База знаний* (БЗ) в ЭС предназначена для хранения долгосрочных данных, описывающих рассматриваемую область (а не текущих данных), и правил, описывающих целесообразные преобразования данных этой области.

*Решатель****,*** используя исходные данные из рабочей памяти и знания из БЗ, формирует такую последовательность правил, которые, будучи примененными к исходным данным, приводят к решению задачи.

*Компонент* *приобретения знаний* автоматизирует процесс наполнения ЭС знаниями, осуществляемый пользователем-экспертом.

*Объяснительный компонент* объясняет, как система получила решение задачи (или почему она не получила решение) и какие знания она при этом использовала, что облегчает эксперту тестирование системы и повышает доверие пользователя к полученному результату.



*Рисунок 4. Пример структуры экспертной системы*

* 1. Состав участников разработки экспертной системы

*Экспертная система* — это программное средство, использующее знания экспертов, для высокоэффективного решения задач в интересующей пользователя предметной области. Она называется системой, а не просто программой, так как содержит базу знаний, решатель проблемы и компоненту поддержки.

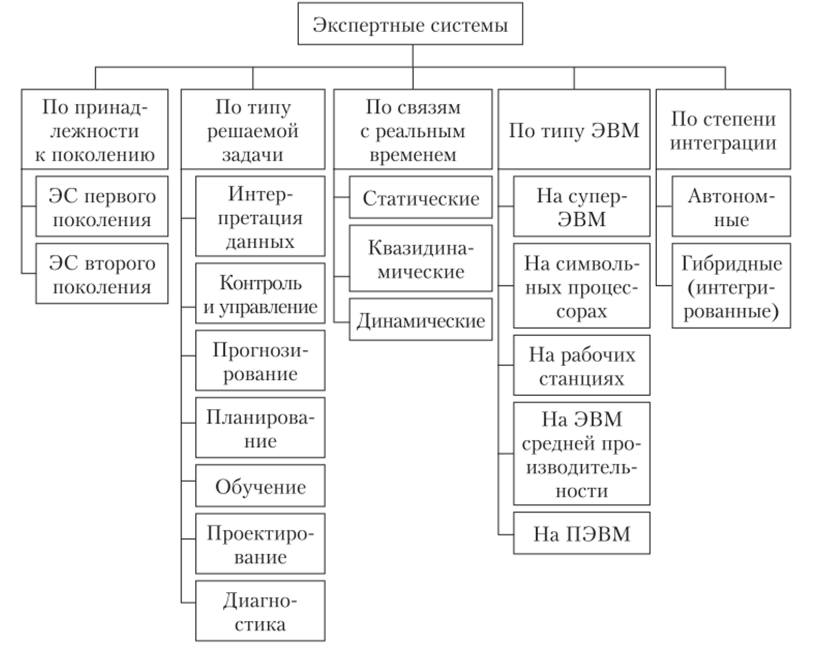
*Эксперт* — это человек, способный ясно выражать свои мысли и пользующийся репутацией специалиста, умеющего находить правильные решения проблем в конкретной предметной области. Эксперт использует свои знания, чтобы сделать поиск решения более эффективным, и ЭС моделирует все его стратегии.

*Инженер знаний* - человек, как правило, имеющий познания в информатике и искусственном интеллекте и знающий, как надо строить ЭС. Инженер знаний опрашивает экспертов, организует знания, решает, каким образом они должны быть представлены в ЭС, и может помочь программисту в написании программ.

*Средство построения ЭС* — это программное средство, используемое инженером знаний или программистом для построения ЭС. Этот инструмент отличается от обычных языков программирования тем, что обеспечивает удобные способы представления сложных высокоуровневых понятий.

*Пользователь* — это человек, который использует уже построенную ЭС.

* 1. Классификация экспертных систем



*Рисунок 5. Классификация экспертных систем*

*Классификация по решаемой задаче*

* Интерпретация данных. Это 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.
* Диагностика. Под диагностикой понимается обнаружение неисправности в некоторой системе. Неисправность — это 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. Важной спецификой является необходимость понимания функциональной структуры ("анатомии") диагностирующей системы.
* Мониторинг. Основная задача мониторинга - непрерывная интерпретация данных в реальном масштабе времени и сигнализация о выходе тех или иных параметров за допустимые пределы. Главные проблемы - "пропуск" тревожной ситуации и инверсная задача "ложного" срабатывания. Сложность этих проблем в размытости симптомов тревожных ситуаций и необходимость учета временного контекста.
* Проектирование. Проектирование состоит в подготовке спецификаций на создание "объектов" с заранее определенными свойствами. Под спецификацией понимается весь набор необходимых документов чертеж, пояснительная записка и т.д. Основные проблемы здесь - получение четкого структурного описания знаний об объекте и проблема "следа". Для организации эффективного проектирования и, в еще 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 ЭС: процесс вывода решения и процесс объяснения.
* Прогнозирование. Прогнозирующие 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.
* Планирование. Под планированием понимается нахождение планов действий, относящихся к объектам, способным выполнять некоторые функции. В таких ЭС используются модели поведения реальных объектов с тем, чтобы логически вывести последствия планируемой деятельности.
* Обучение. Системы обучения диагностируют ошибки при изучении какой-либо дисциплины с помощью ЭВМ и подсказывают правильные решения. Они аккумулируют знания о гипотетическом "ученике" и его характерных ошибках, затем в работе способны диагностировать слабости в знаниях обучаемых и находить соответствующие средства для их ликвидации. Кроме того, они планируют акт общения с учеником в зависимости от успехов ученика с целью передачи знаний.

*Классификация по связи с реальным временем*

* Статические ЭС разрабатываются в предметных областях, в которых база знаний и интерпретируемые данные не меняются во времени. Они стабильны.
* Квазидинамические ЭС интерпретируют ситуацию, которая меняется с некоторым фиксированным интервалом времени.
* Динамические ЭС работают в сопряжении с датчиками объектов в режиме реального времени с непрерывной интерпретацией поступаемых данных.

*Классификация по типу ЭВМ*

На сегодняшний день существуют:

* ЭС для уникальных стратегически важных задач на суперЭВМ (Эльбрус, CRA'. CONVEX и др.);
* ЭС на ЭВМ средней производительности (типа ЕС ЭВМ, mainframe);
* ЭС на символьных процессорах и рабочих станциях (SUN, APOLLO);
* ЭС на мини- и супермини-ЭВМ (VAX, micro-VAX и др.);
* ЭС на персональных компьютерах (IBM PC, MAC II и подобные).

*Классификация по степени интеграции с другими программами*

* Автономные ЭС работают непосредственно в режиме консультаций с пользователем для специфически "экспертных" задач, для решения которых не требуется привлекать традиционные методы обработки данных (расчеты, моделирование и т. д.).
* Гибридные ЭС представляют программный комплекс, агрегирующий стандартные пакеты прикладных программ (например, математическую статистику, линейное программирование или системы управления базами данных) и средства манипулирования знаниями. Это может быть интеллектуальная надстройка над ППП или интегрированная среда для решения сложной задачи с элементами экспертных знаний.

Несмотря на внешнюю привлекательность гибридного подхода, следует отметить, что разработка таких систем являет собой задачу, на порядок более сложную, чем разработка автономной ЭС. Стыковка не просто разных пакетов, а разных методологий (что происходит в гибридных системах) порождает целый комплекс теоретических и практических трудностей.

Глава 2. ПРАКТИЧЕСКАЯ ЧАСТЬ

Задание 1.

* + 1. Продукционная модель

Построить продукционную модель представления знаний в предметной области «*Проблемы с ПЭВМ*».

Описание процесса решения

Для построения продукционной модели представления знаний необходимо выполнить следующие шаги:

1) Определить целевые действия задачи.

2) Определить промежуточные действия или цепочку действий, между начальным состоянием и конечным.

3) Опередить условия для каждого действия, при котором его целесообразно и возможно выполнить. Определить порядок выполнения действий.

4) Добавить конкретики при необходимости, исходя из поставленной задачи.

5) Преобразовать полученный порядок действий и соответствующие им условия в продукции.

6) Для проверки правильности построения продукций записать цепочки продукций, явно проследив связи между ними. Этот набор шагов предполагает движение при построении продукционной модели от результата к начальному состоянию, но возможно и движение от начального состояния к результату (шаги 1 и 2).

