Федеральное государственное бюджетное образовательное учреждение
высшего образования
Национальный исследовательский университет «МЭИ»

Методические указания по курсовому проекту по дисциплине «Программное обеспечение интеллектуальных систем»

Автор:

Доцент кафедры ПМ П.Р. Варшавский

Москва 2017 НИУ «МЭИ»

Оглавление

1 Курсовой проект

Курсовой проект (КП) является самостоятельной научно-исследовательской работой, содержащей результаты теоретических, расчетных, аналитических и экспериментальных исследований по учебной дисциплине. КП выполняется в процессе обучения для решения следующих задач:

- закрепление и более глубокое усвоение теоретических знаний и практических навыков в применении методов для решения конкретных задач;
- развитие самостоятельности при выборе методов достижения цели и творческой инициативы при решении конкретных задач.

В процессе курсового проектирования подготавливается пояснительная записка. Материалы, разработанные в рамках КП, предоставляются на защиту в электронном и печатном виде.

КП оценивается по следующим параметрам: содержание, оформление работы, защита в форме презентации в программе *Microsoft Office PowerPoint*, демонстрация работы реализованного программного приложения.

2 Структура пояснительной записки к КП

Текст пояснительной записки (ПЗ) должен включать в себя следующие структурные элементы в указанной последовательности:

- титульный лист;
- задание (ТЗ);
- содержание;
- определения (основные понятия), обозначения и сокращения (при необходимости);
- введение;
- основную часть (не менее двух глав);
- заключение;
- список литературы;
- приложения.

3 Требования к структурным элементам ПЗ

Текст ПЗ должен раскрывать творческий замысел работы, постановку задачи, выбор и обоснование принципиальных решений, содержать описание методов исследования анализа, расчетов, описание проведенных экспериментальных исследований, анализ их результатов, выводы по ним. Текст ПЗ должен сопровождаться иллюстрациями (графиками, эскизами, диаграммами, схемами и т.п.).

Оформление ПЗ проводится с учетом выполнения требований ГОСТ 7.32-2001;

Рекомендуемый объем основной части КП составляет 20–60 страниц (без приложений).

Текст ПЗ оформляется на русском языке.

Каждый основной структурный элемент ПЗ следует начинать с нового листа.

Название структурного элемента в виде заголовка записывают полужирным шрифтом строчными буквами, начиная с первой прописной.

Отчет по КП должен быть сшит (сброшюрован) и иметь обложку.

Текст ПЗ к КП представляется на бумажном и электронном носителе.

Защита КП происходит в форме презентации (10 мин.), выполненной в программе *Microsoft Office PowerPoint*, и представляется по окончании презентации в электронном варианте.

Текст должен содержать ссылки на литературу в квадратных скобках (в скобках пишется номер работы в соответствии с ее расположением в списке литературы).

КП выполняется на основе **индивидуального задания (ТЗ)**, содержащего требуемые для решения поставленных задач исходные данные, обеспечивающие возможность реализации накопленных знаний в соответствии с уровнем профессиональной подготовки студента. Руководитель совместно со студентом формирует задание, соответствующее тематике КП, которое студент оформляет в соответствии с требованиями.

Содержание ПЗ к КП включает введение, заголовки всех разделов, подразделов, пунктов (если они имеют наименование), заключение, список литературы, наименования приложений с указанием номеров страниц, с которых начинаются эти элементы.

Материалы, представляемые на электронных носителях, должны быть перечислены в содержании с указанием вида носителя, обозначения и наименования документов, имен и форматов соответствующих файлов, а также места расположения носителя в ПЗ.

Структурный элемент **«Определения»** содержит определения, необходимые для уточнения или уставновления терминов, используемых в работе.

Перечень определений начинают со слов «В данной работе применены следующие термины с соответствующими определениями». Определение должно быть оптимально кратким и состоять из одного предложения. При этом дополнительные пояснения приводят в примечаниях.

Если в ПЗ необходимо использовать обозначения или сокращения, то оформляется структурный элемент **«Обозначения и сокращения»**, содержащий перечень обозначений и сокращений, применяемых для данной работы. Запись обозначений и сокращений в этом элементе приводят в порядке их появления в тексте с необходимой расшифровкой и пояснениями. При этом:

- сокращения в виде аббревиатур приводят после термина и отделяют от него точкой с запятой;
- сокращения в виде краткой формы термина приводят после термина в скобках;
- условные обозначения приводят после термина, при этом, после условных обозначений величин приводят обозначения единиц величин, которые отделяют запятой.

Допускается определения, обозначения и сокращения приводить в одном структурном элементе **«Определения, обозначения и сокращения»**.

В тексте документа допускается приводить без расшифровки общепринятые сокращения, установленные в национальных стандартах и правилами русской орфографии: ЭВМ, НИИ, АСУ, с. – страница, т. е. – то есть, т. д. — так далее; т. п. — тому подобное; и др. — и другие; в т. ч. — в том числе; пр. — прочие; т. к. — так как; г. — год; гг. — годы; мин. — минимальный; макс. — максимальный; шт. — штуки; св. — свыше; см. — смотри; включ. — включительно, час. – часы, млн. - миллион, тыс. - тысяча и др.

В ПЗ при многократном упоминании устойчивых словосочетаний могут быть дополнительно установлены сокращения, применяемые только в данном тексте. При этом полное название следует приводить при его первом упоминании в тексте, а после полного названия в скобках — сокращенное название или аббревиатуру. Сокращенное название или аббревиатура выносится на лист «Определение» или «Сокращения». При последующем упоминании употребляют сокращенное название или аббревиатуру.

В тексте ПЗ не допускается:

- применять сокращения слов, кроме установленных правилами русской орфографии, соответствующими государственными стандартами, а также в ПЗ;

- сокращать обозначения единиц физических величин, если они употребляются без цифр, за исключением единиц физических величин в таблицах и в расшифровках буквенных обозначений, входящих в формулы и рисунки.

В тексте следует избегать необоснованных (излишних) сокращений, которые могут затруднить пользование данным документом.

Во **введении** следует

- раскрыть актуальность вопросов темы;
- охарактеризовать проблему, к которой относится тема;
- изложить историю вопроса;
- дать оценку современного состояния теории и практики;
- изложить задачи в области разработки проблемы, т.е. сформулировать цель и задачи темы работы;
- перечислить методы и средства, с помощью которых будут решаться поставленные задачи;
- кратко изложить ожидаемые результаты;
- раскрыть новизну, научное и практическое значение.

Рекомендуемый объем введения – 1-4 страницы.

Содержание **основной части работы** должно отвечать ТЗ и требованиям.

Наименования разделов основной части отражают выполнение задания. Содержание и объем основной части студент и руководитель формируют совместно.

Разделы (главы) начинаются с новой страницы. Разделов (глав) должно быть не менее двух. Номер раздела (главы) пишется арабскими цифрами. Слова «Глава или раздел» не пишутся.

Подразделы следуют друг за другом на странице. Подразделов должно быть не менее двух. Номер подраздела пишется арабскими цифрами (например, 1.1).

В конце главы могут приводиться вывод по главе.

Подразделы могут содержать пункты. Номер пункта пишется арабскими цифрами (например, 1.1.1).

Заключение должно содержать краткие выводы по результатам выполненной работы, оценку полноты решения поставленных задач, рекомендации по конкретному использованию результатов работы, ее научную и практическую значимость. Заключение должно быть не мене, чем 1 страница.

В список литературы включают все источники, на которые имеются ссылки в ПЗ. Источники в списке располагают и нумеруют по алфавиту арабскими цифрами. Сведения об источниках приводятся в

соответствии с требованиями ГОСТ. Схема построения описания источников представляется в следующем виде:

1) **Фамилия И. О. авторов. Основное название книги / Сведения о редакторе (при наличии). – Город: Издательство, год издания. – Количество страниц.**
2) **Фамилия И. О. автора. Название статьи // Название журнала.– Год выпуска.– № журнала.– С. 7-11.**
3) **Фамилия И. О. автора. Название статьи // Название сайта.**

Сведения об иностранных источниках приводятся на иностранном языке.

В **приложения** выносятся: графический материал большого объема или формата, таблицы большого формата, методы расчетов, описания алгоритмов и программ задач, решаемых на ЭВМ, ксероксы статей или документов, фотографии и т.д. В них рекомендуется включать материалы иллюстрационного и вспомогательного характера.

Приложения размещают, как продолжение ПЗ, на последующих страницах и включают в общую с ПЗ сквозную нумерацию страниц. Каждое приложение должно начинаться с нового листа и иметь тематический заголовок и обозначение. В тексте КП на все приложения должны быть даны ссылки. Все приложения должны быть перечислены в содержании КП с указанием их буквенных (или цифровых) обозначений и заголовков.

4 Требования к оформлению ПЗ

Текст ПЗ должен быть выполнен на белой бумаге формата А4 (210х297 мм) с одной стороны листа с применением печатающих или графических устройств вывода ЭВМ – через 1,5 интервала, высота букв и цифр не менее 1,8 мм, цвет – черный. Рекомендуется использовать гарнитуру шрифта Times New Roman – 14, допускается Arial – 12. При печати текстового материала следует использовать двухстороннее выравнивание.

Размеры полей: левое – 30-35 мм, правое - 20 мм, верхнее и нижнее - не менее 20- 25 мм. Нумерация страниц – в правом нижнем углу арабскими цифрами.

Абзацный отступ выполняется одинаковым по всему тексту документа.

Иллюстрации, таблицы и **распечатки** с ЭВМ допускается выполнять на листах формата А3, при этом они должны быть сложены на формат А4.

Таблица помещается в тексте сразу же за первым упоминанием о ней или на следующей странице. Если формат таблицы превышает А4, то ее размещают в приложении к ПЗ. Допускается помещать таблицу вдоль длинной стороны листа документа (альбомный вариант). Таблицы, за исключением приведенных в приложении, нумеруются сквозной нумерацией арабскими цифрами по всей ПЗ. Допускается нумеровать таблицы в пределах раздела. В этом случае номер таблицы состоит из номера раздела и порядкового номера таблицы, разделенных точкой. На все таблицы приводят ссылки в тексте или в приложении (если таблица приведена в приложении).

Все **иллюстрации** (схемы, графики, рисунки, фотографические снимки, диаграммы и т. д.) именуются в тексте рисунками и нумеруются сквозной нумерацией арабскими цифрами по всей ПЗ за исключением иллюстраций в приложениях.

Допускается нумерация рисунков в пределах каждого раздела. Тогда номер иллюстрации составляется из номера раздела и порядкового номера иллюстрации, разделенных точкой (например, Рис. 2.1). На все иллюстрации должны быть даны ссылки в тексте.

Иллюстрация располагается по тексту документа сразу после первой ссылки, если она размещается на листе формата А4. Если формат иллюстрации больше А4, то ее следует помещать в приложении.

Иллюстрации, при необходимости, могут иметь наименование и пояснительные данные (подрисуночный текст).

Графики, отображающие качественные зависимости, изображаются на плоскости, ограниченной осями координат, заканчивающимися стрелками. При этом слева от стрелки оси координат и под стрелкой оси абсцисс проставляется буквенное

обозначение соответственно функции и аргумента без указания их единиц измерения. Графики, по которым можно установить количественную связь между независимой и зависимыми переменными, должны снабжаться координатной сеткой равномерной или логарифмической. Буквенные обозначения изменяющихся переменных проставляются вверху слева от левой границы координатного поля и справа под нижней границей поля. Единицы измерения проставляются в одной строке с буквенными обозначениями переменных и отделяются от них запятой.

Формулы следует выделять из текста в отдельную строку. Пояснение значений символов и числовых коэффициентов, входящих в формулу, если они не пояснены ранее в тексте, должны быть приведены непосредственно под формулой. Значение каждого символа дают с новой строки в той последовательности, в какой они приведены в формуле. Первая строка расшифровки должна начинаться со слова «где» без двоеточия после него. Формулы, следующие одна за другой и не разделенные текстом, отделяют запятой. Формулы должны приводиться в общем виде с расшифровкой входящих в них буквенных значений. Перенос формул допускается только на знаках выполняемых математических операций, причем знак в начале следующей строки повторяют. Формулы, за исключением приведенных в приложении, должны нумероваться сквозной нумерацией в пределах всей ПЗ арабскими цифрами в круглых скобках в крайнем правом положении на строке. Допускается нумерация формул в пределах раздела. В этом случае номер формулы состоит из номера раздела и порядкового номера формулы, разделенных точкой. Формулы, помещаемые в таблицах или в поясняющих данных к графическому материалу, не нумеруют. Допускается применять обозначения единиц в пояснениях обозначений величин к формулам. Помещать обозначение единиц величины в одной строке с формулами, выражающими зависимости между величинами или между их числовыми значениями, представленными в буквенной форме, не допускается. При использовании формул из первоисточников, в которых употреблены несистемные единицы, их конечные значения должны быть пересчитаны в системные единицы. Значения одного и того же параметра в пределах всей ПЗ должно выражаться в одних и тех же единицах величин. При ссылке в тексте на формулы их порядковые номера приводят в скобках.

