МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение   
высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет\_Информационных систем и технологий\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра\_\_\_\_\_Информационные системы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дисциплина\_\_Интеллектуальный анализ данных и процессов\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**КУРСОВОЙ ПРОЕКТ (РАБОТА)**

Тема\_\_\_Автоматизация распределений заданий исполнителям на основе онтологии\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Выполнил студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_\_\_\_Н.П.Иванова\_\_\_\_\_\_\_\_\_/

подпись инициалы, фамилия

Курс\_\_\_\_\_\_\_\_\_2\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Группа\_\_\_\_ИСЭмд-21\_\_\_\_\_\_\_\_\_\_

Направление/ специальность\_\_\_09.04.03\_\_Прикладная информатика\_\_\_\_\_\_\_\_\_\_

Руководитель \_\_\_\_\_\_\_\_профессор, д.т.н, доцент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

должность, ученая степень, ученое звание

\_\_\_\_\_\_\_\_\_\_\_\_\_\_Афанасьева Татьяна Васильевна\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

фамилия, имя, отчество

Дата сдачи:

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_\_г.

Дата защиты:

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_\_г.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ульяновск

20\_\_ г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение   
высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет\_Информационных систем и технологий\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра\_\_\_\_\_Информационные системы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дисциплина\_\_Интеллектуальный анализ данных и процессов\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ (РАБОТУ)**

студенту \_\_\_ ИСЭмд-21\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ивановой Н.П.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

группа фамилия, инициалы

Тема проекта (работы)\_\_\_\_ Автоматизация распределений заданий исполнителям на основе онтологии \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

Срок сдачи законченного проекта (работы) «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_\_г.

Исходные данные к проекту (работе)\_\_\_\_Задание: разработать и исследовать \_\_\_\_\_

(базовое предприятие, характер курсового проекта (работы):

распределение исполнителям с применением онтологии \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

задание кафедры, инициативная НИР, рекомендуемая литература, материалы практики)

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

Содержание пояснительной записки (перечень подлежащих разработке вопросов)

\_\_\_\_ Описание предметной области; Техническое задание; Описание процесса проектирования и реализации ПО; Руководство пользователя. \_\_\_\_

Перечень графического материала (с точным указанием обязательных чертежей)

Диаграммы IDEF0, UML, архитектура ПО и компонент связей с внешней средой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель \_\_\_\_профессор\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_Т.В. Афанасьева\_\_/

должность подпись инициалы, фамилия

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_г

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **/\_\_\_\_\_\_\_\_\_\_** Н.П.Иванова **\_\_\_\_/**

подпись инициалы, фамилия

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_г

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение   
высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

**ОТЗЫВ  
руководителя на курсовой проект (работу)**

студента \_\_\_\_Ивановой Надежды Петровны\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

фамилия, имя и отчество

Факультет\_\_\_ФИСТ\_\_\_\_\_\_\_\_\_\_\_группа\_\_ИСЭмд-21\_\_\_\_\_курс\_\_\_\_\_2\_\_\_\_\_\_\_\_\_\_\_

Дисциплина\_\_\_\_\_\_Интеллектуальный анализ данных и процессов\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Тема проекта (работы) \_\_\_\_\_ Автоматизация распределений заданий исполнителям на основе онтологии\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Отмечаются следующие моменты: актуальность темы исследования; соответствие содержания и структуры курсовой работы ее теме; степень разработанности проблемы, наиболее интересно исследованные вопросы. Оценивается степень самостоятельности и инициативы студента; умение пользоваться различными источниками информации; уровень его теоретической подготовки; умение анализировать научные материалы, делать практические выводы; знание основных концепций, научной и специальной литературы по избранной теме. Содержится оценка проекта (работы) руководителем.

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

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

Руководитель\_\_профессор, д.т.н., доцент\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ /\_ Т.В. Афанасьева\_\_/

должность, учёная степень, ученое звание подпись инициалы, фамилия

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_\_г.