1.2 Решение

1) Обязательные действия, выполняемое в сервисах по ремонту компьютеров – диагностика ПЭВМ и решение проблемы. Значит, есть уже два целевых действия «диагностика ПЭВМ» и «устранение проблемы», которые взаимосвязаны и следуют друг за другом.

2) Прежде чем начать устранение проблемы, нужно узнать проблему, определить причины поломки и сделать вывод (в дальнейшем «диагноз»). Значит, цепочка промежуточных действий: «узнать проблему», «определить причины поломки».

3) Прежде чем начать устранение проблемы, необходимо убедиться, что диагноз верный. Выбор диагноза обуславливается поведением ПЭВМ, с которым пользователь столкнулся при его запуске. Значит сначала идут действия, позволяющие определить диагноз, а затем действия по устранению проблемы.

4) Вышеописанное можно преобразовать в следующие предложения типа «Если, то»:

* Если у пользователя есть проблемы с запуском ПЭВМ, то он обращается в сервис к мастеру.
* Если у пользователя при запуске загрузка дальше BIOS не идет, то вывод на ПЭВМ отсутствует ОС.
* Если у пользователя при запуске загрузка заканчивается и синий экран, то вывод у пользователя проблемы с ОС.
* Если у пользователя при запуске загрузка вообще не начинается, то вывод проблема с комплектующим (в дальнейшем «железо»).
* Если проблема с железом ПЭВМ, то произвести диагностику используя POST карту, сделать вывод по неисправности.
* Если проблема выявлена, то устранить проблему.

Введем обозначения для фактов (Ф), действий (Д) и продукций (П), тогда:

Ф1 = клиент жалуется, что есть проблемы с запуском ПЭВМ

Ф2 = клиент жалуется, что при запуске загрузка дальше BIOS не идет

Ф3 = клиент жалуется, что при запуске после загрузки синий экран

Ф4 = клиент жалуется, что при запуске, загрузка вообще не начинается

Ф5 = диагноз: у пользователя проблемы с ОС

Ф6 = диагноз: у пользователя проблемы с комплектующими

Ф7 = диагностика не произведена

Ф8 = проблема не устранена

Д1 = пользователь может пойти к мастеру

Д2 = мастер ставит диагноз (делает вывод на основе результата диагностики), проблемы с ОС

Д3 = мастер ставит диагноз (делает вывод на основе результата диагностики), проблемы с комплектующими

Д4 = невозможно сделать вывод (заключение, диагноз)

Д5 = проводится диагностика и на основе его результатов, делается вывод (заключение, диагноз)

Д6 = устраняется проблема

Для продукций установим приоритет (в скобках перед запятой, чем выше приоритет, чем раньше проверяется правило).

П1(5, Ф1) = Д1

П2(4, Д1 и (Ф2 или Ф3) и Ф5) = Д2

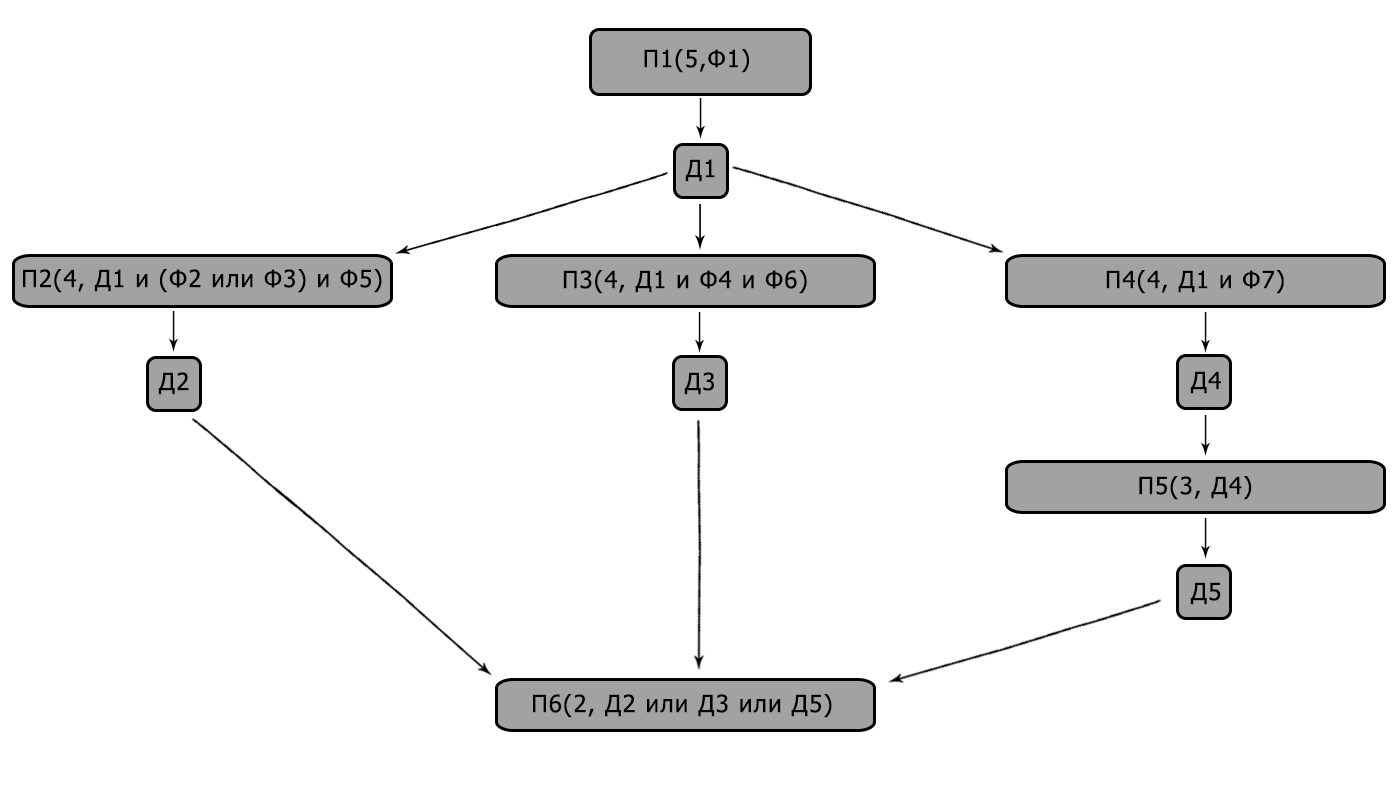
П3(4, Д1 и Ф4 и Ф6) = Д3

П4(4, Д1 и Ф7) = Д4

П5(3, Д4) = Д5

= Д6

6) Для отображения взаимосвязи продукций построим граф (рис. 1).



*Рис. 1. Схема продукций предметной области «Проблемы с ПЭВМ».*

# Фреймовая модель

Построить фреймовую модель представления знаний в предметной области «*Проблемы с ПЭВМ*».

2.1 Описание процесса решения

Для построения фреймовой модели представления знаний необходимо выполнить *следующие* шаги:

1. Определить абстрактные объекты и понятия предметной области, необходимые для решения поставленной задачи. Оформить их в виде фреймов-прототипов (фреймов-объектов, фреймов-ролей).
2. Задать конкретные объекты предметной области. Оформить их в виде фреймов-экземпляров (фреймов-объектов, фреймов-ролей).
3. Определить набор возможных ситуаций. Оформить их в виде фреймов-ситуаций (прототипы). Если существуют прецеденты по ситуациям в предметной области, добавить фреймы-экземпляры фреймы-ситуации).
4. Описать динамику развития ситуаций (переход от одних к другим) через набор сцен. Оформить их в виде фреймов-сценариев.
5. Добавить фреймы-объекты сценариев и сцен, которые отражают данные конкретной задачи.

2.2 Решение

1) Ключевые понятия данной предметной области – компьютер, тот, у кого возникли проблемы с компьютером (клиент) и те, кто его диагностируют и ремонтируют (мастера). У обслуживающего персонала и клиентов есть общие характеристики, поэтому целесообразно выделить общее абстрактное понятие - человек. Ремонтируют компьютеры некоторая компания. Тогда фреймы «Компания по ремонту», «Компьютер» и «Человек» являются прототипами-образцами, а фреймы «Мастер» и «Клиент» - прототипами-ролями. Также нужно определить основные слоты фреймов – характеристик, имеющие значения для решаемой задачи.