Все листы ПЗ, включая приложения, должны иметь сквозную нумерацию. Первым листом является титульный лист. Номер листа проставляется в нижней части страницы. На титульном листе номер не проставляется.

5 Оформление демонстрационного материала

Демонстрационный материал выполняется в виде презентации в программе *Microsoft Office PowerPoint*.

Демонстрационный материал должен содержать:

- заголовок;
- необходимые изображения и надписи (рисунки, схемы, таблицы и т.д.);
- пояснительный текст.

Заголовок должен быть кратким и соответствовать содержанию демонстрационного листа.

Элементы графиков, таблиц, диаграмм (надписи, линии, условные изображения) должны выполняться в соответствии с требованиями действующих стандартов.

При оформлении демонстрационного материала в виде презентации необходимо придерживаться следующих правил:

- титульный (первый) слайд должен содержать тему доклада, сведения об авторах и возможных соавторах;
- шрифт должен быть не менее 20 пунктов;
- все слайды за исключением титульного должны иметь заглавную строку;
- на всех слайдах за исключением титульного должны быть указаны их номера.

При оформлении демонстрационного материала допускается применение цветных изображений и надписей. Принятые цифровые и цветовые обозначения должны быть расшифрованы.

Графический и текстовый материал оформляется с учетом общих требований, приведенных выше.

6 Темы КП

1. Разработка фрагмента гипертекстовой информационной системы.
2. Разработка фрагмента онтологии для интеллектуального поиска в специализированной базе данных.
3. Разработка алгоритмов наследования для обобщенной модели представления предметной области.
4. Разработка программно-инструментального средства для организации представления знаний на основе семантических сетей.
5. Разработка и реализация алгоритмов генерации учебно-тренировочных задач на основе текста учебного материала (на грамматическом уровне).
6. Разработка и реализация алгоритмов генерации учебно-тренировочных задач на основе текста учебного материала (на логическом уровне).
7. Разработка и реализация алгоритмов генерации учебно-тренировочных задач на основе реляционной базы данных.
8. Разработка и реализация алгоритмов генерации учебно-тренировочных задач на основе фрагментов текста учебного материала (перечислений, описаний).
9. Исследований и разработка моделей и алгоритмов навигации по учебному материалу в компьютерных средствах обучения.
10. Исследование возможностей нейропакета *NeuroSolutions* для реализации и обучения искусственной нейронной сети.
11. Разработка системы логического продукционного моделирования.
12. Разработка прототипа системы управления темпоральной базой данных с поддержкой временных запросов.
13. Разработка алгоритма извлечения прецедентов на основе онтологии предметной области для прототипа интеллектуальной системы поиска решения на основе прецедентов (CBR-системы).
14. Исследование и разработка методов правдоподобных рассуждений на основе прецедентов с использованием аппарата нейронных сетей.
15. Исследование и разработка методов рассуждений на основе аналогий для систем экспертного диагностирования.
16. Реализация программного модуля для обучения нейронной сети.
17. Исследование и разработка методов и моделей рассуждения на основе структурной аналогии для интеллектуальных систем.
18. Исследование и разработка методов вывода и машинного обучения на основе накопленного опыта для интеллектуальных систем.
19. Исследование и программная реализация метода Демпстера-Шефера для моделирования рассуждений в условиях неопределенности в экспертной системе медицинской диагностики.

20. Разработка программы-советчика, ориентированной на различные этапы концептуального проектирования техники и технологий (инновационной деятельности).
21. Исследование и реализация системы имитационного моделирования реального времени на основе графовых моделей.

Выбранную тему КП необходимо согласовать с руководителем и получить задание по КП (ТЗ). Согласование выполняется по электронной почте или лично.

7 Образец ПЗ к КП

Федеральное государственное бюджетное образовательное учреждение высшего образования
Национальный исследовательский университет «МЭИ»

Институт **АВТИ**
Кафедра **ПМ**

К У Р С О В О Й П Р О Е К Т
по дисциплине
«Программное обеспечение интеллектуальных систем»

Тема: Разработка алгоритма извлечения прецедентов на основе онтологии предметной области для прототипа интеллектуальной системы поиска решения на основе прецедентов (CBR-системы)

Студент Алехин Р.В. А-13м-ХХ

фамилия, и., о. *группа* *подпись*

Научный руководитель доц., к.т.н. Варшавский П.Р

должность, звание, фамилия, и., о. *подпись*

Москва 2017 НИУ «МЭИ»

Федеральное государственное бюджетное образовательное учреждение
высшего образования
Национальный исследовательский университет «МЭИ»

Институт **АВТИ**
Кафедра **ПМ**

Т Е Х Н И Ч Е С К О Е З А Д А Н И Е

К У Р С О В О Й П Р О Е К Т

по дисциплине
«Программное обеспечение интеллектуальных систем»

Тема: Разработка алгоритма извлечения прецедентов на основе онтологии предметной области для прототипа интеллектуальной системы поиска решения на основе прецедентов (CBR-системы)

Студент Алехин Р.В. А-13м-ХХ

фамилия, и., о. *группа* *подпись*

Научный руководитель доц., к.т.н. Варшавский П.Р

должность, звание, фамилия, и., о. *подпись*

Москва 2017 НИУ «МЭИ»

Обоснование актуальности темы

Тема работы затрагивает актуальную на сегодняшний день проблематику в области искусственного интеллекта, связанную с моделированием человеческих рассуждений (рассуждений «здравого смысла») в интеллектуальных системах (ИС), ориентированных на открытые и динамические предметные области.

Одним из подходов к решению указанной проблемы является использование аппарата нетрадиционных логик – индуктивных, абдуктивных, нечетких, а также методов рассуждений на основе аналогий и прецедентов.

Методы рассуждения на основе прецедентов (CBR – Case-Based Reasoning) активно применяются в динамических ИС, в системах экспертного диагностирования и машинного обучения. Рассуждения на основе прецедентов основываются на накоплении опыта и последующей адаптации решения известной задачи к решению новой. Этот подход позволяет упростить процесс принятия решений в условиях временных ограничений и наличия различного рода неопределенности в исходных данных и экспертных знаниях.

Целью работы является исследование моделей, методов и программных средств для организации представления знаний и поиска решения в ИС на основе прецедентов (CBR-системах), а также разработка алгоритма извлечения прецедентов и его программная реализация.

Исходные данные

- ОС MS Windows 7 Professional.
- Среда программирования MS Visual Studio 2010 Ultimate.
- web-онтологии по различным предметным областям.
- Язык web-онтологий OWL, язык описания данных RDF.
- Библиотеки прецедентов по различным предметным областям.
- Информационные Internet ресурсы www.raai.org, www.ijcai.org, www.wc3.org .

Подпись руководителя проекта Дата

Перечень графического материала

- Постановка задачи.
- Структура библиотек прецедентов.
- Архитектура прототипа CBR-системы.
- Блок-схема разработанного алгоритма.
- Демонстрационный пример.

Рекомендуемая литература

1. **Люгер Д.Ф.** Искусственный интеллект: стратегии и методы решения сложных проблем. Пер. с англ. – 4-е изд. –М.: Издательский дом “Вильямс”, 2003. – 864 с.
2. **Вагин В.Н., Головина Е.Ю., Загорянская А.А., Фомина М.В.** Достоверный и правдоподобный вывод в интеллектуальных системах / Под ред. В.Н. Вагина, Д.А. Поспелова. –М.: ФИЗМАТЛИТ, 2004. – 704 с.
3. **Гаврилова Т.А., Хорошевский В.Ф.** Базы знаний интеллектуальных систем. –СПб.: Питер, 2000. – 384 с.
4. **Варшавский П.Р., Еремеев А.П.** Моделирование рассуждений на основе прецедентов в интеллектуальных системах поддержки принятия решений / Искусственный интеллект и принятие решений. 2009. №2. – с. 45–47.
5. **Варшавский П.Р., Еремеев А.П.** Методы правдоподобных рассуждений на основе аналогий и прецедентов для интеллектуальных систем поддержки принятия решений / Новости искусственного интеллекта, № 3, 2006. –С. 39-62.
6. **Шилдт Г.** Полный справочник по C#.. –М.: Издательский дом “Вильямс”, 2008. – 752 с.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Тема работы затрагивает актуальную на сегодняшний день проблематику в области искусственного интеллекта, связанную с моделированием человеческих рассуждений (рассуждений «здравого смысла») в интеллектуальных системах (ИС), ориентированных на открытые и динамические предметные области.

Одним из подходов к решению указанной проблемы является использование аппарата нетрадиционных логик – индуктивных, абдуктивных, нечетких, а также методов рассуждений на основе аналогий и прецедентов [1, 2].

Методы рассуждения на основе прецедентов (CBR – Case-Based Reasoning) активно применяются в динамических ИС, в системах экспертного диагностирования, в системах машинного обучения, а также для решения задач прогнозирования, обобщения накопленного опыта, поиска решения в малоизученных предметных областях и др. [3].

Рассуждения на основе прецедентов основываются на накоплении опыта и последующей адаптации решения известной задачи к решению новой. Этот подход позволяет упростить процесс принятия решений в условиях временных ограничений и в случае возникновения различных нештатных (аномальных) ситуаций, которые могут возникнуть при наличии различного рода «НЕ-факторов» (неполноты, противоречивости, неопределенности и др.) в исходных данных и экспертных знаниях.

Использование онтологического подхода позволяет задать сложную структуру прецедента, включающую данные разных типов, естественность представления структурированных знаний и достаточно простое их обновление в относительно однородной среде. Последнее свойство особенно важно для ИС, ориентированных на открытые и динамические предметные области.

Целью курсового проекта является разработка и программная реализация алгоритма извлечения прецедентов с учетом онтологии предметной области для прототипа интеллектуальной системы поиска решения на основе прецедентов (CBR-системы).

Решаемые задачи:

- Исследование методов поиска решения на основе прецедентов.
- Разработка алгоритма извлечения прецедентов на основе онтологии предметной области из библиотеки прецедентов системы.
- Программная реализация модуля поиска решения с использованием языка C#.
- Разработка демонстрационного примера.

1 РАССУЖДЕНИЯ НА ОСНОВЕ ПРЕЦЕДЕНТОВ

1.1 Основные концепции

Прецедент определяется как случай, имевший место ранее и служащий примером или оправданием для последующих случаев подобного рода. Рассуждение на основе прецедентов **(**CBR – Case-Based Reasoning**)** является подходом, позволяющим решить новую (неизвестную) задачу, используя или адаптируя решение уже известной задачи.

Как правило, методы рассуждения на основе прецедентов включают в себя четыре основных этапа, образующие так называемый CBR-цикл (Рис 1) [3]:

- *извлечение* наиболее соответствующего (подобного) прецедента (или прецедентов) для сложившейся ситуации из библиотеки прецедентов;
- *повторное использование* извлеченного прецедента для попытки решения текущей проблемы (задачи);
- *пересмотр и адаптация* в случае необходимости полученного решения в соответствии с текущей проблемой (задачей);
- *сохранение* вновь принятого решения как части нового прецедента.