**Оглавление**

[Введение 5](#_Toc30524668)

[1. Анализ известных решений 7](#_Toc30524669)

[2. Постановка задачи распределения 19](#_Toc30524670)

[3. Проектирование системы 25](#_Toc30524671)

[А. Анализ аналогов решения обозначенных в предыдущем разделе задач. 25](#_Toc30524672)

[Б. Cистемное и математическое описание конкретного метода (методов) ИАД для решения поставленных задач. 25](#_Toc30524673)

[В. Описание контрольного примера для каждого метода интеллектуального анализа. 26](#_Toc30524674)

[Г. Проектирование и описание системы хранения данных и систем знаний. 27](#_Toc30524676)

[Д. Проектирование архитектуры программной системы. 28](#_Toc30524677)

[4. Реализация программного обеспечения 34](#_Toc30524678)

[А. Описание инструментальных средств. 34](#_Toc30524679)

[Б. Характеристики программной системы. 34](#_Toc30524680)

[В. Разработка и описание пользовательского интерфейса. 35](#_Toc30524681)

[Г. Описание используемых компонент системы. 35](#_Toc30524682)

[5. Инструкция пользователю 38](#_Toc30524683)

[Заключение 40](#_Toc30524684)

[Список используемых источников 41](#_Toc30524685)

[Приложение 1 44](#_Toc30524686)

[Листинг программы 44](#_Toc30524687)

# 

# Введение

В настоящее время предметом исследований большинства современных разработчиков информационных систем являются методы и подходы, чтобы построить не просто распределенные информационные системы, но и интеллектуальные, предусматривающие возможность их адаптации к конкретной предметной области. Подобная потребность появилась при рассмотрении распределенной системы отслеживания задач (таск-трекер) по технике Помодоро, чтобы исключить необходимость в роли человека, распределяющего задачи, и автоматизировать этот процесс оптимально. Такая потребность возникает и во множестве других таск-трекеров.

Существует множество инструментов для исследований и разработки таких систем, анализа больших массивов данных с учетом специфики предметной области. Одной из наиболее универсальных методик представления экспертных знаний, в данном случае знаний при распределении заданий по исполнителям, полностью описывающие семантически, является онтологический подход.

В статье [[13](#_az13aw79ygda)] описывают ситуации, для которых применяются онтологии:

* необходимость достижения общего понимания структуры информации широким кругом людей или программными агентами;
* постоянная необходимость повторного использования знаний и анализа предметной области;
* образовать базу знаний в совокупности с оригинальным набором экземпляров классов.

Актуальность применения онтологий для реализации менеджмента описана в статье «Онтология для управления проектами: обследование» [[14](#_fbqlbsjopug9)], где представлен систематический обзор литературы, посвященный онтологиям управления проектами. Автор выявил отсутствие стандартизации в терминологии и концепциях, отсутствие систематического моделирования областей и использования онтологий, главным образом, в прототипных системах онтологий, которые затрагивают довольно ограниченные аспекты процессов управления.

Цель разработки ПО: уменьшить трудоемкость системы управления задачами (распределить, выполнить) для сотрудников IT-компании с применением онтологии.

Задачи ПО:

* Провести анализ прикладной области IT-компании, описать обзор применения онтологии в бизнес-процессах;
* Изучить методы и способы распределения задач по исполнителям, выявить особенности;
* Спроектировать автоматизированную систему (Создать диаграммы IDEF0, диаграммы UML, Техническое задание);
* Провести предобработку данных, представления данных в статическом и динамическом виде;
* Построить онтологию ПрО задач в компании;
* Разработать интерфейс оптимального распределения задач по исполнителям;
* Разработать и протестировать программную систему распределения задач;
* Создать документацию ПО (инструкция пользователя).