Таблица 1. Фрейм прототип-образец "Человек"

|  |  |  |  |
| --- | --- | --- | --- |
| **ЧЕЛОВЕК** | | | |
| **Имя слота** | **Значение слота** | **Способ получения значения** | **Демон** |
| Пол | Мужской или женский | Из внешних источников |  |
| Возраст | От 0 до 100 лет | Из внешних источников |  |

Таблица 2. Фрейм прототип-образец "Компьютер"

|  |  |  |  |
| --- | --- | --- | --- |
| **КОМПЬЮТЕР** | | | |
| **Имя слота** | **Значение слота** | **Способ получения значения** | **Демон** |
| Название |  | Из внешних источников |  |
| Материнская плата |  | Из внешних источников |  |
| Видеокарта |  | Из внешних источников |  |
| Процессор |  | Из внешних источников |  |
| ОЗУ |  | Из внешних источников |  |
| ОС |  |  |  |

Таблица 3. Фрейм прототип-образец " Компания по ремонту"

|  |  |  |  |
| --- | --- | --- | --- |
| **КОМПАНИЯ ПО РЕМОНТУ** | | | |
| **Имя слота** | **Значение слота** | **Способ получения значения** | **Демон** |
| Название |  | Из внешних источников |  |
| Адрес |  | Из внешних источников |  |
| Часы работы |  | Из внешних источников |  |

Фреймы-наследники содержат все слоты своих родителей, они явно прописываются только в случае изменения какого-либо параметра.

Таблица 4. Фрейм прототип-роль " Мастер "

|  |  |  |  |
| --- | --- | --- | --- |
| **МАСТЕР (ЧЕЛОВЕК)** | | | |
| **Имя слота** | **Значение слота** | **Способ получения значения** | **Демон** |
| Возраст | От 18 до 60 лет | Из внешних источников |  |
| Зарплата | 15к - 70к | Из внешних источников |  |
| Место работы | Фрейм-объект | Из внешних источников |  |
| График работы | 08:00 - 18:00 | Из внешних источников |  |

Таблица 5. Фрейм прототип-роль "Клиент"

|  |  |  |  |
| --- | --- | --- | --- |
| **КЛИЕНТ (ЧЕЛОВЕК)** | | | |
| **Имя слота** | **Значение слота** | **Способ получения значения** | **Демон** |
| Вид оплаты | Наличный или безналичный | Из внешних источников |  |
| Статус заказа | Воспользовался услугой мастера или нет | Из внешних источников |  |

2) Фреймы-образцы описывают конкретную ситуацию: какие компании имеются в городе, как именно организовывается посещение, кто является клиентом, кто работает в выбранной компании т.д. Поэтому определим следующие фреймы-образцы, являющиеся наследниками фреймов-прототипов:

Таблица 6. Фрейм-образец "Вуди Сервис"

|  |  |  |  |
| --- | --- | --- | --- |
| **КОМПАНИЯ “** **Вуди Сервис” (КОМПАНИЯ ПО РЕМОНТУ)** | | | |
| **Имя слота** | **Значение слота** | **Способ получения значения** | **Демон** |
| Название | Вуди Сервис | Из внешних источников |  |
| Адрес | г. Сургут, улица ​Игоря Киртбая, 11 | Из внешних источников |  |
| Часы работы | 10:00 – 20:00 | Из внешних источников |  |

Таблица 7. Фрейм-образец "Родина Сервис"

|  |  |  |  |
| --- | --- | --- | --- |
| **КОМПАНИЯ “Родина Сервис” (КОМПАНИЯ ПО РЕМОНТУ)** | | | |
| **Имя слота** | **Значение слота** | **Способ получения значения** | **Демон** |
| Название | Родина Сервис | Из внешних источников |  |
| Адрес | г. Сургут, улица ​ ​Игоря Киртбая, 19 | Из внешних источников |  |
| Часы работы | 10:00 – 20:00 | Из внешних источников |  |

Таблица 8. Фрейм-образец "Мобильная клиника"

|  |  |  |  |
| --- | --- | --- | --- |
| **КОМПАНИЯ “Мобильная клиника” (КОМПАНИЯ ПО РЕМОНТУ)** | | | |
| **Имя слота** | **Значение слота** | **Способ получения значения** | **Демон** |
| Название | Мобильная клиника | Из внешних источников |  |
| Адрес | г. Сургут, улица ​ ​ ​Геологическая, 10​ 1 этаж; | Из внешних источников |  |
| Часы работы | 10:00 – 22:00 | Из внешних источников |  |

Таблица 9. Фрейм-образец "Василий"

|  |  |  |  |
| --- | --- | --- | --- |
| **ВАСИЛИЙ (МАСТЕР)** | | | |
| **Имя слота** | **Значение слота** | **Способ получения значения** | **Демон** |
| Возраст | 27 | Из внешних источников |  |
| Пол | Мужской | Из внешних источников |  |
| Зарплата | 35 000 | Из внешних источников |  |
| Место работы | Родина Сервис | Из внешних источников |  |
| График работы | 10:00 – 20:00 | Из внешних источников |  |

Таблица 10. Фрейм-образец "Петр"

|  |  |  |  |
| --- | --- | --- | --- |
| **ПЕТР (МАСТЕР)** | | | |
| **Имя слота** | **Значение слота** | **Способ получения значения** | **Демон** |
| Возраст | 24 | Из внешних источников |  |
| Пол | Мужской | Из внешних источников |  |
| Зарплата | 40 000 | Из внешних источников |  |
| Место работы | Вуди Сервис | Из внешних источников |  |
| График работы | 10:00 – 20:00 | Из внешних источников |  |

Таблица 11. Фрейм-образец "Екатерина"

|  |  |  |  |
| --- | --- | --- | --- |
| **ЕКАТЕРИНА (МАСТЕР)** | | | |
| **Имя слота** | **Значение слота** | **Способ получения значения** | **Демон** |
| Возраст | 22 | Из внешних источников |  |
| Пол | Женский | Из внешних источников |  |
| Зарплата | 70 000 | Из внешних источников |  |
| Место работы | Мобильная клиника | Из внешних источников |  |
| График работы | 10:00 – 22:00 | Из внешних источников |  |

Таблица 12. Фрейм-образец "Jr101"

|  |  |  |  |
| --- | --- | --- | --- |
| **JR101 (КОМПЬЮТЕР)** | | | |
| **Имя слота** | **Значение слота** | **Способ получения значения** | **Демон** |
| Название | Jr101 | Из внешних источников |  |
| Материнская плата | MSI B460M PRO-VDH | Из внешних источников |  |
| Видеокарта | GTX 1080 | Из внешних источников |  |
| Процессор | Intel Core i5-10600K | Из внешних источников |  |
| ОЗУ | Kingston HyperX FURY Black [HX426C16FB3K2/16] 16 ГБ | Из внешних источников |  |
| ОС | Windows 10 | Из внешних источников |  |

Таблица 13. Фрейм-образец " 8112-19"

|  |  |  |  |
| --- | --- | --- | --- |
| **8112-19 (КОМПЬЮТЕР)** | | | |
| **Имя слота** | **Значение слота** | **Способ получения значения** | **Демон** |
| Название | 8112-19 | Из внешних источников |  |
| Материнская плата | GIGABYTE B450M S2H | Из внешних источников |  |
| Видеокарта | GTX 1050 Ti | Из внешних источников |  |
| Процессор | AMD Ryzen 3 2200G | Из внешних источников |  |
| ОЗУ | AMD Radeon R7 Performance Series [R748G2606U2S-U] 8 ГБ | Из внешних источников |  |
| ОС | Windows 8 | Из внешних источников |  |

Таблица 14. Фрейм-образец "Василий"

|  |  |  |  |
| --- | --- | --- | --- |
| **ВАСИЛИЙ (КЛИЕНТ)** | | | |
| **Имя слота** | **Значение слота** | **Способ получения значения** | **Демон** |
| Возраст | 25 | Из внешних источников |  |
| Пол | Мужской | Из внешних источников |  |
| Вид оплаты | Безналичный | По умолчанию (безналичный) |  |
| Статус заказа | Воспользовался услугой мастера | По умолчанию (не воспользовался) |  |

3) Фреймы-ситуации описывают возможные ситуации. В компании клиент попадает в несколько типичные ситуаций: заказ и оплата. Возможны и другие не типичные ситуации: клиент подавился, у клиента нет наличности для оплаты счета и т.д. Рассмотрим типичные ситуации (их может быть больше):

Таблица 15. Фрейм-ситуация "Заказ"

|  |  |  |  |
| --- | --- | --- | --- |
| **ЗАКАЗ** | | | |
| **Имя слота** | **Значение слота** | **Способ получения значения** | **Демон** |
| Перечень работ | Фрейм-образцы | Из внешних источников | IF-ADDED (изменение слота “Цены”) |
| Цены |  | Присоединенная процедура | IF-ADDED (изменение слота “Сумма заказа”) |
| Сумма заказа |  | Присоединенная процедура |  |
| Принял | Фрейм-образец | Из внешних источников |  |
| Сделал | Фрейм-образец | Из внешних источников |  |

Таблица 16. Фрейм-ситуация "Оплата"

|  |  |  |  |
| --- | --- | --- | --- |
| **ОПЛАТА** | | | |
| **Имя слота** | **Значение слота** | **Способ получения значения** | **Демон** |
| Вид оплаты |  | Из внешних источников |  |
| Заказ | Фрейм-образец | Из внешних источников | IF-ADDED (изменение слота “Оплатил”) |
| Оплатил | Фрейм-образец | Присоединенная процедура |  |

4) Ситуации возникают после наступления каких-то событий, выполнения условий и могут следовать одна за другой. Динамику предметной области можно отобразить в фреймах-сценариях. Их может быть множество, опишем наиболее общий и типичный сценарий посещения компании по ремонту.

