Министерство науки и высшего образования Российской Федерации
Федеральное государственное образовательное учреждение высшего образования
«Уральский федеральный университет имени первого Президента России Б.Н.Ельцина»
Институт технологий открытого образования
Программная инженерия

ОТЧЁТ

экспертная система продукционного типа для выявления опасного состояния системы

Интеллектуальные системы

‘

Преподаватель:
Лимановская О.В.
Студент: Бредихин Б.А.
Группа: ФО-480006д
Дата: 9 января 2022 г.

Екатеринбург
2021

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Часто возникает задача выявления аномалий [1], например, в медицине — врач, по параметрам пациента, может сделать вывод о его состоянии здоровья. При этом нормальные параметры, а также варианты нормы и действия, которые необходимо выполнить при тех или иных отклонениях от нормы, как правило, заранее известны.

Цель работы: создание экспертной системы с интерфейсами пользователя и эксперта, которая сможет выявлять опасное состояние системы и рекомендовать действия для приведения системы в нормальное состояние.

1 АРХИТЕКТУРА СИСТЕМЫ

Предполагается следующая архитектура системы (рис. 1.1).

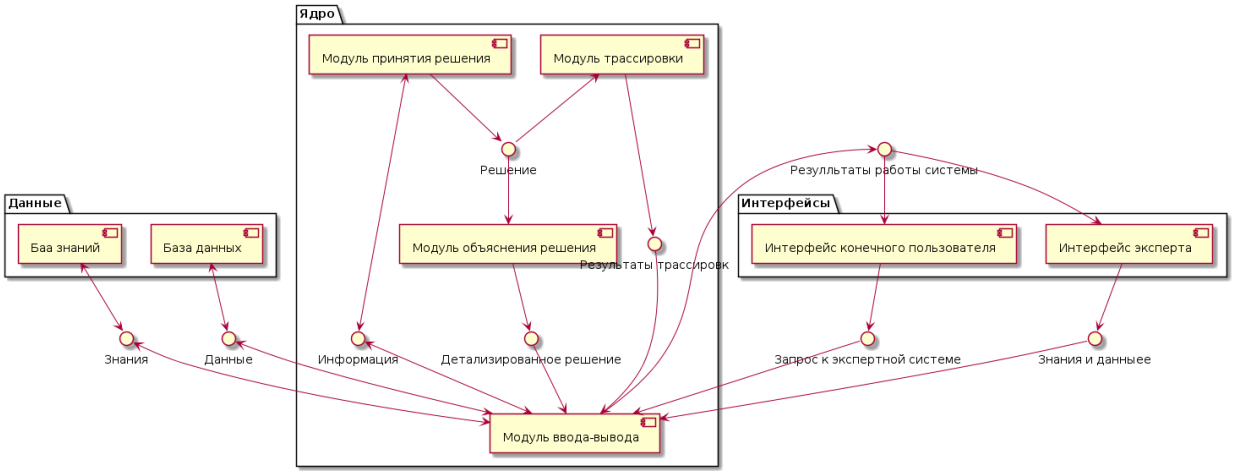

Рис. 1.1 – Архитектура интеллектуальной системы

Для построения интерфейсов пользователя будет использоваться tkinter, база данных будет представлена в виде словаря Python, а база знаний будет храниться в JSON.

Работа системы основана на очереди сообщений.

1.1 База знаний и интерфейс эксперта

Правила задаются в виде пар (антецедент, консеквент).

Причём антецедент должен быть представлен в виде конъюнктов элементарных высказываний. Элементарные высказывания задаются тройкой (переменная, оператор сравнения, переменная или константа), вместе с правилом будет храниться автоматический список используемых переменных для ускорения поиска соответствующих правил.

Консеквент будет содержать список пар (переменная, значение). При истинном значении антецедента в модуль трассировки будет отправляться правило вместе с текущими значениями переменных. Пользователю будут отображаться вычисленные значения целевых переменных с возможными рекомендациями. Также в консеквенте может быть введена новая переменная

на основе имеющихся в базе данных значений (выражения записываются в lisp-подобной нотации).

Для облегчения парсинга имена переменной должны начинаться со знака $

Минимальный пример базы знаний:

```
{
        ”rules”: [
        {
                ”antecedent”: [
                [”$t”, ”>”, 36.6]
                ],
                ”consequent”: [
                ”result”,
                ”температура повышена”
                ],
                ”recommendation”: ”Снизить температуру”
        },
        {
                ”antecedent”: [
                [”$t”, ”<”, 35.5],
                [”$P_low”, ”<”, 120]
                ],
                ”consequent”: [
                ”$state”,
                {
                        ”type”: ”str”,
                        ”description”: ”Состояние”
                }
                ],
                ”recommendation”: ”Низкие температура и давление”
        },
        {
                ”antecedent”: [
                [”$t”, ”<”, 34.5],
                [”$P_low”, ”<”, 120]
                ],
```

```
                "consequent": [
                "$state",
                ["+", "$t", ["*", 2, "$P_low"]]
                ],
                "recommendation": "Низкие температура и давление"
        }
        ],
        "input_variables": [
        {
                "name": "$t",
                "type": "float",
                "description": "Текущая температура тела
   человека"
        },
        {
                "name": "$P_low",
                "type": "int",
                "description": "Нижне давление"
        }
        ]
}
```

1.2 База данных

Для хранения базы данных используется словарь Python, хранящий пары (имя переменной, значение).

Так как используется только один экземпляр базы данных на всю экспертную систему, используется паттерн «Одиночка» [2].

1.3 Модуль принятия решения

Экземпляр модуля принятия решения создаётся в ядре. Затем анализируется база знаний и отправляется запрос в модуль ввода-вывода для получения начальных данных.

После заполнения базы знаний модуль в бесконечном цикле итерирует по правилам. Если на текущей итерации не выполнилось ни одно правило, то работа алгоритма останавливается и отправляется сообщение об отсутствии подходящих правил.

Если значение антецедента истинно вычисляется консеквент с помощью обхода его дерева в глубину. В модуль трассировки отправляется сработавшее правило и значения всех используемых в нём переменных, при наличии добавляется экспертный комментарий.

1.4 Модуль ввода-вывода

Модуль ввода-вывода представлен абстрактным классом со следующими методами: чтение переменной, вывод сообщения.

1.4.1 Интерфейс пользователя

Разработан web клиент в виде чата, который общается с системой по протоколу websocket

1.5 Ядро

Ядро системы представляет собой класс для связи других элементов системы.

При инициализации ядро принимает путь к файлу базы знаний и загружает её в память. Затем подаёся сигнал модулю ввода-вывода о загрузке начальных данных. А после загрузки начальных данных сигнал о начале логического вывода.

2 ОПИСАНИЕ РАБОТЫ СИСТЕМЫ

2.1 Со стороны конечного пользователя

При запуске системы в память загружается база знаний, в которой указаны правила и список начальных переменных. Затем пользователю предлагается вести значения этих переменных.

После ввода значений система делает несколько проходов по базе правил, возможно, создавая при этом вспомогательные переменные и давая экспертные рекомендации. При достижении правила с меткой «`result`» система делает вывод и завершает свою работу.

2.2 Со стороны эксперта

Эксперт может:

- Просмотреть правила
- Добавить правила
- Изменить правила
- Удалить правила

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Выявление аномалий // Википедия. 2020.

2. Design patterns: elements of reusable object-oriented software / ed. Gamma E. Reading, Mass: Addison-Wesley, 1995. 395 p.