## 

# Анализ известных решений

Для начала стоит изучить, как решают менеджеры задачу разумной загрузки сотрудников. В статье [[1](#_5wqyylszeih9)] описаны действия руководителям, а именно:

1. планирование загрузки исполнителя
   1. замерить производительность труда каждого исполнителя
   2. замерить время для решения задачи
   3. вычислить количество решаемых задач каждого исполнителя за необходимый период (например, день)
2. распределение задач сотрудников
   1. охарактеризовать планируемые задачи по критериям из таблицы 1
   2. определить оценочную трудоемкость решения типичных задач - в человеко-часах или человеко-днях
   3. планировать время, необходимое для решения программных и сезонных задач
   4. учесть и время на решение постоянных задач
   5. рассчитать для каждой должности ежедневную и ежемесячную оценочную нагрузку (штатное расписание)
   6. При необходимости перераспределить задачи с перегруженных сотрудников или перераспределить сотрудников по имеющимся должностям в соответствии с их производительностью труда и реальной квалификацией

Таблица 1.Классификационные признаки управленческих решений и задач

|  |  |  |  |
| --- | --- | --- | --- |
| **N**  **группы** | **Классификацион-**  **ный признак** | **Группы управленческих решений и соответствующих задач** | **Предпочтительный**  **уровень сотрудников-**  **исполнителей** |
| **1** | **Значимость цели и**  **длительность**  **действия** | 1) стратегические;  2) тактические;  3) операционные | 1) руководитель;  2) среднее звено;  3) рядовые сотрудники |
| **2** | **Сфера воздействия** | 1) глобальные;  2) локальные | 1) руководитель;  2) среднее звено,  рядовые сотрудники |
| **3** | **Степень**  **повторяемости**  **проблемы** | 1) традиционные;  2) нетипичные | 1) рядовые сотрудники;  2) среднее звено и/или  руководитель |
| **4** | **Периодичность**  **возникновения** | 1) постоянно;  2) по программе (в  заранее известное время);  3) сезонно;  4) эпизодически;  5) разово | Рядовые сотрудники и  среднее звено (для  всех случаев) |
| **5** | **По**  **прогнозируемости**  **результатов**  **реализации** | 1) с определенным  результатом;  2) с вероятностным  исходом;  3) с неопределенным  результатом | 1) рядовые сотрудники;  2) среднее звено;  3) руководитель |
| **6** | **Число альтернатив** | 1) бинарные;  2) малоальтернативные;  3) многоальтернативные | 1) рядовые сотрудники;  2) среднее звено;  3) руководитель |
| **7** | **Степень**  **определенности**  **ситуации** | 1) определенность;  2) неопределенность;  3) риск | 1) рядовые сотрудники;  2) среднее звено;  3) руководитель |
| **8** | **Метод разработки**  **решения** | 1) формализованные;  2) неформализованные | 1) рядовые сотрудники;  2) среднее звено |
| **9** | **Количество**  **критериев выбора** | 1) однокритериальные;  2) многокритериальные | 1) рядовые сотрудники;  2) среднее звено |
| **10** | **Степень**  **уникальности** | 1) рутинные;  2) селективные;  3) адаптивные;  4) инновационные | 1) рядовые сотрудники;  2) рядовые сотрудники;  3) среднее звено;  4) руководитель |
| **11** | **Место и функции в**  **процессе**  **управления** | 1) информационные;  2) организационные;  3) технологические | 1) руководитель;  2) среднее звено;  3) рядовые сотрудники |
| **12** | **Основания для**  **принятия решения** | 1) интуитивные;  2) решения на суждениях;  3) рациональные | 1) руководитель;  2) среднее звено;  3) рядовые сотрудники |
| **13** | **Тип личности**  **менеджера,**  **принимающего**  **решения** | 1) импульсивный;  2) рискованный;  3) инертный;  4) осторожный;  5) уравновешенный | 1) руководитель;  2) руководитель;  3) рядовые сотрудники;  4) рядовые сотрудники;  5) среднее звено |

Для оптимальной загрузки сотрудников автор предлагает несколько принципов.

* Согласно "Бритве Оккама"[[1]](#footnote-1) новую, но периодическую задачу поручить уже имеющимся подразделениям или сотрудникам.
* В принципе Родена[[2]](#footnote-2) сосредоточиться на жизненно важном самому, а остальные задачи делегировать. Уточненная версия этого принципа содержится в матрице Эйзенхауэра. Все дела в ней классифицируются по степени важности и срочности. Автор советует внедрить матрицу в работу каждого сотрудника для личного контроля.
* Работу предприятия анализировать по принципу Парето[[3]](#footnote-3) для принятия решений и при выполнении проектной работы. В результате руководитель получает удобный инструмент контроля за загрузкой сотрудников и одновременно инструмент коррекции этой загрузки.