Таблица 17. Фрейм-сценарий "Ремонта компьютера"

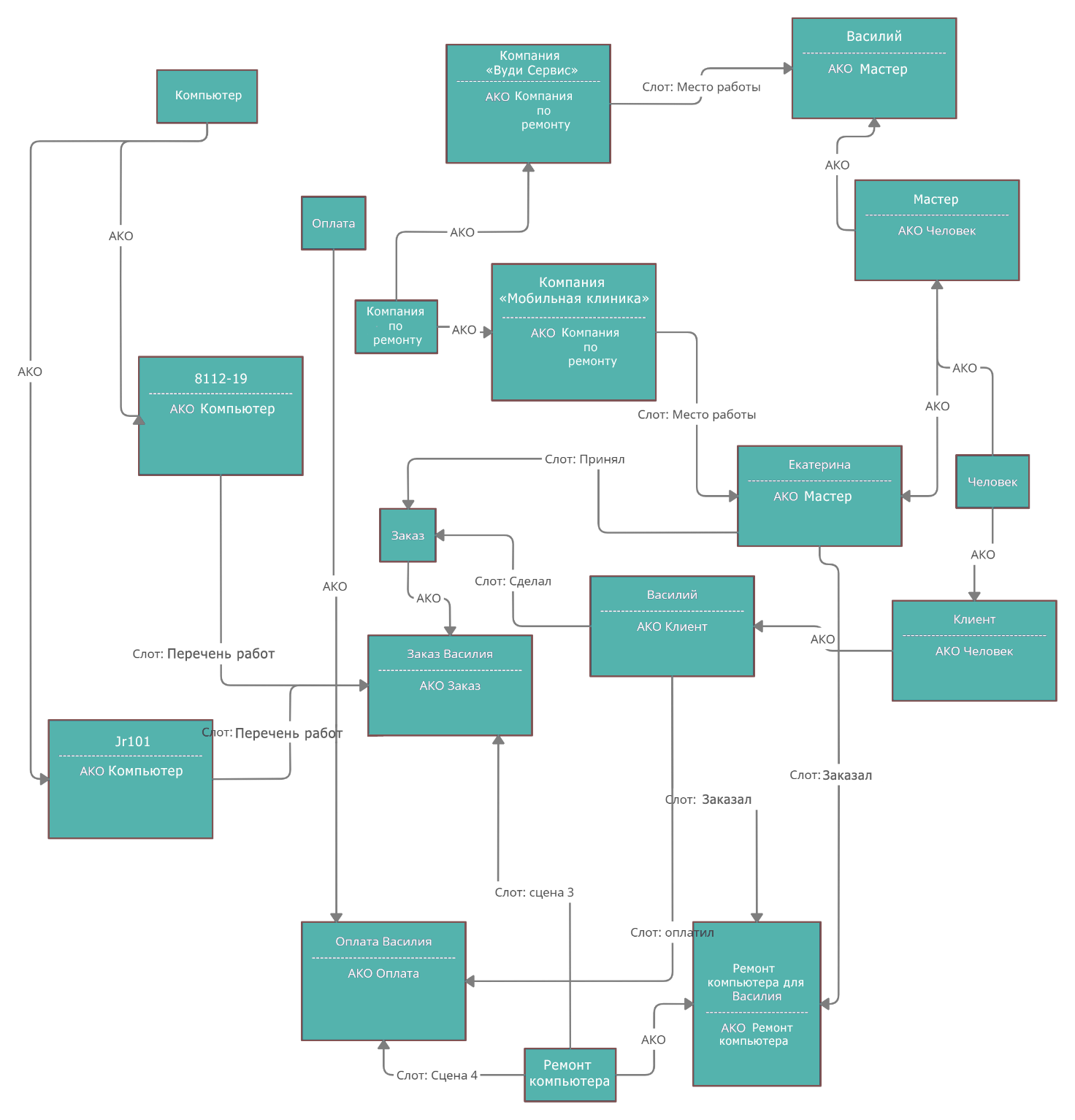
|  |  |  |  |
| --- | --- | --- | --- |
| **РЕМОНТ КОМПЬЮТЕРА** | | | |
| **Имя слота** | **Значение слота** | **Способ получения значения** | **Демон** |
| Воспользовался услугой | Фрейм-объект | Из внешних источников | IF-ADDED (изменение слота “Мастер”) |
| Компания | Фрейм-объект | Из внешних источников |  |
| Мастер | Фрейм-объект | Присоединенная процедура |  |
| Сцена 1 | Вход, название проблемы | Из внешних источников |  |
| Сцена 2 | Представление возможных решений | Из внешних источников |  |
| Сцена 3 | Выбор клиента | Из внешних источников |  |
| Сцена 4 | Оплата | Из внешних источников |  |
| Сцена 5 | Выход | Из внешних источников |  |

Таблица 18. Фрейм-образец "Заказ Василия"

|  |  |  |  |
| --- | --- | --- | --- |
| **ЗАКАЗ ВАСИЛИЯ (ЗАКАЗ)** | | | |
| **Имя слота** | **Значение слота** | **Способ получения значения** | **Демон** |
| Перечень работ | Диагностика, замена термопасты | Из внешних источников | IF-ADDED (изменение слота “Цены”) |
| Цены | 1000, 500 | Присоединенная процедура | IF-ADDED (изменение слота “Сумма заказа”) |
| Сумма заказа | 1500 | Присоединенная процедура |  |
| Принял | Петр | Из внешних источников |  |
| Сделал | Иван | Из внешних источников |  |

Таблица 19. Фрейм-образец "Оплата Василия"

|  |  |  |  |
| --- | --- | --- | --- |
| **ОПЛАТА ВАСИЛИЯ (ОПЛАТА)** | | | |
| **Имя слота** | **Значение слота** | **Способ получения значения** | **Демон** |
| Вид оплаты | Безналичный | Из внешних источников |  |
| Заказ | Заказ Василия | Из внешних источников | IF-ADDED (изменение слота “Оплатил”) |
| Оплатил | Василий | Присоединенная процедура |  |

**

*Рис. 2. Схема Фреймовой модели предметной области «Проблемы с ПЭВМ».*

# Семантическая модель

Построить *сетевую модель* представления знаний в предметной области «*Проблемы с ПЭВМ*».

3.1 Описание процесса решения

Для построения сетевой модели представления знаний необходимо выполнить *следующие* шаги:

1. Определить абстрактные объекты и понятия предметной области, необходимые для решения поставленной задачи. Оформить их в виде вершин.
2. Задать свойства для выделенных вершин, оформив их в виде вершин, связанных с исходными вершинами атрибутивными отношениями
3. Задать связи между этими вершинами, используя функциональные, пространственные, количественные, логические, временные, атрибутивные отношения, а также отношения типа «являться наследником» и «являться частью».
4. Добавить конкретные объекты и понятия, описывающие решаемую задачу. Оформить их в виде вершин, связанных с уже существующими отношениями типа «являться экземпляром», «есть».
5. Проверить правильность установленных отношений.