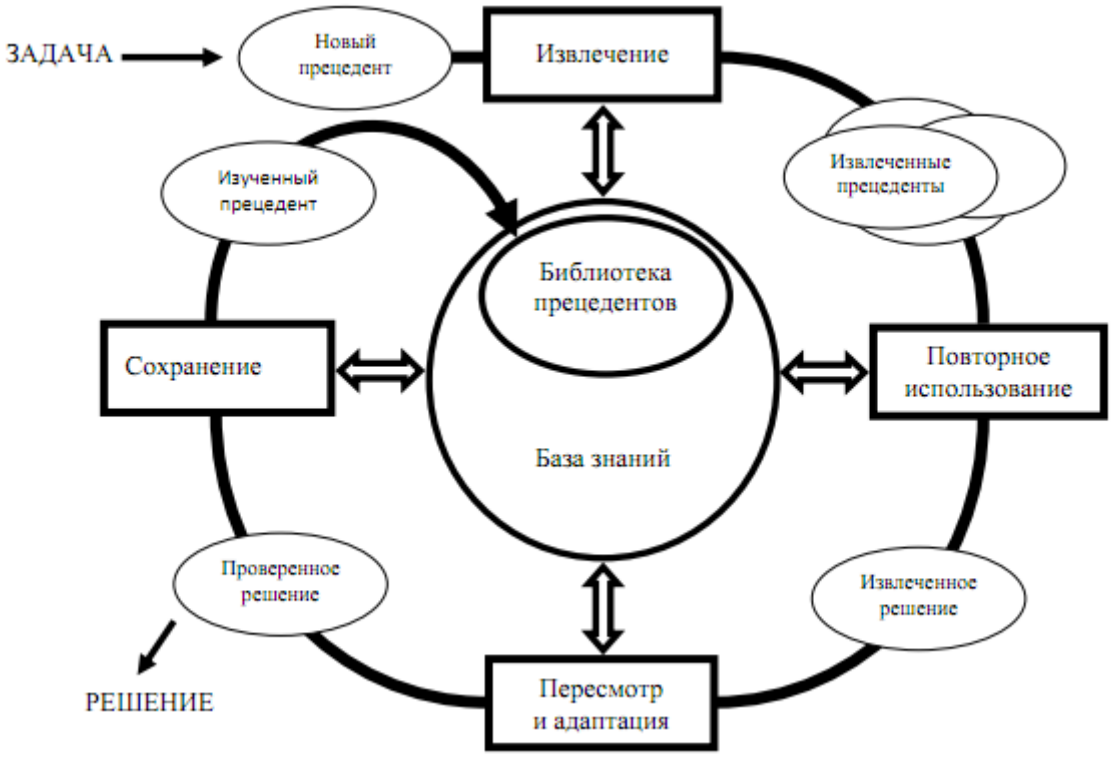


Рис. 1. Цикл рассуждения по прецедентам (CBR цикл)

Основная цель использования аппарата прецедентов в рамках ИС заключается в выдаче готового решения ЛПР для текущей ситуации на основе прецедентов, которые уже имели место в прошлом.
Преимущества рассуждений на основе прецедентов:

- возможность напрямую использовать опыт, накопленный системой, без интенсивного привлечения эксперта в той или иной предметной области;
- возможность сокращения времени поиска решения за счет использования уже имеющегося решения для подобной задачи;
- возможность исключения повторного получения ошибочного решения;
- отсутствие необходимости углубленного изучения и использования всех имеющихся знаний по предметной области, так как можно ограничиться учетом только существенных особенностей предметной области;
- возможно применение эвристик, повышающих эффективность процесса поиска решения.

К недостаткам рассуждений на основе прецедентов можно отнести следующее:

- при описании прецедентов обычно ограничиваются поверхностными знаниями о предметной области;
- большое количество прецедентов может привести к снижению производительности системы;
- проблематичным является определение критериев для индексации и сравнения прецедентов;
- сложности с отладкой алгоритмов определения подобных (аналогичных) прецедентов;
- невозможность получения решения задач, для которых нет прецедентов или степень их подобия меньше заданного порогового значения.

1.2 Методы представления прецедентов

В общем случае модель представления прецедента включает описание ситуации и решение для данной ситуации:

$$CASE = (Situation, Solution), \quad (1)$$

где *Situation* – ситуация, описывающая данный прецедент, а *Solution* – решение (диагноз) и рекомендации ЛПР. Различия способов представления прецедентов заключаются в разных способах описания этих компонент [4].

Кроме того, прецедент может содержать результат применения решения. Описание результата применения решения может включать список выполненных действий, дополнительные комментарии и ссылки на другие прецеденты. Прецедент может иметь как положительный, так и отрицательный исход применения решения, а также в некоторых случаях может приводиться обоснование выбора данного решения и возможные альтернативы.

Прецеденты могут быть представлены в виде записей в БД, концептуальных графов, семантической сети, древовидных структур, предикатов, фреймов, рисунков и мультимедийной информации.

1.3 Методы извлечения прецедентов

Существует целый ряд методов извлечения прецедентов и их модификаций.

1.3.1 Метод ближайшего соседа (NN – Nearest Neighbor)

Это самый распространенный метод сравнения и извлечения прецедентов [5]. Он позволяет довольно легко вычислить степень сходства текущей проблемной ситуации и прецедентов из БП системы. С целью определения степени сходства на множестве параметров, используемых для описания прецедентов и текущей ситуации, вводится определенная метрика. Далее в соответствии с выбранной метрикой определяется расстояние от целевой точки, соответствующей текущей проблемной ситуации, до точек, представляющих прецеденты из БП, и выбирается ближайшая к целевой точка.

Основными преимуществами данного метода являются простота реализации и универсальность в смысле независимости от специфики конкретной проблемной области. К существенным недостаткам метода можно отнести сложность выбора метрики для определения степени сходства и прямую зависимость требуемых вычислительных ресурсов от размера БП, а также неэффективность при работе с неполными и зашумленными исходными данными.

1.3.2 Метод извлечения прецедентов на основе деревьев решений

Этот метод предполагает нахождение требуемых прецедентов путем разрешения вершин дерева решений. Каждая вершина дерева указывает, по какой ее ветви следует осуществлять дальнейший поиск решения. Выбор ветви осуществляется на основе информации о текущей проблемной ситуации. Таким образом, необходимо добраться до концевой вершины, которая соответствует одному или нескольким прецедентам. Если концевая вершина связана с некоторым подмножеством прецедентов, то тогда для

выбора наиболее подходящего из них может использоваться метод ближайшего соседа. Такой подход рекомендуется применять для больших БП, т.к. основная часть работы по извлечению прецедентов выполняется заранее на этапе построения дерева решений, что значительно сокращает время поиска решения.

К недостаткам этого методы относится сложность решения задачи классификации или кластеризации на начальном этапе и при добавлении в базу новых прецедентов.

1.3.3 Метод извлечения прецедентов на основе знаний

В отличие от методов, описанных выше, данный метод позволяет учесть знания экспертов (ЛПР) по конкретной предметной области (коэффициенты важности параметров, выявленные зависимости и т.д.) при извлечении. Метод может успешно применяться совместно с другими методами извлечения прецедентов, особенно когда БП имеет большие размеры и предметная область является открытой и динамической.

1.3.4 Метод извлечения с учетом применимости прецедентов

В большинстве систем, использующих механизмы рассуждений на основе прецедентов, предполагается, что наиболее схожие с текущей проблемной ситуацией прецеденты являются наиболее применимыми в этой ситуации. Однако это не всегда так. В основе понятия извлечения на основе применимости (адаптируемости) лежит то, что извлечение прецедентов базируется не только на их сходстве с текущей проблемной ситуацией, но и на том, насколько хорошую для желаемого результата модель они собой представляют. Т.е. на выбор извлекаемых прецедентов влияет возможность их применения в конкретной ситуации. В некоторых системах эта проблема решается путем сохранения прецедентов вместе с комментариями по их применению. Использование указанного подхода позволяет сделать поиск решения более эффективным, заранее отбрасывая часть заведомо неперспективных прецедентов.

2 ПРЕДСТАВЛЕНИЕ ПРЕЦЕДЕНТОВ С ИСПОЛЬЗОВАНИЕМ ОНТОЛОГИЧЕСКОГО ПОДХОДА

2.1 Понятие онтологии

Понятие *онтология* происходит от др.-греч. “онтос” - сущее, “логос” - учение, понятие, т.е это раздел философии, изучающий бытие.

Среди специалистов, занимающихся проблемами компьютерной лингвистики, наиболее устоявшимся (классическим) считается определение

онтологии, данное Губертом: “*онтология* – это спецификация концептуализации” [6].

Концептуализация — это структура реальности, рассматриваемая независимо от словаря предметной области и конкретной ситуации.

2.2 Модель онтологии

Определение онтологии как формального представления предметной области, построенного на базе концептуализации, предполагает выделение ее трех взаимосвязанных компонентов: таксономии терминов, описаний смысла терминов, а также правил их использования и обработки [6,7]. Таким образом, *модель онтологии O* задает тройка:

$$O = (X, R, Ф), \quad (2)$$

где X — конечное множество концептов (понятий, терминов) предметной области, которую представляет онтология; R — конечное множество отношений между концептами; $Ф$ — конечное множество функций интерпретации, заданных на концептах и (или) отношениях.

2.3 Отнологический подход к представлению знаний

Выбор онтологии для представления прецедентов обусловлен рядом важных достоинств, отличающих их от других моделей представления знаний [8, 9]. Использование онтологического подхода позволяет задать сложную структуру прецедента, включающую данные разных типов, естественность представления структурированных знаний и достаточно простое их обновление в относительно однородной среде. Последнее свойство особенно важно для ИС, ориентированных на открытые и динамические предметные области.

Онтология является базой знаний, содержащей знания предметной области, используемые для поддержки цикла рассуждений по прецедентам, в частности, на этапе поиска и извлечения прецедентов. Также онтология задает структуру прецедента и является хранилищем прецедентов.

В качестве языка описания онтологии выбран язык OWL.

Знания о предметной области и модель прецедентов описываются в виде иерархии концептов онтологии, а каждый прецедент из БП в виде иерархии экземпляров концептов, связанных [10].

OWL имеет ряд преимуществ:

- Широкие возможности для описания прецедентов обеспечиваются выразительными средствами языка OWL.
- Легкость обновления и расширения базы знаний и БП.
- Компактность размещения данных.
- Нет необходимости в использовании средств хранения и описания прецедентов, кроме онтологии.
- Позволяет организовать эффективный поиск по БП.

Фрагмент онтологии на языке OWL:

```
<Declaration>
<Class IRI="#Blood_pressure"/>
</Declaration>
<Declaration>
<Class IRI="#Blood_pressure_stbl"/>
</Declaration>
…
<SubClassOf>
<Class IRI="#Blood_pressure_stbl"/>
<Class IRI="#Stability"/>
</SubClassOf>
…
<ObjectPropertyDomain>
<ObjectProperty IRI="#IsA"/>
<Class IRI="#Case"/>
</ObjectPropertyDomain>
```

3 АЛГОРИТМ ИЗВЛЕЧЕНИЯ ПРЕЦЕДЕНТОВ НА ОСНОВЕ ОНТОЛОГИИ ПРЕДМЕТНОЙ ОБЛАСТИ

Предлагается осуществлять извлечение и определение сходства прецедента *C* и текущей ситуации *Q* в два этапа:

- Определение сходства прецедента с текущей ситуацией на основе онтологии предметной области и формирование парных соответствий.
- Определение сходства прецедента и текущей ситуации по методу ближайшего соседа с учетом полученных парных соответствий.

Рассмотрим первый этап подробнее.

3.1 Определение сходства на основе онтологии предметной области

На первом этапе описание ситуации прецедента и текущая ситуация сравниваются по структуре. Цель данного этапа – определить возможные парные соответствия между прецедентом и текущей ситуацией и оценить их сходство по структуре.

3.1.1 Парное соответствие

Если рассматривать объекты C и Q как множества, элементами которых являются $c \in C$ и $q \in Q$, тогда парным соответствием φ между объектами (множествами) C и Q будем называть подмножество их прямого произведения $\varphi \subset C \times Q$. Таким образом, соответствие φ включает множество пар элементов множеств C и Q ($<c,q> \in \varphi$), отображающих наличие между ними какой-либо связи.

Существуют различные способы задания соответствия: графический, табличный, перечислением. При использовании последнего соответствие задается в виде упорядоченных пар – «двоек», содержащих элементы обоих множеств C и Q, между которыми есть соответствие ($\varphi=\{<c_1,q_1>, \dots, <c_n,q_n>\}$).

3.1.2 Алгоритм поиска парных соответствий

Будем считать, что имеется онтология предметной области. Онтология содержит описание предметной области, описание модели прецедента и БП. Таким образом, получить множество парных соответствий можно, используя следующий алгоритм:

На вход алгоритма подается онтология предметной области O, которая так же является БП. Имя ситуации прецедента C и имя текущей ситуации Q.

На выходе алгоритм возвращает множество парных соответствий F.

Шаг 1. F=∅.
Шаг 2. Вызов процедуры F = Pairs(Q,C,O).
Шаг 3. Если F=∅, то к шагу 4, иначе добавить ко всем парным соответствиям из F пару <Q,C> (F = F × <Q,C>) и к шагу 4.
Шаг 4. Выход.