Большинство решений по планированию и распределению задач делается на веб-сайтах. Так называемые планировщики задач или   
таск-менеджеры помогают человеку ничего не забыть, расставить приоритеты и назначить ответственных. Занесение задачи в таск-менеджер не гарантирует ни ее выполнения, ни лучшего понимания всех деталей процесса. Поэтому важно, с одной стороны, не переусердствовать в планировании (иногда оно занимает больше времени, чем сами дела), с другой — раз завели таск-менеджер, кто-то за ним должен постоянно приглядывать. Например, в одних программах задачи выглядят как карточки, которые можно перетягивать, в других — как графы, а в-третьих — как гигантские таблицы.

SEO-специалисты одной web-студии[[2](#_ugowg1sf5qhs)] протестировали множество систем управления проектами и задачами (таск-менеджеры), чтобы выбрать хорошие идеи для собственной разработки.

Особенность работы в web-студии – быстрота и одновременная работа над несколькими проектами. Многие проекты находятся на разных этапах. И сотрудники вынуждены жёстко приоритизировать задачи, помнить о многих незначительных деталях. В исследовании рассмотрены и поделены на категории таск-менеджеры:

1. Система = таск-менеджер
   1. TrackStudio (любая СУБД, нет шаблонов)
   2. Redmine (бесплатно, минималистичный дизайн, простое меню и хороший набор понятных настроек)
   3. JIRA (если больше ориентирована на разработку; работа в своём проекте и в то же время в единой функциональной системе)
2. Таск-менеджер – модуль системы (CRM/ERP/xRM)
   1. amoCRM (решает проблемы отдела продаж, нельзя создать вехи и подзадачи, неочевидный интерфейс)
   2. Битрикс24 (целый корпоративный портал с модулем управления продажами и CRM)
   3. Мегаплан (изначально как система управления проектами, не подходит только компаниям со специфическими требованиями к ИС, нет чек-листа)
3. «профильные» таск-менеджеры
   1. Pyrus (отличается от системы управления проектами, документооборотом и бизнес-процессами; возможности интеграции с множеством сервисов и наличие API; после завершения задачи убирается привязка к проекту)
   2. iQ300 (для небольших компаний; понятный интерфейс и хорошие мануалы; нельзя отобразить чек-листы в поле проекта, они в самой задаче)
   3. WorkFlow Soft (простая настройка несложных бизнес-процессов без использования нотации BPMN)
4. Лучше всего справились с поставленными задачами:
   1. разработка команды автора PTYSH (Специальные функции для Веб-студий, SEO-компаний и Фрилансеров, чек-листы по SEO, Бесплатно для команды из 3-х человек)
   2. Basecamp (самая наглядная и неформальная, для небольшой компании)
   3. Wrike (можно создать группу пользователей; подзадачи организованы как чек-лист; расширенные возможности визуализации в виде таблицы и на диаграмме Ганта; настройка отображения всего дерева задач и отмены отображений для завершенных задач)
   4. Asana (простой старт, прозрачная логика работы, удобный интерфейс, но на английском, хороший мануал, возможность управления горячими клавишами, возможность привязки задачи к нескольким проектам)

Описанное исследование популярных таск-менеджеров представляет системы, работающие не только как таск-менеджеры, но и более узкие как багтрекер или система управления инцидентами и тикетами. К критериям разрабатываемого ПО из системы для web-студий можно выделить:

* вложенные иерархии проект-задача-подзадача
* возможность назначения и изменения исполнителей и наблюдателей
* возможность изменения названия сущностей.

Все популярные системы однако не автоматизируют работу распределения задач. Поэтому далее рассмотрено применение решений поставленной цели интеллектуального анализа данных.

В работе Мошкина кафедры «Информационные системы» УлГТУ изложены теоретические и практические вопросы в области моделирования слабоструктурированных информационных ресурсов. Рассматриваются основные подходы к моделированию таких ресурсов на основе онтологий и грануляции временных рядов. А именно представлена подсистема на основе интеграции нечеткой онтологии (FuzzyOWL) и систем продукционных правил (SWRL) с использованием механизма прецедентов, которые «представляют собой простейшие логические структуры, связывающие условия со следствием» [[4](#_8nzspn3jmg2d)].