3.2 Решение

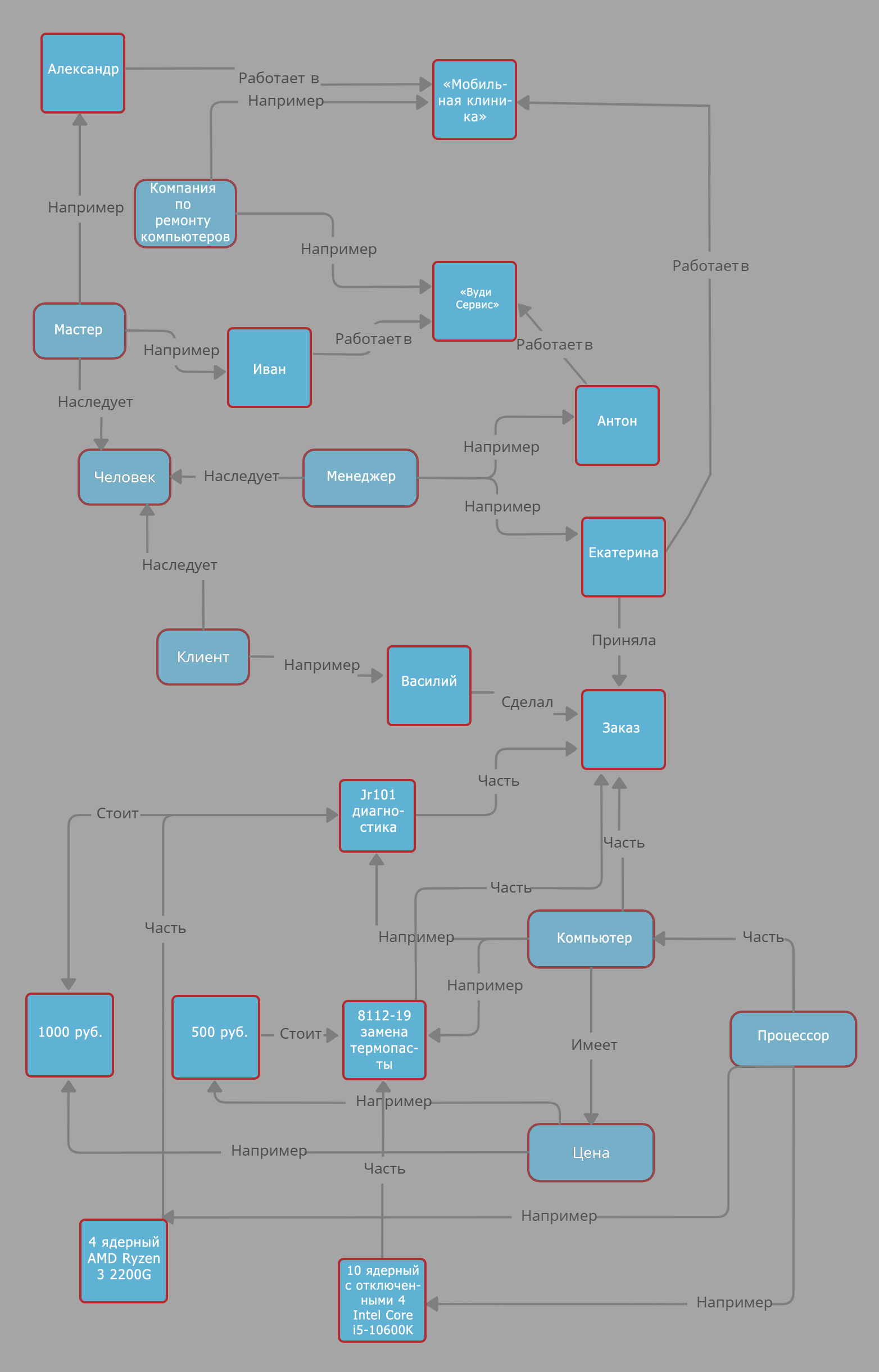
1. Ключевые понятия данной предметной области – компьютер, тот, у кого возникли проблемы с компьютером (клиент) и те, кто его диагностируют и ремонтируют (мастера). У обслуживающего персонала и клиентов есть общие характеристики, поэтому целесообразно выделить общее абстрактное понятие - человек. Некоторая компания, ремонтирующая компьютеры. Воспользуются услугами ремонта мастера, клиенты. Компьютер состоит из комплектующих. У комплектующих есть свои характеристики.

Исходя из этого, вершины графа будут следующими: «Компьютер», «Процессор», «Человек», «Мастер», «Менеджер», «Клиент», «Компания по ремонту компьютеров».

1. У этих объектов есть определенные свойства и атрибуты. Например, компании располагаются по определенным адресам, каждый компьютер из перечня имеет свои дефекты. Поэтому добавим вершины «Адрес» и «Цена».
2. Определим для имеющихся вершин отношения и их типы, используя таблицу.
3. Добавим знание о конкретных фактах решаемой задачи. Пусть имеется две компании по ремонту и обслуживанию компьютеров: «Вуди Сервис» и «Мобильная клиника», в первой работают Менеджер Антон и Мастер Иван, а во втором Менеджер Екатерина и Мастер Александр. Клиент Василий решил пойти в компанию «Вуди Сервис» и сделал заказ на диагностику и обслуживание компьютера: обслуживание компьютера Jr101 за 1000 р. - диагностика,замена термопасты за 500 р**.** Также известны адреса этих компании и их специфика. Исходя из этого, добавим соответствующие вершины в граф и соединим их функциональными отношениями и отношениями типа «например, или являться экземпляром». Полученный в результате граф изображен на рис. 7.
4. Осуществим проверку установленных связей. Например, возьмем вершину «Компьютер» и пройдем по установленным связям. Получаем следующую информацию: компьютер является частью заказа, примерами компьютеров могут служить Jr101, 8112-19, а проблемы, связанные с железом, с операционной системой.

Для *получения* ответа на какой-либо вопрос по этой задачи, необходимо найти соответствующий участок сети и, используя связи, получить результат.

1. Например, вопрос «Какова цена заказа Василия (сколько Василий заплатил за заказ)?» Из запроса понятно, что необходимо найти следующие вершины: «Цена», «Василий» и «Заказ» или «Заказ Василия». Часть семантической сети, находящаяся между этими вершинами, содержит ответ, а именно, частью заказа Василия являются обслуживание компьютера Jr101 за 1000 р. - диагностика,замена термопасты за 500 р**.** Больше информации о заказе Василия в модели нет, поэтому делаем вывод, что- Василий заплатил 1500 р.



*Рис. 3. Схема Семантической модели предметной области «Проблемы с ПЭВМ».*

Вывод: в результате исследования семантической, фреймовой, продукционной моделей представления знаний для разработки экспертной системы была выбрана продукционная модель, потому что:

1. Она легко может быть описана на современных языках программирования высокого уровня;
2. Правила продукционной модели можно изменять без проблем, не изменяя бизнес-логику экспертной системы.

4. Экспертная система

Разработанная экспертная система использует механизм вывода обратной цепочки рассуждений. Для его реализации был использован алгоритм поиска в глубину. По известной основной переменной начинается поиск в результатах правил, далее по переменным в условиях этих правил ищутся другие правила с таким же результатом, и так далее рекурсивно. Если переменной не окажется в результате какого-либо правила, она запрашивается у пользователя. Если правило окажется истинным, результат сохраняется, а само правило удалится. Если правило ложное, то правило удаляется, и поиск продолжится. Перебор правил продолжается до тех пор, пока не будет найдена основная переменная, или стек переменных окажется пустым. Для работы системы необходимо задать начальную переменную.