Pairs.
Входные переменные:
Q – Имя текущей ситуации.
C – Имя прецедента.
O – Онтология предметной области.
Выходные переменные:
F – Полученное множество парных соответствий.
Промежуточные переменные:
i – параметр цикла.
Rel – текущее отношение.
Fres – Переменная возвращающая множество парных соответствий.
Ftemp – временная переменная, возвращающая множество парных соответствий.
Frec – переменная, возвращающая множество парных соответствий в результате рекурсии.

Шаг 1. i=0, Fres=∅, Ftemp=∅.
Шаг 2. Если нет непроверенных концептов, связанных с Q, то к Fres добавить Ftemp (объединение), перейти к шагу 6. Иначе i=i+1 и выбираем непроверенный концепт Q_i связанный с Q отношением Rel, переходим к шагу 3.
Шаг 3. Поиск концепта C_j связанного с C отношением Rel и удовлетворяющего условию – имя концепта C_j совпадает с Q_i. Если такой C_j найден, то формируем соответствие $F_j = <Q_i,C_j>$, выполняем вызов Frec = Pairs(Q_i,C_j,O), производим добавление полученных парных соответствий во временное множество Ftemp (Ftemp= Frec × F_j, где × - декартово произведение множеств) и переходим к шагу 2, иначе переходим к шагу 4.
Шаг 4. Поиск концептов C_j связанных с C отношением Rel и удовлетворяющих условию – концепт C_j подобен Q_i. Переходим к шагу 5.

> *Примечание:* Подобие между концептами онтологии можно определять руководствуясь следующим принципом. Концепты A и B подобны, если родительские концепты A подобны или совпадают с родительскими концептами B.

Шаг 5. Пока есть C_j, удовлетворяющие условиям шага 4, добавляем к Fres временное множество Ftemp (объединение). Формируем соответствие $F_j=<Q_i,C_j>$ и выполняем вызов Frec = Pairs(Q_i,C_j,O), запоминаем парные соответствия Ftemp = Frec × F_j . Помечаем Q_i как проверенный и выполняем вызов Frec = Pairs(Q,C,O), производим добавление полученных парных

соответствий во временное множество Ftemp = Frec × Ftemp, а Q_i – помечаем как непроверенный концепт и переходим к шагу 5. Если больше нет C_j, удовлетворяющих условиям шага 4 переходим к шагу 2.

Шаг 6. Выход.

Пример работы алгоритма

Рассмотрим работу данного алгоритма на примере. Случай когда на онтологии задано отношение is-a.

Имеется онтология предметной области (Рис 2)

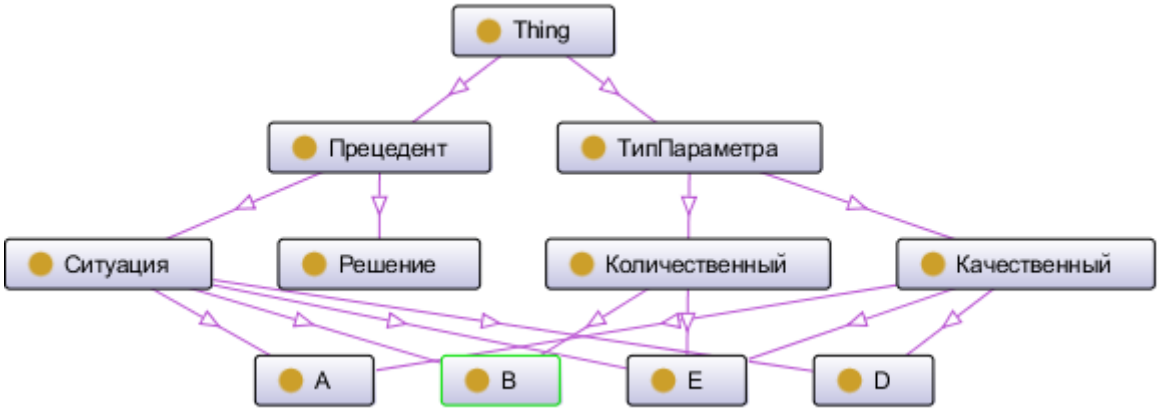
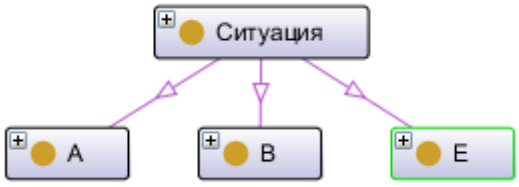
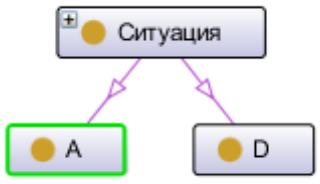


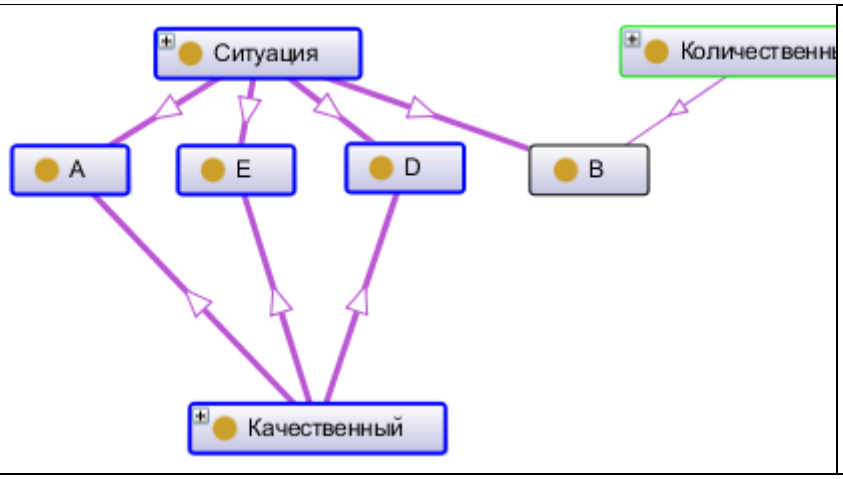
Рис. 2. Онтология предметной области

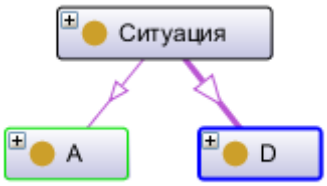
Рассмотрим нахождение парного соответствия между прецедентом, содержащимся в БП и текущей проблемной ситуацией (таблица 1).

Таблице 1.

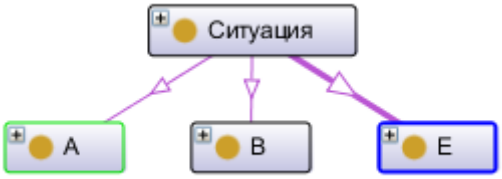
Пример работы алгоритма извлечения прецедентов

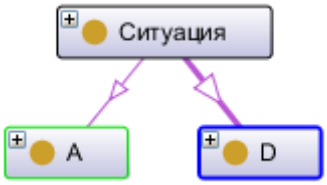
На входе алгоритма имеем:	
Описание ситуации прецедента	Описание текущей ситуации
	
Выбираем Q_i =D, для D нет точного совпадения по имени. Но есть подобные концепты A и E. B не является подобным B так как имеет другой тип параметра	



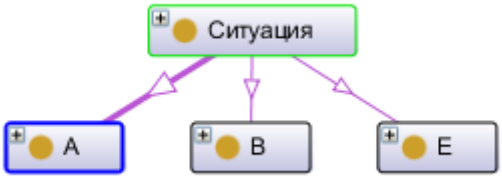


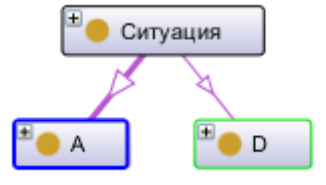
Сформируем пару <D, E>. У D нет потомков.





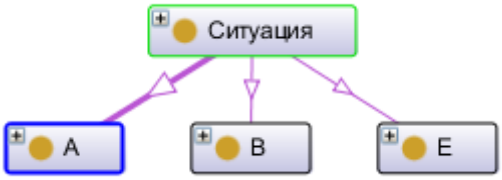
Продолжим поиск соответствия для A. Находим точное совпадение по названию. Формируем пару <A, A>.

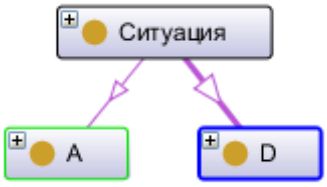




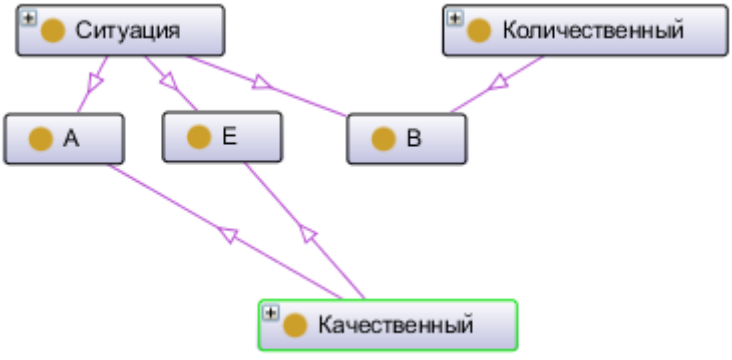
У A нет потомков, поэтому
Получаем парное соответствие {<A, A>,<D, E>}. Возвращаемся назад.

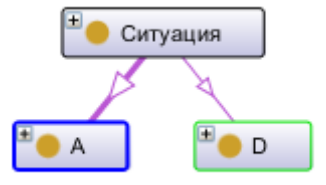
Сформируем пару <D, A>. У D нет потомков.



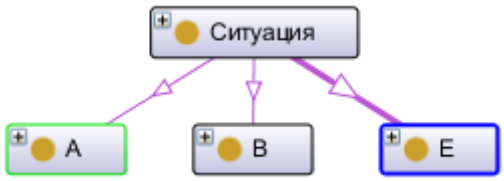


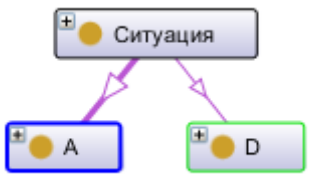
Продолжим поиск соответствия для A. Нет неиспользованных концептов с точным совпадением по названию. Но есть подобный концепт E.





Сформируем пару <A, E>





У А нет потомков, поэтому получаем парное соответствие {<A, E>,<D, A>}. Возвращаемся назад. Вариантов для парного соответствия у D больше нет. Приписываем в каждое из полученных множеств пару <Q, C>.
В итоге имеем два парных соответствия
{<A, A>, <D, E>, <Q, C>}, {<A, E>, <D, A>, <Q,C>}

3.1.3 Оценка сходства на основе онтологии предметной области

Для каждого парного соответствия можно определить оценку близости ситуации прецедента и текущей ситуации по структуре.

Оценку сходства на основе онтологии предметной области будем вычислять следующим образом:

$$Sstruct_i(C,Q) = \frac{|F_i|}{\max(|C|,|Q|)}, \quad (3)$$

Где F_i- i-е парное соответствие, C – множество концептов ситуации прецедента, Q –множество концептов текущей ситуации.

3.2 Определение соответствия по методу ближайшего соседа

Как уже отмечалось ранее это самый популярный и часто используемый метод для извлечения прецедентов, который также широко применяется для решения задач классификации, регрессии, распознавания образов и т.д. В эго основе лежит определенный способ измерения степени схожести (близости) прецедента и текущей проблемной ситуации [11].

В данном случае на вход метода ближайшего соседа подаются парные соответствия, определенные на предыдущем этапе. Для каждого парного соответствия определяются расстояние d_{CQ} между текущей ситуацией и прецедентом. Для определения значения степени сходства Snn(C,Q) необходимо найти максимальное расстояние d_{MAX} в выбранной метрике, используя границы диапазонов параметров ($x_i^{нач}$ и $x_i^{кон}$, i=1,…,n) (*Примечание:* Границы диапазонов параметров жестко задаются в описании концептов онтологии предметной области). Затем можно вычислить значение степени сходства:

Snn(C,Q)=1–d_{CQ}/d_{MAX} или в процентах *Snn(C,Q)=(1–d_{CQ}/d_{MAX})*100%.*

4 РЕАЛИЗАЦИЯ ПРОТОТИПА CBR-СИСТЕМЫ

4.1 Архитектура системы

На Рис 3 представлена архитектура разработанного прототипа CBR-системы.