Для логического вывода экспертных рекомендаций в порядке их релевантности авторы используют результаты анализа исходных параметров исследуемой системы и обработанной онтологии. Полное представление их разработанной системы можно видеть на рис.1.1

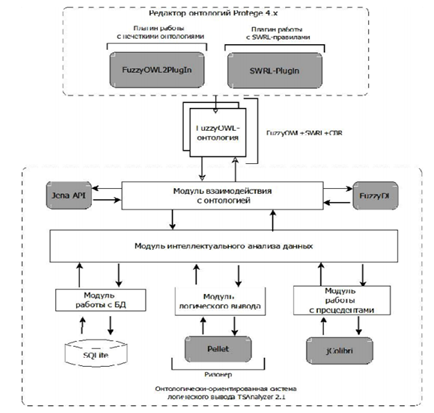


Рис.1.1 Архитектура онтологически-ориентированной системы логического вывода.

Заметим, что онтология используется на всех этапах жизненного цикла СППР. На начальном этапе разработки системы онтологии позволяют:

* структурировать ПО,
* выделить в онтологии основные концепты и их взаимосвязи,
* определить функции разрабатываемой системы,
* описать на концептуальном уровне вне конкретной программной реализации и физических средств представления
  + классы решаемых задач,
  + методы их решения
  + используемые данные.

На этапе программной реализации онтология может:

* разрабатывать структуры ЛХД
* связывает на высоком уровне к внешним источникам данных.
* определять структуры данных, используемые в программном коде системы.

В процессе работы онтология четко регламентирует функции СППР, а именно:

* манипулирует только данными, которые описаны в онтологии ПрО,
* решает только те задачи и только теми методами, которые представлены в онтологии задач,
* структурированно хранит данные решаемой задачи в ЛХД,
* формирует данные как контекст задачи для специализированной машины вывода => эффективнее автоматически решает некоторых вспомогательных задачи.

В другой статье [[6](#_Создание_семантического_веб-приложе)] также рассмотрена формализация функций менеджера на основе разработанного приложения.

* В приложении есть люди (FOAF) и проекты (DOAP)
* У каждого человека есть аккаунт и набор компетенций, которыми он владеет.
* В рамках проекта имеются задачи
* Каждая задача требует набора компетенций.
* Набор компетенций имеет уровни в виде численного представления.

Выявленные автором этой статьи условия помогут в построении онтологии. Согласно [[4]](https://docs.google.com/document/d/1o__CM6b_fC4x9gq2STOaRf6tykg2Imze/edit#heading=h.k77ohn8o02s1), [[5]](#_v0s2e4q3puil), наиболее формализованные онтологии представляют собой *логические теории, построенные на* произвольных логических *утверждениях* о понятиях аксиомах. Для описания таких формальных онтологии применяются различные логики (дескриптивные логики, модальные логики, логика предикатов первого порядка) и различные языки описания онтологии DAML+OIL, OWL, CycL, Ontolingua.

В другой статье [[7](#_Карпов_Л.Е._Методы)] по рассматриваемой предметной области автор рассматривает применение интеллектуальных систем поддержки принятия решений в организационном управлении. Предложены задачи, модели и методы управления принятием решений на основе онтологии. Описание онтологии произведено на языке OWL-DL, так как удовлетворяет требованиям ясности, декларативности, портативности, предметной независимости.

Показатели качества работы сотрудников измеряются по сбалансированной системе показателей (ССП или the Balanced Scorecard, BSC). Эта методологии помогает управлять стратегическим развитием компании с помощью «карты стратегии». Такая карта визуализирует стратегические цели, показатели и причинно-следственных связи бизнес-процесса. На ее основе анализируются перспективы, четыре направления: финансы, клиенты, внутренние процессы, обучение и рост [2].

Для разработки собственной онтологии выделены важные условия в организации. Цели отдельных подразделений должны совпадать с целью организации в целом. При этом цели и критерии могут противоречить друг другу и требовать согласования, но мы предполагаем, что цели согласованы.