* 1. Переменные экспертной системы:

ПК – персональный компьютер (*запускается; не запускается*);

ПР – проводка (*есть проблемы; нет проблем*)

В – вывод (*исправен; не исправен*)

ОС – операционная система (*начинает загрузку; не начинает загрузку*)

ПСЖ – проблема с железом (*есть; нет*)

ОЗУ – оперативно запоминающее устройство (*исправен; не исправен*)

ВК – видеокарта (*исправен; не исправен*)

ЦП – центральный процессор (*исправен; не исправен*)

СЭ – синий экран (*есть; нет*)

СОС – статус операционной системы (*установлен; отсутствует*)

* 1. Правила экспертной системы:

1) ЕСЛИ ОС = *не начинает загрузку*, ТО ПСЖ = *есть*

2) ЕСЛИ ОС = *начинает загрузку*, ТО ПСЖ = *нет*

3) ЕСЛИ ПК = *не запускается* И ПР = *нет проблем*, ТО ПСЖ = *есть*

4) ЕСЛИ ПСЖ = *нет* И СОС = *установлен,* ТО В = *исправен*

5) ЕСЛИ ПСЖ = *есть* И ОЗУ = *исправен*, ТО В = *исправен*

6) ЕСЛИ ПСЖ = *есть* И ОЗУ = *исправен* И ВК = *не* *исправен*, ТО В = *не* *исправен*

7) ЕСЛИ ПСЖ = *есть* И ОЗУ = *исправен* И ВК = *исправен*, ТО В = *исправен*

8) ЕСЛИ ПСЖ = *есть* И ОЗУ = *исправен* И ВК = *исправен*, И ЦП = *не исправен* ТО В = *не исправен*

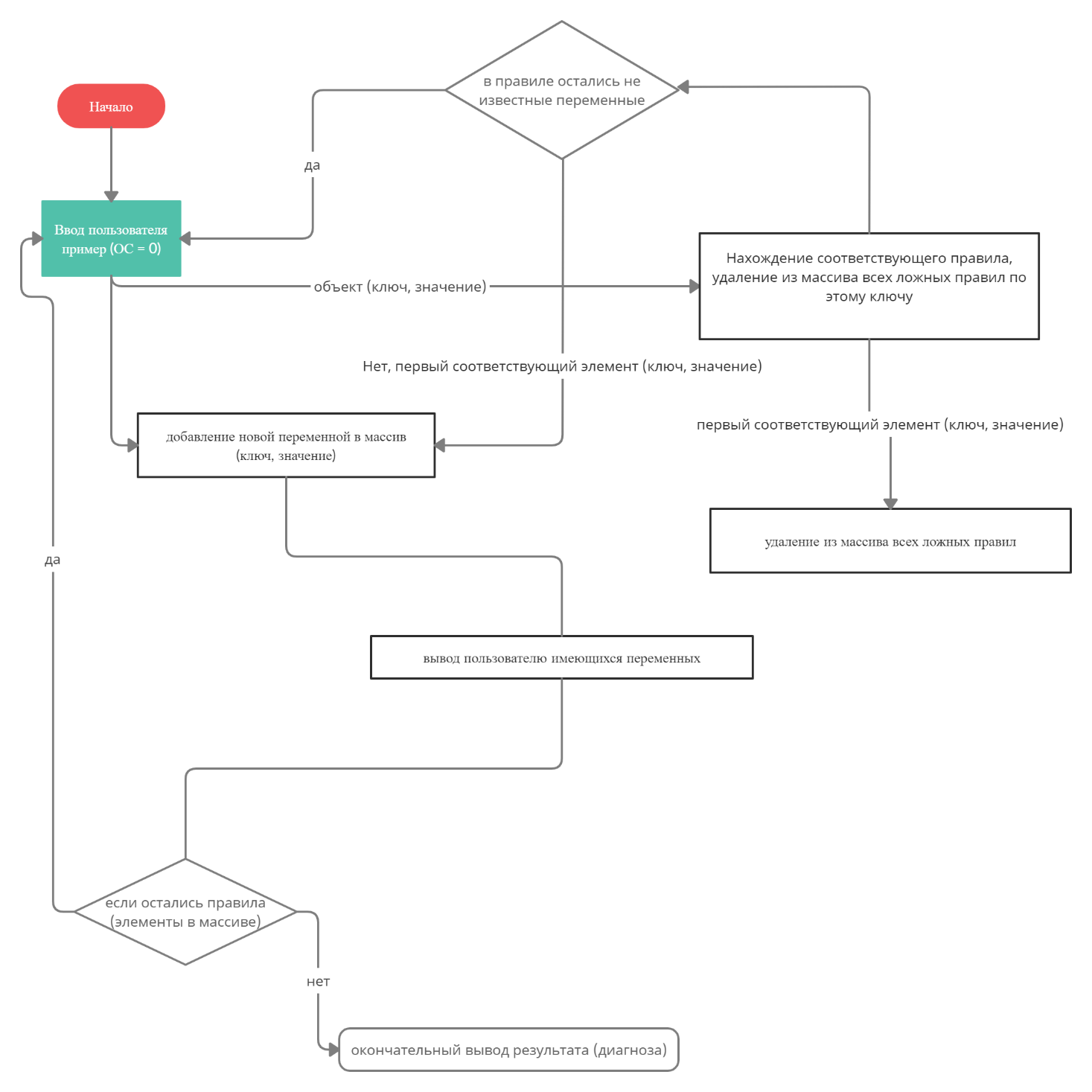
9) ЕСЛИ ПСЖ = *есть* И ОЗУ = *исправен* И ВК = *исправен*, И ЦП = *исправен* ТО В = *исправен*

10) ЕСЛИ ПСЖ = *нет* И ОС = *не начинает загрузку* И СЭ – *нет*, ТО СОС = *отсутствует*

11) ЕСЛИ ПСЖ = *есть* И ОС = *не начинает загрузку* И СЭ – *есть*, ТО СОС = *установлен*

12) ЕСЛИ ПСЖ = *есть* И ОЗУ = *исправен* И ВК = *исправен*, И ЦП = *исправен* ТО В = *исправен*

13) ЕСЛИ ПК = *не запускается* И ПР = *есть проблемы*, ОС = *не начинает загрузку*, ТО ПСЖ = *есть*



*Рис. 4. Алгоритм работы экспертной системы предметной области «Проблемы с ПЭВМ».*

* 1. Стек технологий для разработки экспертной систем

1. Язык программирования JavaScript

2. Библиотека React JS

3. Фреймворк Electron (Desktop)

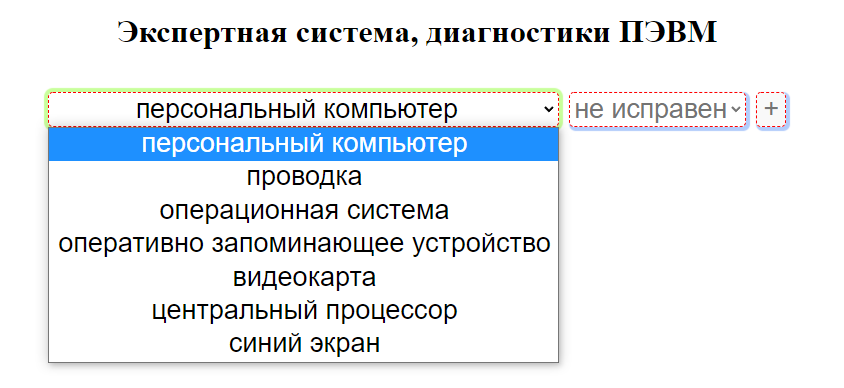
4. Среда разработки Visual Studio Code

5. Фреймворк react-native (Mobile)

6. Прослойка expo

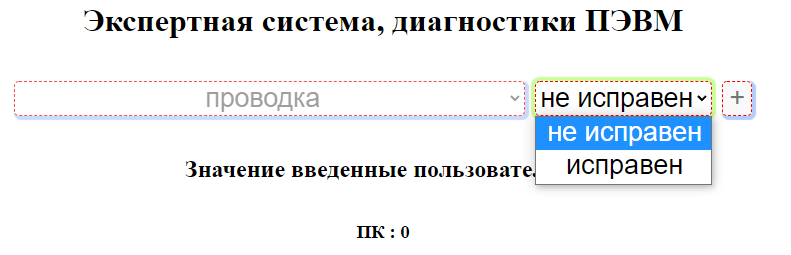
* 1. Демонстрация работы экспертной системы

В начале пользователь отвечает на вопросы ЭС указывая с какими частями ПЭВМ были проблемы. Он может выбрать из предложенного списка переменную и задает значение либо «исправен», либо «не исправен».



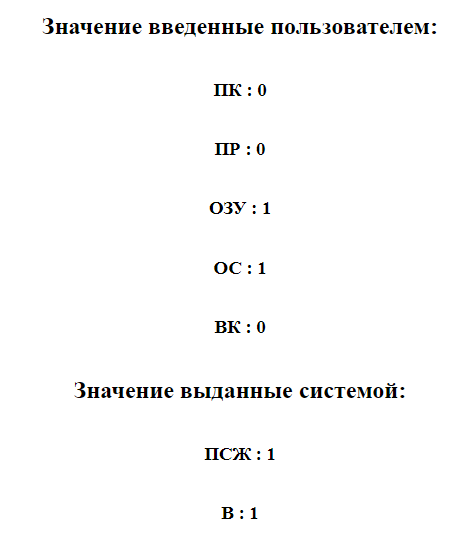
*Рис. 5. Этап выбора переменной*

Далее экспертная система запрашивает у пользователя недостающие переменные.