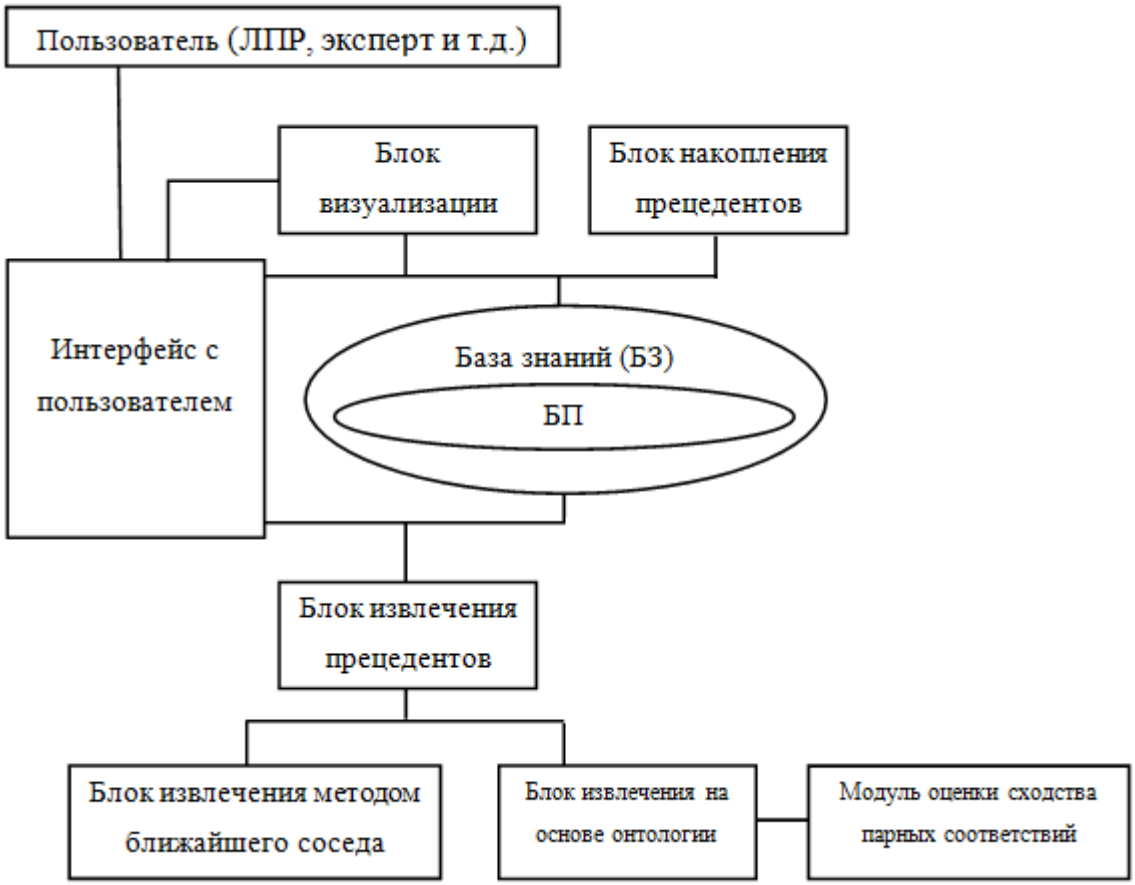


Рис. 3. Архитектура прототипа CBR-системы

4.2 Средства реализации

Разработка программных модулей была осуществлена на языке C# [12] под операционную систему Windows на платформе MS .NET Framework 4. Для хранения БЗ и БП использовался OWL-документ. Для визуализации используется библиотека для прорисовки графов GLEE (ограниченная бесплатная версия Microsoft Automatic Graph Layout (MSAGL) [13].

Архитектура системы спроектирована таким образом, чтобы обеспечить ее расширяемость (добавление новых программных модулей).

В качестве редактора онтологий используется Protégé.

4.3 Демонстрационный пример

Работа реализованного прототипа CBR-системы рассмотрена на примере решения задачи экспертного диагностирования с использованием набор данных из хранилища UCI Machine Learning Repository [14]

(Калифорнийский университет, США). UCI Machine Learning Repository содержит реальные и модельные задачи машинного обучения в области биологии, медицины, физики, техники, социологии и др. Задачи (наборы данных, data set) именно этого репозитория чаще всего используются научным сообществом для эмпирического анализа алгоритмов.

В демонстрационном примере решается задача экспертной диагностики постоперационных случаев.

Рассмотрен реальный набор медицинских данных из хранилища UCI Machine Learning Repository.

Ситуация представлена 8 параметрами, которые описывают состояние пациента после операции. На основе этих данных необходимо принять одно из трех решений:

- Отправить пациента в реанимацию.
- Продолжить наблюдение в общей палате.
- Выписать пациента.

В данном случае поставленную задачу экспертного диагностирования можно свести к задаче классификации. То есть необходимо отнести текущую ситуацию к одному из трех классов решений.

Онтология предметной области была спроектирована в редакторе онтологий Protégé. Онтология содержит 18 концептов и отношение «is-a». На рис. 4 представлена иерархия концептов онтологии. В данной онтологии представлено 16 прецедентов.

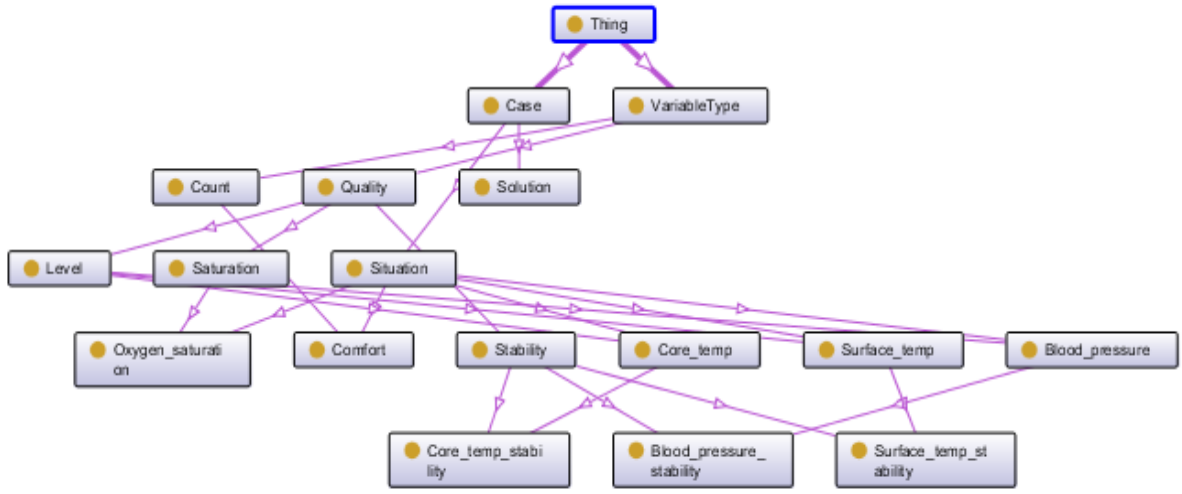


Рис. 4. Иерархия концептов онтологии

Пользовательский интерфейс состоит из нескольких вкладок.

- Вкладка «Онтология» (Рис. 5) – вид онтологии предметной области.

а) Графическое преставление онтологии.
б) Дерево концептов онтологии.

в) Блок кнопок управления отображением: Общий вид – отображение всех концептов и связей онтологии, Предки – всех предков выбранного класса, Потомки – всех потомков выбранного класса.

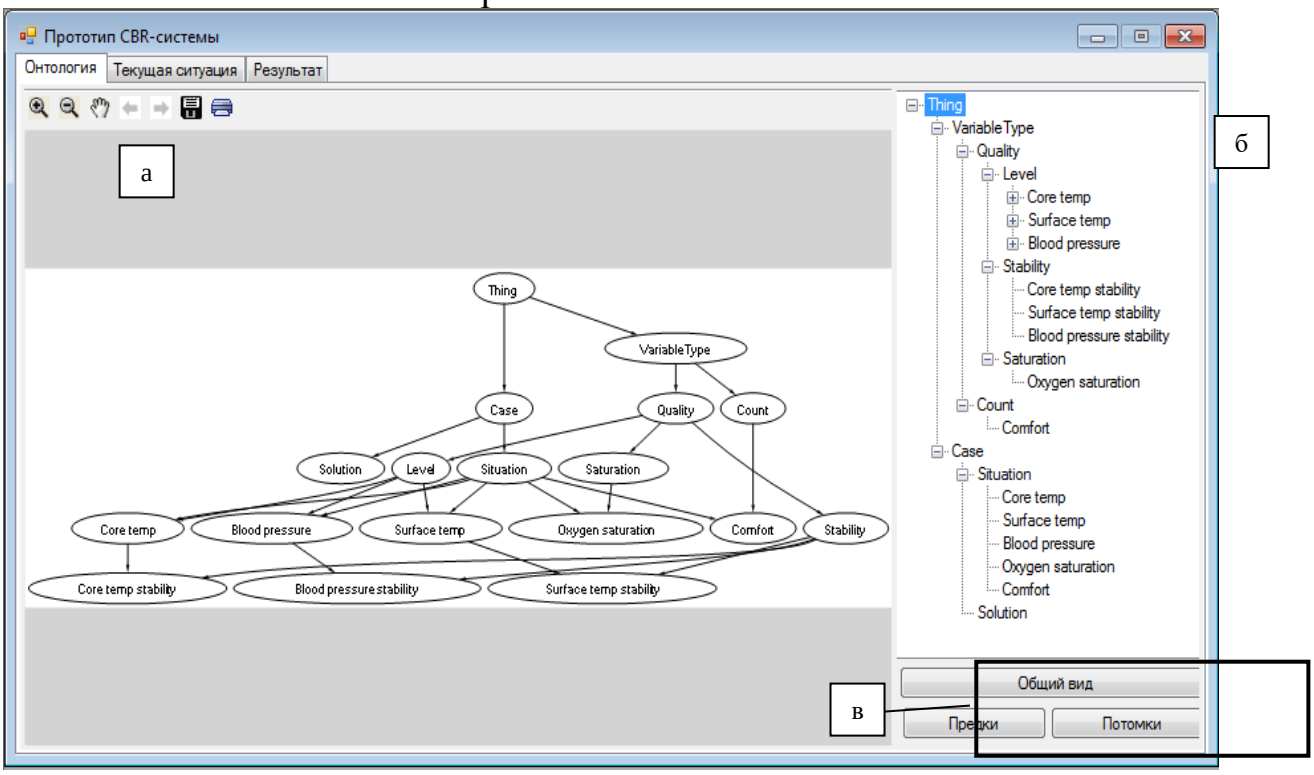


Рис. 5. Вкладка «Онтология»

- Вкладка «Текущая ситуация» (Рис. 6) – обеспечивает редактирование текущей ситуации и содержит все элементы вкладки «Онтология», также дополнительные элементы:

а) Блок кнопок для редактирования текущей ситуации.
б) Поля для задания пороговых значений при извлечении прецедентов.
в) Поле-таблица для редактирования значений параметров прецедента.