В статье [[17](https://cyberleninka.ru/article/n/sistema-podderzhki-prinyatiya-resheniy-v-it-podrazdeleniyah-na-osnove-integratsii-pretsedentnogo-podhoda-i-ontologii)] улучшали передачу, хранение и извлечение знаний в области поддержки пользователей в IT-подразделениях. Ошибки пользователей консультантов-аналитиков описываются прецедентным подходом. Онтология описывает работу консультантов-аналитиков из трех таксономий: по бухгалтерскому учету, расчетам с персоналом, договорному блоку.

Авторы интегрируют прецеденты IT-консультирования с онтологиями предметной области и установливают ассоциативные связи прецедентов с концептами онтологии. Такой подход называется Case-Based Reasoning (CBR). Решение новых задач происходит из прошлых успешных прецедентов через их адаптацию к новым условиям. Когда рассматривается новая ситуация, система находит подобный прецедент в базе знаний как аналог решаемой задачи и пытается использовать решение найденного прецедента. Если необходимо, близкий прецедент адаптируется к текущей ситуации.

Терминалы (классы, не имеющие потомков) будут играть роль ключевых слов при индексировании прецедентов. Вес связи задается от концепта-родителя Ci(l-1) к концепту-ребенку Ci(l). Прецедент ассоциируется как терминальный концепт kwj (ключевое слово). Описанный алгоритм изображен на рис.1.2.

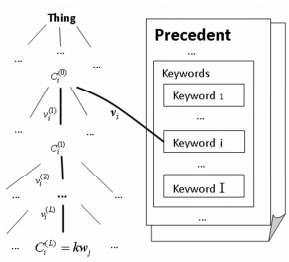


Рис. 1.2. Вычисление силы связи прецедента с ключевыми словами

Методы CBR уже применяются во множестве прикладных задач – в медицине, управлении проектами, для анализа и реорганизации среды, разработки товаров массового спроса с учетом предпочтений разных групп потребителей и т. д.

В работе [[5](#_l3fxarbztsob)] предложен метод принятия решений, основанный на совместном применении методов извлечения знаний и вывода по прецедентам. На момент написания статьи все существующие интегрированные системы подобного рода строят для себя модель данных как способ получения фонового знания. Меры близости определяют наиболее подходящих прецедентов и адаптации найденного решения. Для выбора прецедентов популярный метод "ближайшего соседа" авторы предлагают заменить на локальную контекстно-зависимую метрику. При ее построении может быть использована как предварительная кластеризация базы прецедентов, так и разбиение базы на классы эквивалентности с привлечением экспертного знания.

Выводы

В результате анализа рассмотрены решения поставленной задачи как с точки зрения специалистов, так и с использованием ПО, интеллектуальных средств анализа данных. Наиболее подходящие решения будут реализованы в курсовом проекте:

* Приведенная архитектура модели (рис.1.1) ляжет в основу данного проекта;
* Описанные основы в работе [1] помогут определить приоритетность данного проекта.
* Основные функции ПО: простой старт, прозрачная логика работы, удобный интерфейс, работа с группой пользователей;
* Желательные функции:
  + подзадачи организованы как чек-лист;
* без визуализации в виде таблицы и на диаграмме Ганта - не входит в задачу таск-менеджера;
* Возможные функции:
  + настройка отображения всего дерева задач и отмены отображений для завершенных задач
  + возможность управления горячими клавишами,
  + возможность привязки задачи к нескольким проектам.
* OWL используется для наиболее формализованных онтологий.
* Задача представления базы знаний по выбранной ПрО как часть решения задачи проекта решается смешанным образом. Например, CBR подход из онтологического и прецедентного подхода.
* локальная контекстно-зависимая метрика для критерия качества, аналогичный "ближайшему соседу".

# Постановка задачи распределения

## Цель разработки проекта в конкретной прикладной области для выбранных данных

Основными объектами проекта являются внутренние задачи IT-компании и сотрудники. Для задач имеются признаки области применения (домен), длительности исполнения (время), оценки сложности задания. Признаки сотрудников: области специализации (домен), время занятости, уровень квалификации.