*Рис. 6. Ввод пользователем недостающих переменных*

После того, как пользователь ответил на все вопросы, экспертная система делает вывод.



*Рис. 7. Вывод экспертной системы*

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта были изучены продукционную, семантическую, фреймовую модели представления знаний. Для каждой из них были выделены достоинства и недостатки, на основе сделанного вывода по достоинствам и недостаткам был выбран наиболее эффективная модель для предметной области «Проблемы с ПЭВМ». Была изучена данная предметная область, выделены основные переменные, которые необходимы для работы экспертной системы и переменные, которую может вернуть экспертная система для помощи мастеру в выявлении и устранении проблем с ПЭВМ.

Для разработки экспертной системы был выбран механизм вывода прямой цепочки рассуждений. Для его реализации был использован алгоритм поиска в глубину. То есть по известной переменной начинается поиск в переменной в массиве объектов, если переменных недостаточно для того, чтобы сделать вывод система просит у пользователя еще переменную. Если правило оказалось ложным удаляется из массива, а если истинным, то переменные сохраняются, а само правило удаляется. Перебор правил продолжается до тех пор, пока не будет найдена основная переменная, или стек переменных не окажется пустым. Для работы системы необходимо задать начальную переменную.

ЛИТЕРАТУРА

1. Достоинства и недостатки фреймового представления [Электронный ресурс]. – Режим доступа: https://studme.org/244092/informatika/dostoinstva\_nedostatki\_freymovogo\_predstavleniya. – дата обращения: 14.03.2022.

2. Достоинства и недостатки семантических сетей [Электронный ресурс]. – Режим доступа: https://mydocx.ru/3-46450.html. – дата обращения: 14.04.2022.

3. Достоинства и недостатки продукционных систем - Системы искусственного интеллекта. Часть 1 [Электронный ресурс]. – Режим доступа: https://ozlib.com/1004850/tehnika/dostoinstva\_nedostatki\_produktsionnyh\_sistem. – дата обращения: 14.03.2022.

4. Классификация экспертных систем [Электронный ресурс]. – Режим доступа: http://www.aiportal.ru/articles/expert-systems/classification.html. – дата обращения: 14.03.2022.

5. Классификация экспертных систем [Электронный ресурс]. – Режим доступа: https://studme.org/117631/informatika/klassifikatsiya\_ekspertnyh\_sistem. – дата обращения: 14.03.2022.

6. Продукционная модель знаний [Электронный ресурс]. – Режим доступа: http://www.aiportal.ru/articles/knowledge-models/production-model.html. – дата обращения: 14.03.2022.

7. Продукционная модель представления знаний [Электронный ресурс]. – Режим доступа: https://spravochnick.ru/informatika/produkcionnaya\_model\_predstavleniya\_znaniy/. – дата обращения: 14.03.2022.

8. Продукционная модель [Электронный ресурс]. – Режим доступа: https://itteach.ru/predstavlenie-znaniy/produktsionnaya-model-predstavleniya-znaniy. – дата обращения: 14.03.2022.

9. Продукционная модель представления знаний - Учебная и научная деятельность Анисимова Владимира Викторовича [Электронный ресурс]. – Режим доступа: https://sites.google.com/site/anisimovkhv/learning/iis/lecture/tema4. – дата обращения: 14.03.2022.

10. Продукционная модель представления знаний - Представление знаний [Электронный ресурс]. – Режим доступа: https://itteach.ru/predstavlenie-znaniy/produktsionnaya-model-predstavleniya-znaniy. – Дата доступа: 24.03.2021.

11. Продукционная модель представления знаний - Учебная и научная деятельность Анисимова Владимира Викторовича [Электронный ресурс]. – Режим доступа: https://www.sites.google.com/site/anisimovkhv/learning/iis/lecture/tema4. – дата обращения: 14.03.2022.

12. Семантическая модель | matematicus.ru [Электронный ресурс]. – Режим доступа: https://www.matematicus.ru/teoriya-sistem/semanticheskaya-model. – дата обращения: 13.04.2022.

13. Семантическая модель | matematicus.ru [Электронный ресурс]. – Режим доступа: https://www.matematicus.ru/. – дата обращения: 14.03.2022.

14. Семантические сети и концептуальные графы - Учебная и научная деятельность Анисимова Владимира Викторовича [Электронный ресурс]. – Режим доступа: https://www.sites.google.com/site/anisimovkhv/learning/iis/lecture/tema5. – дата обращения: 14.03.2022.

15. Фреймовая модель знаний [Электронный ресурс]. – Режим доступа: https://intellect.icu/frejmovaya-model-znanij-3890. – дата обращения: 14.03.2022.

16. Фреймы - Учебная и научная деятельность Анисимова Владимира Викторовича [Электронный ресурс]. – Режим доступа: https://www.sites.google.com/site/anisimovkhv/learning/iis/lecture/tema6. – дата обращения: 14.03.2022.

17. Экспертные системы - Информационные технологии и системы

[Электронный ресурс]. – Режим доступа: https://studref.com/384657/informatika/ekspertnye\_sistemy. – дата обращения: 14.03.2022.

18. Экспертные системы [Электронный ресурс]. – Режим доступа: https://works.doklad.ru/view/\_nhksByjk6k/all.html. – дата обращения: 14.03.2022.

19. Экспертные системы [Электронный ресурс]. – Режим доступа: https://revolution.allbest.ru/programming/00357918\_0.html. – дата обращения: 14.03.2022.

20. Экспертные системы [Электронный ресурс]. – Режим доступа: https://russianblogs.com/article/10321992231/. – дата обращения: 14.03.2022.

ПРИЛОЖЕНИЕ

1. Основная логика и код:

*import* { useEffect, useState, useRef } *from* "react";

*import* { category, regulation, abbreviations } *from* "./components/data";

*import* './styles/css/app.css'

*let* tRULE = {}

*function* App() {

*const* [flag, setFlag] = useState(true)

*const* [newAnsewerUser, setNewAnsewerUser] = useState([])

*const* [newAnsewerApp, setNewAnsewerApp] = useState([])

*const* [sel, setSel] = useState([])

*const* [rule, setRule] = useState({})

*const* inputAb = useRef()

*const* inputValue = useRef()

*const* \_regulation = regulation

*const* \_category = Array.from(category)

  useEffect(() *=>* {

*const* arr = []

    \_regulation.forEach(array *=>* {

      Object.keys(array).forEach(element *=>* {

        arr.push(<h3 *key*={Math.*random*()}>{element} : {array[element]}</h3>)

      })

      arr.push(<h3 *key*={Math.*random*()}>--------------</h3>)

    })

    category.forEach(el *=>* {

*const* [\_k, \_v] = Object.entries(el)[0]

*if* (\_k === "user") {

*const* \_ab = abbreviations.find(el *=>* Object.keys(el)[0] === \_v)

        setSel(oldEl *=>* [...oldEl, <option *key*={Math.*random*()} *value*={\_v}>{\_ab[\_v]}</option>])

      }

    })

  }, [\_regulation])

*const* ruleCheck = (\_allRules, rul) *=>* {

*/\*\**

*\* function takes all user rules and*

*\* and checks the current rule against*

*\*/*

*for* (*const* [\_key, \_value] of Object.entries(\_allRules)) {

*for* (*const* [rKey, rValue] of Object.entries(rul)) {

*if* (rKey === \_key)

*if* (*parseInt*(rValue) !== *parseInt*(\_value))

*return* false

      }

    }

*return* true

  }

*const* getType = (ab) *=>* {

*/\*\**

*\* function takes anabbreviation as a parameter and*

*\* returns type of abbreviation*

*\*/*

*const* index = category.findIndex(el *=>* Object.values(el).toString() === ab.toString())

*return* index > -1 *?* Object.keys(category[index]).toString() *:* false

  }

*const* checkFinish = (\_appKey, \_myRul) *=>* {

*/\*\**

*\* function takes a key and user rule and*

*\* checks whether the application is terminated or not*

*\*/*

*const* \_delArr = []

*if* (\_regulation.length > 0) {

*for* (*let* i = 0; i < \_category.length; i++) {

*const* [\_key, \_value] = Object.entries(\_category[i])[0]

*if* (\_key === "user" || \_appKey === \_value) \_delArr.push(i)

      }

    }

    \_delArr.sort((a, b) *=>* b - a)