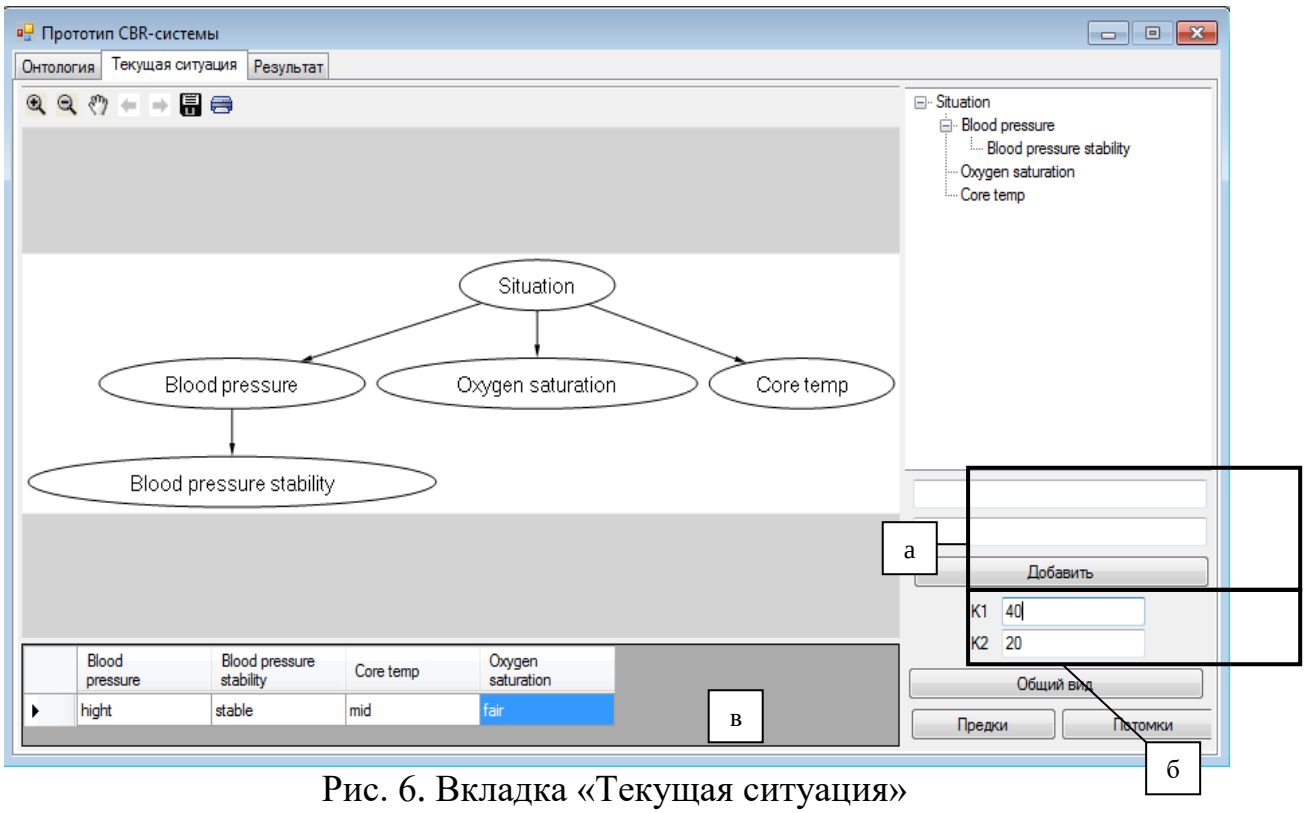


Рис. 6. Вкладка «Текущая ситуация»

- Вкладка «Результат» (Рис 7) – содержит сводную таблицу по прецедентам.
  - Прецедент – название прецедента.
  - Оценка по онтологии – оценка сходства на основе онтологии предметной области.
  - Оценка МБС – оценка сходства методом ближайшего соседа.
  - Кнопка выбора прецедента.
  - Решение – решение прецедента, рекомендация.

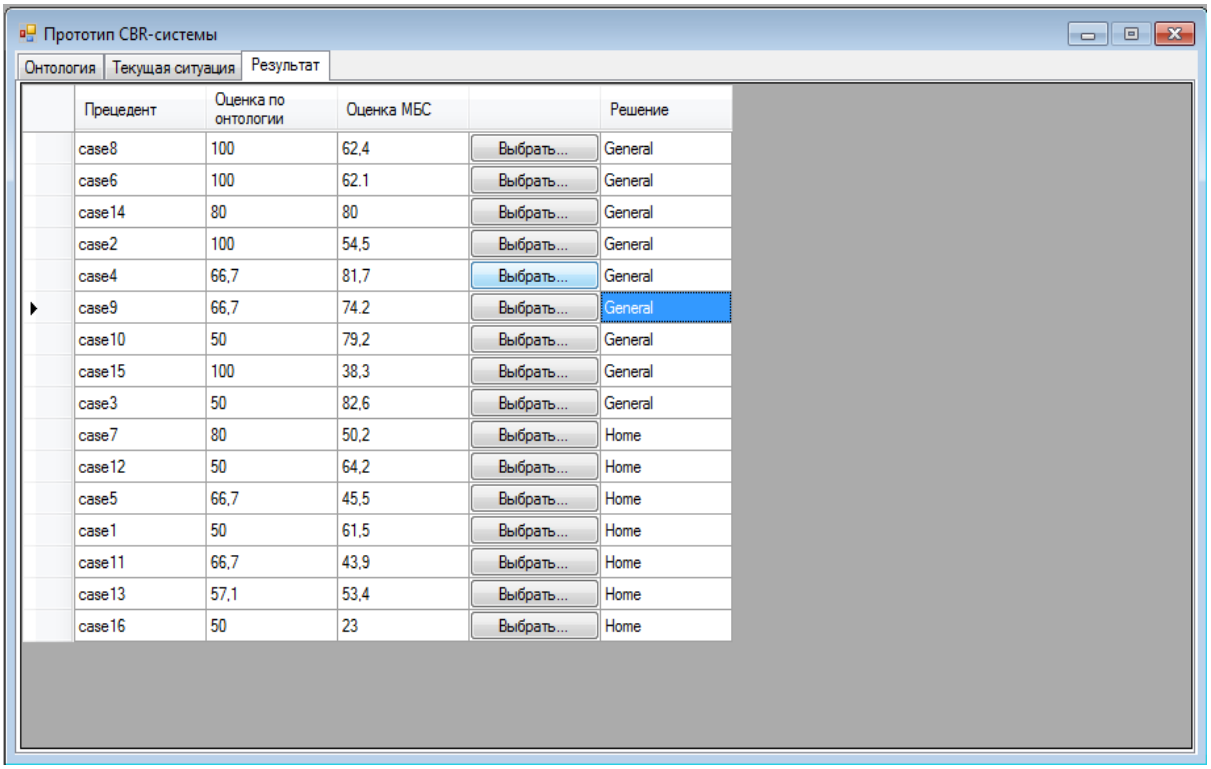


Рис. 7. Вкладка «Результат»

В данном случае можно сделать вывод о том, что в текущей ситуации следует перевести пациента в общую палату, что соответствует корректной классификации.

ЗАКЛЮЧЕНИЕ

Основные результаты проделанной работы:

1. Рассмотрены общие понятия и особенности CBR-технологии, а также процесс поиска решения на основе прецедентов, представляющий собой CBR-цикл. Описаны методы извлечения прецедентов.
2. Для реализации представления знаний (прецедентов) в CBR-системе выбрана онтологическая модель.
3. Предложен обобщенный алгоритм извлечения прецедентов, использующий на первом этапе алгоритм извлечения прецедентов с учетом онтологии предметной области, а на втором алгоритм извлечения прецедентов методом ближайшего соседа.
4. Разработана архитектура и выполнена программная реализация прототипа CBR-системы на языке C# под операционную систему Windows XP/Vista/7 на платформе MS .NET Framework 4.
5. Рассмотрен пример использования разработанной системы для решения задачи экспертной диагностики на основе реальных данных из хранилища UCI Machine Learning Repository.

СПИСОК ЛИТЕРАТУРЫ

1. **Вагин В.Н., Головина Е.Ю., Загорянская А.А., Фомина М.В.** Достоверный и правдоподобный вывод в интеллектуальных системах // Под ред. В.Н. Вагина, Д.А. Поспелова. –М.: ФИЗМАТЛИТ, 2004. – 704с.
2. **Поспелов Д.А.** Моделирование рассуждений. Опыт анализа мыслительных актов. –М.: Радио и связь. 1989.
3. **Aamodt A., Plaza E.** Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches// Artificial Intelligence Communications. IOS Press. – 1994. – Vol.7, №1. – P.39-59.
4. **Sankar Pal, Simon Shiu**. Foundations of soft case-based reasoning. Wiley-Interscience. – 2004. – 274 p.
5. **Люгер Д.Ф.** Искусственный интеллект: стратегии и методы решения сложных проблем. Пер. с англ. – 4-е изд. –М.: Издательский дом “Вильямс”, 2003. – 864 с.
6. **Башмаков А.И., Башмаков И.А.** Интеллектуальные информационные технологии: Учеб. пособие. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2005. – 304 с.
7. **Гаврилова Т.А., Хорошевский В.Ф.** Базы знаний интеллектуальных систем. –СПб.: Питер, 2000. – 384 с.
8. **Дворянкин А.М., Сипливая М.Б., Жукова И.Г., Капыш А.С., Кульцов А.Е.** Разработка модели представления прецедента на основе онтологии и создание базы знаний для интеллектуальной системы поддержки инженерного анализа в области контактной механики / // Изв. ВолгГТУ. Серия «Актуальные проблемы управления, вычисли-тельной техники и информатики в технических системах»: межвуз. сб. науч. ст. / Вол-гГТУ. - Волгоград, 2008. - Вып. 4, № 2. - С. 94-98.
9. **Алехин Р.В.** Использование языка OWL для формирования онтологий предметной области // Радиоэлектроника, электротехника и энергетика: XVII Междунар. науч.-техн. конф. студентов и аспирантов: Тезисы докладов в 3 т. Т. 1. М.: Издательский дом МЭИ, 2011. – с. 353–354.
10. OWL, язык веб-онтологий. Руководство (http://www.w3.org/TR/owl-guide/)
11. **Варшавский П.Р., Еремеев А.П.** Методы правдоподобных рассуждений на основе аналогий и прецедентов для интеллектуальных систем поддержки принятия решений // Новости искусственного интеллекта, № 3, 2006. –С. 39-62.
12. **Шилдт Г.** Полный справочник по C#.. –М.: Издательский дом “Вильямс”, 2008. – 752 с.
13. MSAGL (http://research.microsoft.com/en-us/projects/msagl/).
14. Хранилище UCI Machine Learning Repository (http://archive.ics.uci.edu/ml/).

Приложение 1 Пример OWL-документа

<Ontology xmlns="http://www.w3.org/2002/07/owl#" xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" xmlns:xsd="http://www.w3.org/2001/XMLSchema#"xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xml:base="http://www.semanticweb.org/ontologies/2011/5/medical.owl"ontologyIRI="http://www.semanticweb.org/ontologies/2011/5/medical.owl">
<Prefix name="rdf" IRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
<Prefix name="rdfs" IRI="http://www.w3.org/2000/01/rdf-schema#"/>
<Prefix name="xsd" IRI="http://www.w3.org/2001/XMLSchema#"/>
<Prefix name="owl" IRI="http://www.w3.org/2002/07/owl#"/>
<Declaration>
<Class IRI="#Blood_pressure"/>
</Declaration>
<Declaration>
<Class IRI="#Blood_pressure_stability"/>
</Declaration>
<Declaration>
<Class IRI="#Case"/>
</Declaration>
<Declaration>
<Class IRI="#Comfort"/>
</Declaration>
<Declaration>
<Class IRI="#Core_temp"/>
</Declaration>
<Declaration>
<Class IRI="#Core_temp_stability"/>
</Declaration>
<Declaration>
<Class IRI="#Count"/>
</Declaration>
<Declaration>
<Class IRI="#Level"/>
</Declaration>
<Declaration>
<Class IRI="#Oxygen_saturation"/>
</Declaration>
<Declaration>
<Class IRI="#Quality"/>
</Declaration>
<Declaration>
<Class IRI="#Saturation"/>
</Declaration>
<Declaration>
<Class IRI="#Situation"/>
</Declaration>
<Declaration>
<Class IRI="#Solution"/>
</Declaration>
<Declaration>
<Class IRI="#Stability"/>
</Declaration>
<Declaration>
<Class IRI="#Surface_temp"/>
</Declaration>
<Declaration>
<Class IRI="#Surface_temp_stability"/>
</Declaration>
<Declaration>
<Class IRI="#VariableType"/>
</Declaration>

<SubClassOf>
<Class IRI="#Blood_pressure"/>
<Class IRI="#Level"/>
</SubClassOf>
<SubClassOf>
<Class IRI="#Blood_pressure"/>
<Class IRI="#Situation"/>
</SubClassOf>
<SubClassOf>
<Class IRI="#Blood_pressure_stability"/>
<Class IRI="#Blood_pressure"/>
</SubClassOf>
<SubClassOf>
<Class IRI="#Blood_pressure_stability"/>
<Class IRI="#Stability"/>
</SubClassOf>
<SubClassOf>
<Class IRI="#Case"/>
<Class abbreviatedIRI="owl:Thing"/>
</SubClassOf>
<SubClassOf>
<Class IRI="#Comfort"/>
<Class IRI="#Count"/>
</SubClassOf>
<SubClassOf>
<Class IRI="#Comfort"/>
<Class IRI="#Situation"/>
</SubClassOf>
<SubClassOf>
<Class IRI="#Core_temp"/>
<Class IRI="#Level"/>
</SubClassOf>
<SubClassOf>
<Class IRI="#Core_temp"/>
<Class IRI="#Situation"/>
</SubClassOf>
<SubClassOf>
<Class IRI="#Core_temp_stability"/>
<Class IRI="#Core_temp"/>
</SubClassOf>
<SubClassOf>
<Class IRI="#Core_temp_stability"/>
<Class IRI="#Stability"/>
</SubClassOf>
<SubClassOf>
<Class IRI="#Count"/>
<Class IRI="#VariableType"/>
</SubClassOf>
<SubClassOf>
<Class IRI="#Level"/>
<Class IRI="#Quality"/>
</SubClassOf>
<SubClassOf>
<Class IRI="#Oxygen_saturation"/>
<Class IRI="#Saturation"/>
</SubClassOf>
<SubClassOf>
<Class IRI="#Oxygen_saturation"/>
<Class IRI="#Situation"/>
</SubClassOf>
<SubClassOf>
<Class IRI="#Quality"/>
<Class IRI="#VariableType"/>

```
</SubClassOf>
<SubClassOf>
<Class IRI="#Saturation"/>
<Class IRI="#Quality"/>
</SubClassOf>
<SubClassOf>
<Class IRI="#Situation"/>
<Class IRI="#Case"/>
</SubClassOf>
<SubClassOf>
<Class IRI="#Solution"/>
<Class IRI="#Case"/>
</SubClassOf>
<SubClassOf>
<Class IRI="#Stability"/>
<Class IRI="#Quality"/>
</SubClassOf>
<SubClassOf>
<Class IRI="#Surface_temp"/>
<Class IRI="#Level"/>
</SubClassOf>
<SubClassOf>
<Class IRI="#Surface_temp"/>
<Class IRI="#Situation"/>
</SubClassOf>
<SubClassOf>
<Class IRI="#Surface_temp_stability"/>
<Class IRI="#Stability"/>
</SubClassOf>
<SubClassOf>
<Class IRI="#Surface_temp_stability"/>
<Class IRI="#Surface_temp"/>
</SubClassOf>
<SubClassOf>
<Class IRI="#VariableType"/>
<Class abbreviatedIRI="owl:Thing"/>
</SubClassOf>
<DisjointClasses>
<Class IRI="#Blood_pressure"/>
<Class IRI="#Comfort"/>
<Class IRI="#Core_temp"/>
<Class IRI="#Oxygen_saturation"/>
<Class IRI="#Surface_temp"/>
</DisjointClasses>
<DisjointClasses>
<Class IRI="#Blood_pressure"/>
<Class IRI="#Core_temp"/>
<Class IRI="#Surface_temp"/>
</DisjointClasses>
<DisjointClasses>
<Class IRI="#Blood_pressure_stability"/>
<Class IRI="#Core_temp_stability"/>
<Class IRI="#Surface_temp_stability"/>
</DisjointClasses>
<DisjointClasses>
<Class IRI="#Case"/>
<Class IRI="#VariableType"/>
</DisjointClasses>
<DisjointClasses>
<Class IRI="#Count"/>
<Class IRI="#Quality"/>
</DisjointClasses>
<DisjointClasses>
```

<Class IRI="#Level"/>
<Class IRI="#Saturation"/>
<Class IRI="#Stability"/>
</DisjointClasses>
<DisjointClasses>
<Class IRI="#Situation"/>
<Class IRI="#Solution"/>
</DisjointClasses>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label"/>
<IRI>#Blood_pressure</IRI>
<Literal xml:lang="ru" datatypeIRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#PlainLiteral">Blood pressure</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label"/>
<IRI>#Blood_pressure_stability</IRI>
<Literal xml:lang="ru" datatypeIRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#PlainLiteral">Blood pressure stability</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label"/>
<IRI>#Case</IRI>
<Literal xml:lang="ru" datatypeIRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#PlainLiteral">Case</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label"/>
<IRI>#Comfort</IRI>
<Literal xml:lang="ru" datatypeIRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#PlainLiteral">Comfort</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label"/>
<IRI>#Core_temp</IRI>
<Literal xml:lang="ru" datatypeIRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#PlainLiteral">Core temp</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label"/>
<IRI>#Core_temp_stability</IRI>
<Literal xml:lang="ru" datatypeIRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#PlainLiteral">Core temp stability</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label"/>
<IRI>#Count</IRI>
<Literal xml:lang="ru" datatypeIRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#PlainLiteral">Count</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label"/>
<IRI>#Level</IRI>
<Literal xml:lang="ru" datatypeIRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#PlainLiteral">Level</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label"/>
<IRI>#Oxygen_saturation</IRI>
<Literal xml:lang="ru" datatypeIRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#PlainLiteral">Oxygen saturation</Literal>
</AnnotationAssertion>
<AnnotationAssertion>

<AnnotationProperty abbreviatedIRI="rdfs:label"/>
<IRI>#Quality</IRI>
<Literal xml:lang="ru" datatypeIRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#PlainLiteral">Quality</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label"/>
<IRI>#Saturation</IRI>
<Literal xml:lang="ru" datatypeIRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#PlainLiteral">Saturation</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label"/>
<IRI>#Situation</IRI>
<Literal xml:lang="ru" datatypeIRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#PlainLiteral">Situation</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label"/>
<IRI>#Solution</IRI>
<Literal xml:lang="ru" datatypeIRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#PlainLiteral">Solution</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label"/>
<IRI>#Stability</IRI>
<Literal xml:lang="ru" datatypeIRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#PlainLiteral">Stability</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label"/>
<IRI>#Surface_temp</IRI>
<Literal xml:lang="ru" datatypeIRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#PlainLiteral">Surface temp</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label"/>
<IRI>#Surface_temp_stability</IRI>
<Literal xml:lang="ru" datatypeIRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#PlainLiteral">Surface temp stability</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label"/>
<IRI>#VariableType</IRI>
<Literal xml:lang="ru" datatypeIRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#PlainLiteral">VariableType</Literal>
</AnnotationAssertion>
</Ontology>

Приложение 2 Набор медицинских данных

Описание набора данных
1. Core temperature - внутренняя температура
high (> 37), mid (>= 36 and <= 37), low (< 36)
2. Surface temperature - температура поверхности
high (> 36.5), mid (>= 36.5 and <= 35), low (< 35)
3. Oxygen saturation - насыщение кислородом
excellent (>= 98), good (>= 90 and < 98), fair (>= 80 and < 90), poor (< 80)
4. Blood measure - кровяное давление
high (> 130/90), mid (<= 130/90 and >= 90/70), low (< 90/70)
5. Surface temperature stability - стабильность температуры поверхности
stable, mod-stable, unstable
6. Core temperature stability - стабильность внутренне температуры
stable, mod-stable, unstable
7 Blood measure stability – стабильность кровяного давления
stable, mod-stable, unstable
8. COMFORT – субъективная оценка состояния, пациент сам оценивает свое состояние по 20 бальной системе
9. Decision – решение (диагноз)
I – реанимация
S – выписка
A – общая палата

mid,low,excellent,mid,stable,stable,stable,15,A
mid,high,excellent,high,stable,stable,stable,10,S
high,low,excellent,high,stable,stable,mod-stable,10,A
mid,low,good,high,stable,unstable,mod-stable,15,A
mid,mid,excellent,high,stable,stable,stable,10,A
high,low,good,mid,stable,stable,unstable,15,S
mid,low,excellent,high,stable,stable,mod-stable,05,S
high,mid,excellent,mid,unstable,unstable,stable,10,S
mid,high,good,mid,stable,stable,stable,10,S
mid,low,excellent,mid,unstable,stable,mod-stable,10,S
mid,mid,good,mid,stable,stable,stable,15,A
mid,low,good,high,stable,stable,mod-stable,10,A
high,high,excellent,high,unstable,stable,unstable,15,A
mid,high,good,mid,unstable,stable,mod-stable,10,A
mid,low,good,high,unstable,unstable,stable,15,S
high,high,excellent,high,unstable,stable,unstable,10,A
low,high,good,high,unstable,stable,mod-stable,15,A
mid,low,good,high,unstable,stable,stable,10,A
mid,high,good,mid,unstable,stable,unstable,15,A
mid,mid,good,mid,stable,stable,stable,10,A
low,high,good,mid,unstable,stable,stable,15,A
low,mid,excellent,high,unstable,stable,unstable,10,S
mid,mid,good,mid,unstable,stable,unstable,15,A
mid,mid,good,mid,unstable,stable,stable,10,A
high,high,good,mid,stable,stable,mod-stable,10,A
low,mid,good,mid,unstable,stable,stable,10,A
high,mid,good,low,stable,stable,mod-stable,10,A
low,mid,excellent,high,stable,stable,mod-stable,10,A
mid,mid,excellent,mid,stable,stable,unstable,15,A
mid,mid,good,mid,unstable,stable,unstable,10,S
mid,mid,good,high,unstable,stable,stable,10,A
low,low,good,mid,unstable,stable,unstable,10,A
mid,mid,excellent,high,unstable,stable,mod-stable,10,A
mid,low,good,mid,stable,stable,stable,10,A
low,mid,excellent,high,stable,stable,mod-stable,10,A
mid,mid,good,mid,stable,stable,stable,10,A
low,mid,excellent,mid,stable,stable,stable,10,S
low,low,good,mid,unstable,stable,unstable,10,S

low,low,good,mid,stable,stable,stable,07,S
mid,mid,good,high,unstable,stable,mod-stable,10,A
low,low,good,mid,unstable,stable,stable,10,A
low,mid,good,mid,stable,stable,stable,15,S
high,high,good,high,unstable,stable,stable,15,S
mid,mid,good,mid,stable,stable,stable,10,S
low,low,excellent,mid,stable,stable,stable,10,A
low,mid,good,mid,unstable,stable,stable,10,S
low,mid,good,high,unstable,stable,stable,?,I
mid,mid,excellent,mid,unstable,stable,stable,10,A
high,high,excellent,high,stable,stable,unstable,?,A
mid,high,good,low,unstable,stable,stable,10,A
mid,high,good,mid,unstable,mod-stable,mod-stable,10,A
low,high,excellent,mid,unstable,stable,stable,10,A
mid,low,excellent,high,unstable,stable,unstable,10,A
mid,mid,good,mid,unstable,stable,mod-stable,10,S
high,high,excellent,mid,unstable,stable,mod-stable,10,A
mid,mid,good,mid,unstable,stable,stable,15,A
high,mid,good,high,stable,stable,unstable,15,A
mid,low,good,high,unstable,stable,mod-stable,10,A
low,low,good,high,stable,stable,stable,10,A
mid,high,good,mid,stable,stable,mod-stable,10,A
mid,high,good,mid,unstable,stable,unstable,10,A
mid,low,excellent,high,stable,stable,stable,10,A
mid,mid,good,mid,stable,stable,unstable,10,A
mid,low,excellent,mid,stable,stable,unstable,10,S
high,mid,excellent,mid,unstable,unstable,unstable,10,A
mid,mid,good,high,stable,stable,stable,10,S
mid,low,excellent,mid,unstable,stable,stable,10,A
mid,mid,excellent,mid,unstable,stable,stable,10,A
mid,mid,excellent,high,stable,stable,stable,10,A
mid,mid,excellent,low,stable,stable,stable,10,A
mid,low,excellent,mid,unstable,unstable,unstable,?,A
low,low,excellent,mid,stable,stable,stable,10,A
mid,mid,excellent,mid,stable,stable,mod-stable,10,S
mid,mid,excellent,high,stable,stable,stable,10,A
mid,low,excellent,high,stable,stable,mod-stable,10,A
low,mid,good,mid,stable,stable,unstable,10,A
mid,mid,excellent,mid,stable,stable,mod-stable,10,A
mid,mid,excellent,mid,stable,stable,unstable,10,A
mid,mid,excellent,mid,unstable,unstable,stable,10,S
mid,mid,good,high,stable,stable,stable,10,A
mid,mid,excellent,mid,stable,stable,stable,15,A
mid,mid,excellent,mid,stable,stable,stable,10,S
mid,low,good,mid,stable,stable,unstable,10,I
high,mid,excellent,mid,unstable,stable,unstable,05,A
mid,mid,excellent,mid,stable,stable,unstable,10,A
mid,mid,excellent,mid,unstable,stable,stable,10,A
mid,mid,excellent,mid,unstable,stable,stable,15,S
mid,mid,good,mid,unstable,stable,stable,15,A
mid,mid,excellent,mid,unstable,stable,stable,10,A
mid,mid,good,mid,unstable,stable,stable,15,S