    \_delArr.forEach(i *=>* \_category.splice(i, 1))

*const* key = Object.values(\_category[0])

*for* (*let* i = 0; i < \_regulation.length; i++) {

*const* arrKeyRegulations = Object.keys(\_regulation[i])

*if* (arrKeyRegulations.findIndex(e *=>* e == key) !== -1) {

        tRULE = \_regulation[i]

        \_regulation.splice(i, 1)

*break*;

      }

    }

    checkFillingRule(\_myRul)

  }

*const* getAppKey = (\_appKey, \_myRul) *=>* {

*/\*\**

*\* function takes a key and user rule and*

*\* push new application key and value*

*\*/*

*if* (ruleCheck(\_myRul, tRULE)) {

*const* \_obj = Object.entries(tRULE)

*const* oKey = \_obj[\_obj.length - 1][0]

*const* oValue = \_obj[\_obj.length - 1][1]

*const* oBject = { [oKey]: oValue }

      setRule(old\_o *=>* ({ ...old\_o, ...oBject }))

      setNewAnsewerApp(item *=>* [...item, <h3 *key*={Math.*random*()}>{oKey} : {oValue}</h3>])

      checkFinish(\_appKey, { ...\_myRul, ...oBject })

    } *else* {

*for* (*let* i = 0; i < \_regulation.length; i++) {

*const* arrKeyRegulations = Object.keys(\_regulation[i])

*if* (arrKeyRegulations.findIndex(e *=>* e == \_appKey) !== -1) {

*if* (ruleCheck(\_myRul, \_regulation[i])) {

*const* \_obj = Object.entries(\_regulation[i])

*const* oKey = \_obj[\_obj.length - 1][0]

*const* oValue = \_obj[\_obj.length - 1][1]

*const* oBject = { [oKey]: oValue }

            setRule(old\_o *=>* ({ ...old\_o, ...oBject }))

            setNewAnsewerApp(item *=>* [...item, <h3 *key*={Math.*random*()}>{oKey} : {oValue}</h3>])

            checkFinish(\_appKey, \_myRul)

*break*

          }

        }

      }

    }

  }

*function* onInput(\_key) {

*/\*\**

*\* function takes a key and*

*\* enables the user to select the value of a variable*

*\*/*

    inputAb.current.value = \_key

    inputAb.current.disabled = true

*// inputValue.current.value = ""*

    inputValue.current.focus()

    setFlag(true)

  }

*const* seatchKey = (param) *=>* {

*/\*\**

*\* function takes a key as a parameter and*

*\* return index of the regulation that conains this key*

*\*/*

*let* \_flag = true

*let* temp

*let* \_delArr = []

*for* (*const* [key, value] of Object.entries(param)) {

*for* (*let* i = 0; i < \_regulation.length; i++) {

*const* arrKeyRegulations = Object.keys(\_regulation[i])

*if* (arrKeyRegulations.findIndex(e *=>* e == key) !== -1) {

*const* result = \_regulation[i]

*if* (result[key] === *parseInt*(value) && \_flag) {

            \_delArr.push(i)

            temp = result

            \_flag = false

          }

*else* *if* (*parseInt*(Object.values(\_regulation[i])) !== *parseInt*(value))

            \_delArr.push(i)

        }

      }

    }

    \_delArr.sort((a, b) *=>* b - a)

    \_delArr.forEach(i *=>* \_regulation.splice(i, 1))

*return* temp *?* temp *:* false

  }

*const* checkFillingRule = (\_myRul = rule) *=>* {

*/\*\**

*\* function check and filling rule*

*\*/*

*let* checkRes = ruleCheck(\_myRul, tRULE)

*do* {

      checkRes = ruleCheck(\_myRul, tRULE)

*if* (!checkRes) {

        tRULE = seatchKey(\_myRul)

      }

    } *while* (!checkRes);

*const* keys = Object.keys(tRULE)

*const* \_rL = keys.length

*for* (*let* i = 0; i < keys.length; i++) {

*const* \_key = keys[i]

*if* (Object.keys(\_myRul).findIndex(k *=>* k === \_key) === -1) {

*const* \_index = Object.keys(tRULE).findIndex((e) *=>* e === \_key)

*const* type = getType(\_key)

*if* (\_index === \_rL - 1 && type === "app") {

          getAppKey(\_key, \_myRul)

*break*;

        } *else* *if* (type === "user") {

          onInput(\_key)

*break*;

        }

      }

    }

  }

*const* searchRegulations = (arrKeys) *=>* {

*/\*\**

*\* function takes an array of objects as a parameter and*

*\* returns number (index) of the regulation that contains this key(s)*

*\*/*

    setRule(old\_o *=>* ({ ...old\_o, ...arrKeys }))

*if* (Object.keys(tRULE).length === 0) {

*const* res = seatchKey(arrKeys)

      tRULE = res

    }

    checkFillingRule({ ...rule, ...arrKeys })

  }

*const* calculate = () *=>* {

    setFlag(false)

*const* ab = inputAb.current.value.toUpperCase()

*const* val = inputValue.current.value

    console.clear()

    setNewAnsewerUser(item *=>* [...item, <h3 *key*={Math.*random*()}>{ab} : {val}</h3>])

    searchRegulations({ [ab]: val })

  }

*return* (

    <div *className*="app">

      {flag &&

        <div *className*="center">

          <h1>Экспертная система, диагностики ПЭВМ</h1>

          <h2 *className*="abTitle">

            <select *ref*={inputAb} *className*="select\_menu text\_center" >

              {sel}

            </select>

            {*/\* <input placeholder="Аб." className="ab" ref={inputAb} /> \*/*}

            {*/\* <input placeholder={"значение"} ref={inputValue} className="select\_menu" /> \*/*}

            <select *ref*={inputValue} *className*="select\_menu big">

              <option *value*={0}>не исправен</option>

              <option *value*={1}>исправен</option>

            </select>

            <button *onClick*={() *=>* calculate()} *className*="select\_menu">+</button>

          </h2>

        </div>

      }

      {

        newAnsewerUser.length > 0 && (

          <div *className*="center">

            <h2>Значение введенные пользователем:</h2>

            {newAnsewerUser}

          </div>

        )

      }

      {

        newAnsewerApp.length > 0 && (

          <div *className*="center">

            <h2>Значение выданные системой:</h2>

            {newAnsewerApp}

          </div>

        )

      }

    </div>

  );

}

*export* *default* App;

Вспомогательные массивы data.js:

*export* *const* regulation = [

    { "ОС": 0, "ПСЖ": 1 },

    { "ОС": 1, "ПСЖ": 0 },

    { "ПК": 0, "ПР": 0, "ПСЖ": 1 },

    { "ПСЖ": 0, "СОС": 1, "В": 1 },

    { "ПСЖ": 1, "ОЗУ": 1, "В": 1 },

    { "ПСЖ": 1, "ОЗУ": 1, "ВК": 1, "В": 0 },

    { "ПСЖ": 1, "ОЗУ": 1, "ВК": 1, "В": 1 },

    { "ПСЖ": 1, "ОЗУ": 1, "ВК": 1, "ЦП": 0, "В": 0 },

    { "ПСЖ": 1, "ОЗУ": 1, "ВК": 1, "ЦП": 1, "В": 1 },

    { "ПСЖ": 0, "ОС": 0, "СЭ": 0, "СОС": 0 },

    { "ПСЖ": 1, "ОС": 0, "СЭ": 1, "СОС": 1 },

    { "ПСЖ": 1, "ОЗУ": 1, "ВК": 1, "ЦП": 1, "В": 1 },

    { "ПК": 0, "ПР": 1, "ОС": 0, "ПСЖ": 1 },

]

*export* *const* abbreviations = [

    { "ПК": "персональный компьютер" },

    { "ПР": "проводка" },

    { "В": "вывод" },

    { "ОС": "операционная система" },

    { "ПСЖ": "проблема с железом" },

    { "ОЗУ": "оперативно запоминающее устройство " },

    { "ВК": "видеокарта" },

    { "ЦП": "центральный процессор" },

    { "СЭ": "синий экран" },

    { "СОС": "статус операционной системы" },

]

*export* *const* category = [

    { "app": "В" },

    { "app": "СОС" },

    { "app": "ПСЖ" },

    { "user": "ПК" },

    { "user": "ПР" },

    { "user": "ОС" },

    { "user": "ОЗУ" },

    { "user": "ВК" },

    { "user": "ЦП" },

    { "user": "СЭ" },

]