Приложение 3 Фрагменты программного кода

```
//Фрагмент реализации интерфейсной части
public Graph GLEEGraph()
{
    Graph g = new Graph(graph.Name);
    g.GraphAttr.LayerDirection = LayerDirection.None;
    for (int i = 0; i < graph.FOgraph.Count(); i++)
    {
        for (int j = 0; j < graph.FOgraph[i].Count(); j++)
        {
            string vertex1 = graph.VertexList[i].Name;
            if (graph.VertexList[i].Label != "")
            {
                vertex1 = graph.VertexList[i].Name + " | value: " + graph.VertexList[i].Label;
            }
            string vertex2 = graph.FOgraph[i][j].Name;
            if (graph.FOgraph[i][j].Label != "")
            {
                vertex2 = graph.FOgraph[i][j].Name + " | value: " + graph.FOgraph[i][j].Label;
            }
            g.AddEdge(vertex1, vertex2);
        }
    }
    return g;
}

public Graph GLEEGraph(string vertexName)
{
    Graph g = GLEEGraph();
    if (graph.SearchVertex(vertexName) != -1 && graph.Count > 1)
    {
        (g.FindNode(vertexName) as Node).Attr.Color = Color.Red;
        (g.FindNode(vertexName) as Node).Attr.Fontcolor = Color.Red;
    }
    return g;
}

public void TreeViewGraph(TreeView treeView)
{
    List<Vertex> sortedVertexList = graph.SortedVertexList();
    List<int> checkedVertexList = new List<int>(sortedVertexList.Count);
    int k = 0;
    while (k < sortedVertexList.Count)
    {
        checkedVertexList.Add(graph.ReverseFOgraph[k].Count);
        if (checkedVertexList[k] == 0)
        {
            checkedVertexList[k]++;
        }
        k++;
    }
```

```
    treeView.BeginUpdate();
    treeView.Nodes.Clear();
    BuildTreeViewFromTreeGraph(treeView.Nodes, sortedVertexList,
checkedVertexList);
    treeView.EndUpdate();
}

private void BuildTreeViewFromTreeGraph(TreeNodeCollection treeNode,
    List<Vertex> sortedVertexList, List<int> checkedVertexList)
{
    int i = 0;
    foreach (Vertex vertex in sortedVertexList)
    {
        int vertexIndex = graph.SearchVertex(vertex.Name);
        if (checkedVertexList[vertexIndex] > 0)
        {
            treeNode.Add(graph.VertexList[vertexIndex].Name);
            checkedVertexList[vertexIndex]--;
            List<Vertex> sortedChildList =
graph.SortedVertexList(graph.FOgraph[vertexIndex]);
            BuildTreeViewFromTreeGraph(treeNode[i].Nodes, sortedChildList,
checkedVertexList);

            i++;
        }
    }
}


    public TreeGraph ParentsSubGraph(string VertexName)
{
    return ParentsSubGraph(VertexName, "parents");
}


private TreeGraph ParentsSubGraph(string VertexName, string graphName)
{
    int VertexIndex = SearchVertex(VertexName);
    TreeGraph resultGraph = new TreeGraph("none");
    if (VertexIndex != -1)
    {
        resultGraph.Name = graphName + "Of" + VertexName;
        resultGraph.AddVertex(VertexList[VertexIndex]);
        List<Vertex> VertexParentsList = VertexParents(VertexName);
        foreach (Vertex Vertex in VertexParentsList)
        {
            resultGraph.AddVertex(Vertex);
            resultGraph.AddEdge(Vertex.Name, VertexName);
        }
    }
    return resultGraph;
}
```

```
public TreeGraph AncestorsSubGraph(string VertexName)
{
   int VertexIndex = SearchVertex(VertexName);
   TreeGraph resultGraph = new TreeGraph("none");
   if (VertexIndex != -1)
   {
      resultGraph = ParentsSubGraph(VertexName, "ancestors");
      int parentsCount = resultGraph.VertexList.Count();
      for (int i = 1; i < parentsCount; i++)
      {
         TreeGraph subGraph = AncestorsSubGraph(resultGraph.VertexList[i].Name);
         if (subGraph.Count > 1)
         {
            //Vertex Vertex = subGraph.VertexList[0];
            for (int j = 1; j < subGraph.VertexList.Count(); j++)
            {
               Vertex subVertex = subGraph.VertexList[j];
               resultGraph.AddVertex(subVertex);
               //resultGraph.AddEdge(Vertex.Name, subVertex.Name);
            }
            for (int j = 0; j < subGraph.VertexList.Count(); j++)
            {
               Vertex Vertex = subGraph.VertexList[j];
               for (int k = 0; k < subGraph.ReverseFOgraph[j].Count(); k++)
               {
                  resultGraph.AddEdge(subGraph.ReverseFOgraph[j][k].Name,
Vertex.Name);
               }
            }
         }
      }
   }
   return resultGraph;
}


public List<Vertex> SortedVertexList()
{
   return
      (
      from vertex in VertexList
      orderby DescendantsSubGraph(vertex.Name).Count descending
      select vertex
      ).ToList();
}


public List<Vertex> SortedVertexList(List<Vertex> vertexList)
{
   return
```

```
        (
        from vertex in vertexList
        orderby DescendantsSubGraph(vertex.Name).Count descending
        select vertex
        ).ToList();
}


public TreeGraph RelativesSubGraph(string VertexName)
{
   return RelativesSubGraph(VertexName, "relatives");
}

//метод ближайшего соседа
private double NearestNeighbor(MatchPair MP, Precedent Q, Precedent C, Metric
metric)
{
         if (metric != null)
   {
      _similarityCalc.Metric = metric;
   }
   else
   {
      _similarityCalc.Metric = EuklidMetrics;
   }
         _results.Clear();

         List<double> maxCoordDistances =
_coordDistanceCalc.GetMaxCoordDistances(_root);
         /// непосредственно расчёт, метод ближайшего соседа

         List<double> coordDistances = _coordDistanceCalc.GetCoordDistances(MP);

         double similarity = _similarityCalc.GetSimilarity(coordDistances,
maxCoordDistances);

}

//реализация Евклидовой метрики

public class EuklidMetrics : Metric
{
   public override double GetDistance(IList<double> coordDistances)
   {
      double result = 0;

      foreach (double d in coordDistances)
      {
         result += d * d;
      }

      result = Math.Sqrt(result);
```

```
        return result;
    }
}


//формирование парных соответствий
 private List<MatchPair> StructCompare(ArrayList APairList, TreeNode ANode1,
TreeNode ANode2)
{


    for (int i = 0; i <= ANode1.Nodes.Count - 1; i++)
    {
        if (ANode1.Nodes[i].FirstNode != null) EkzListNode1.Add(ANode1.Nodes[i]);
        else
        {
            for (int j = 0; j <= ANode2.Nodes.Count - 1; j++)
            {
                if (ANode2.Nodes[j].FirstNode != null)
                {
                    bool fl = false;
                    for (int k = 0; k <= EkzListNode2.Count - 1; k++)
                    {
                        NewNode1 = (TreeNode)(EkzListNode2[k]);
                        if (NewNode2.Text == ANode2.Nodes[j].Text) fl = true;
                    }
                    if (!fl) EkzListNode2.Add(ANode2.Nodes[j]);
                }
                else
                {
                    PairList.Add("<" + ANode1.Nodes[i].Text + ", " + ANode2.Nodes[j].Text +
">");
                }
            }
        }
    }
    if (PairList.Count != 0)
        for (int i = 0; i <= APairList.Count - 1; i++)
            for (int j = 0; j <= PairList.Count - 1; j++)
            {
                NewPair.Add((string)(APairList[i]) + ", " + (string)(PairList[j]));
            }
    else NewPair = APairList;

    if (EkzListNode1.Count != 0 && EkzListNode2.Count != 0)
    {
        PairList.Clear();
        for (int i = 0; i <= EkzListNode1.Count - 1; i++)
        {
            NewNode1 = (TreeNode)(EkzListNode1[i]);
            for (int j = 0; j <= EkzListNode2.Count - 1; j++)
```

```
            {
                NewNode2 = (TreeNode)(EkzListNode2[j]);
                if (NewNode1.Text == NewNode2.Text)
                {
                    for (int k = 0; k <= NewPair.Count - 1; k++)
                    {
                        PairList.Add((string)(NewPair[k]) + ", " + "<" + NewNode1.Text + ", " +
NewNode2.Text + ">");
                    }
                    NewPair.Clear();
                    NewPair = SetPair(PairList, NewNode1, NewNode2);
                }
            }
        }
    }
    return NewPair;
}

//Преобразование дерева
 private TreeNode ModifyTree(string APair)
{
    TreeNode NewTree = new TreeNode();
    string[] rules;
    string[] code;
    string[] pairs;
    string NewCode;
    string NewRule = "";

    rules = tvRules.Nodes[0].Text.Split('&');
    for (int j = 0; j <= rules.Length - 1; j++)
    {
        code = rules[j].Split('<', '>');
        pairs = APair.Split('<', '>');
        NewCode = CheckCode(code[0].Trim(), pairs);
        if (NewCode != "")
        {
            if (NewRule != "")
                NewRule += " & " + NewCode + rules[j].Trim()[rules[j].Trim().Length - 1];
            else NewRule = NewCode + rules[j].Trim()[rules[j].Trim().Length - 1];
        }
    }
    if (NewRule != "")
    {
        NewTree.Text = NewRule;
    }

    ModifyNode(NewTree, tvRules.Nodes[0], APair);
    return NewTree;
}
//Преобразование узла дерева.
private TreeNode ModifyNode(TreeNode AParentNode, TreeNode ANode, string APair)
{
```

```
    TreeNode NewNode = new TreeNode();
    string[] rules;
    string[] code;
    string[] pairs;
    string NewCode;
    string NewRule = "";

    for (int i = 0; i <= ANode.Nodes.Count - 1; i++)
    {
        rules = ANode.Nodes[i].Text.Split('&');
        for (int j = 0; j <= rules.Length - 1; j++)
        {
            code = rules[j].Split('<', '>');
            pairs = APair.Split('<', '>');
            NewCode = CheckCode(code[0].Trim(), pairs);
            if (NewCode != "")
            {
                if (NewRule != "")
                    NewRule += " & " + NewCode + rules[j].Trim()[rules[j].Trim().Length - 1];
                else NewRule = NewCode + rules[j].Trim()[rules[j].Trim().Length - 1];
            }
            if (ANode.Nodes[i].Text[0] == 'd') NewRule = ANode.Nodes[i].Text;

        }
        if (NewRule != "")
        {
            AParentNode.Nodes.Add(NewRule);
            if (ANode.Nodes[i].Text[0] != 'd')
ModifyNode(AParentNode.Nodes[AParentNode.Nodes.Count - 1], ANode.Nodes[i],
APair);
            NewRule = "";
            NewNode = AParentNode.Nodes[AParentNode.Nodes.Count - 1];
        }
        else AParentNode.Nodes.Add("Rule destroyed!");
    }
    return NewNode;
}

//Объединение парных соответствий
private ArrayList FinallyUnion(ArrayList APairList)
{
    ArrayList ObjectList = new ArrayList();
    ArrayList NewPair = new ArrayList();
    string res;
    string txt, txt2;
    string[] objects, objects2;
    for (int i = 0; i <= APairList.Count - 1; i++)
    {
        txt = (string)(APairList[i]);
        objects = txt.Split('<', '>');
        for (int j = i + 1; j <= APairList.Count - 1; j++)
        {
```

```
            txt2 = (string)(APairList[j]);
            objects2 = txt2.Split('<', '>');
            res = CheckStrings(objects, objects2);
            NewPair.Add((string)APairList[i] + res);
        }
    }
    int max = 0;
    string s1;
    for (int i = 0; i <= NewPair.Count - 1; i++)
    {
        s1 = (string)(NewPair[i]);
        if (s1.Length > max) max = s1.Length;
    }

    for (int i = 0; i <= NewPair.Count - 1; i++)
    {
        s1 = (string)(NewPair[i]);
        if (s1.Length < max)
        {
            NewPair.Remove(NewPair[i]);
            i = i - 1;
        }
    }
    return NewPair;
}
```

```
public List<CaseResult> Retrieve(OntData O, double k1, double k2, Precedent Q)
{
    List<CaseResult> result = new List<CaseResult>();
    foreach (Precedent precedent in OntData.Precedents)
    {
        CaseResult cRes = new CaseResult();
        List<MatchPair> MP = new List<MatchPair>();

        MP = StructCompare(O.Hierarchy, Q.Graph, precedent.Graph)
        Sstruct(MP, k1);
        if (MP.Count == 0)
        {
            continue;
        }
        else
        {
            MatchPair maxMP = MP.Max(item => item.Similarity);
            double Snn = NearNeighbor(maxMP, Q, C, null);
            if (Snn < k2)
            {
                continue;
            }
```

```
            else
            {
                cRes.Name = Precedent.Name;
                cRes.Str = maxMp.Similarity;
                cRes.NN = Snn;
                cRes.Res = Precedent.Result;
                result.Add(cRes);
            }

        }
    }
    return result;

    }
}
```