Цель разработки: сопоставить признаки задачи соответственно признакам сотрудников с применением онтологии.

## Перечень и методы решаемых задач интеллектуального анализа для конкретных данных из выбранной прикладной области.

Методы решаемых задач интеллектуального анализа:

* моделирования бизнес-процессов;
* прецедентный подход;
* ограничения онтологии ПрО.

Для анализа данных будет применяться прецедентный подход на основе работы [5]. Для задачи сопоставления признаков объектов задач и сотрудников применяются меры близости. Предлагаемая локальная контекстно-зависимая метрика позволяет определить наиболее подходящих прецедентов и адаптации найденного решения. Предлагаемый способ построения метрики, а именно разбиение базы на классы эквивалентности с привлечением экспертного знания, будет осуществляться возможностями онтологии.

Так, в работе [23] авторы выделяют одним из важных аспектов представления знаний и управление знаниями в онтологических системах – наполнение онтологии знаниями и интеграция накопленных ранее знаний.

Главный элемент языка онтологии – это тройка, или триплет. Тройка представляет собой совокупность трех сущностей: Субъект, Объект, Предикат.

С математической точки зрения, тройка представляет собой экземпляр некоторого бинарного отношения. Отношение – это множество последовательностей, состоящих в точности из n элементов для некоторого заранее определенного натурального числа n. Если n = 2, то отношение называется бинарным, т.е. бинарное отношение представляет собой множество пар. Например, отношение «семейные пары» задает множество пар элементов вида («муж», «жена»).

Каждая тройка определяет одну пару из некоторого бинарного отношения, но кроме этого дополнительно задает еще имя отношения, т.е. если имеется пара («муж», «жена») отношения «семейные пары», то эту пару можно выразить тройкой («муж», «семейные пары», «жена»). Для полноценного описания, кроме задания отношений, необходимо еще задать ограничения на их содержимое. Например, для отношения «семейные пары» необходимо уточнить, что первый элемент каждой пары этого отношения должен быть мужчиной, а второй – женщиной. Для задания такого ограничения необходимо ввести понятия «мужчина» и «женщина», после чего задать ограничение, выражающее тот факт, что если имеется тройка вида («имя 1», «семейные пары», «имя 2»), то сущность «имя 1» должна быть экземпляром понятия «мужчина», а сущность «имя 2» – экземпляром понятия «женщина». Это делается посредством т.н. высказываний.

Для высказываний существует свой язык, включающий переменные и логические операции. В качестве логических операций могут выступать: логическое «или» (дизъюнкция), логическое «и» (конъюнкция) и логическое следование (импликация). Имеются также кванторы существования и всеобщности, позволяющие ограничивать область применения высказывания.

Для таких описаний онтологии консорциумом W3 разработаны специальные языки: RDF и OWL.

Язык OWL можно рассматривать как расширение языка RDF. Он позволяет описывать различные характеристики классов и свойств, которые обычно задаются как разного рода ограничения на структуру связей между своими экземплярами. Эти ограничения выражаются в виде предопределенных соотношений, называемых в языке OWL аксиомами.

В этом состоит основное отличие языка OWL от RDF (RDFS). Эти ограничения позволяют выражать в онтологии более тонкие вещи. Например, в языке OWL имеется аксиома owl:equivalentClass, которая позволяет сказать, что множества экземпляров двух данных классов совпадают. Например, можно задать класс с именем ГенеральныйДиректор, а можно еще сказать о классе ГлаваКомпании.

Также можно задавать ограничения для атрибутов (свойств) классов. Например, ограничения количества элементов (cardinality constraints) накладывают ограничения на мощность множества значений данного свойства, если в качестве субъекта выступает некоторый конкретный класс. Так, можно указать, что количество значений свойства Игрок класса ФутбольнаяКоманда должно равняться числу 11. Таких ограничений в языке OWL множество. Но все они подобраны таким образом, чтобы не снизить производительность алгоритма логического вывода по фактам, которые описаны в онтологии.

## Общие требования к данным и результатам интеллектуального анализа на уровне вход/выход.

Объектом исследования является открытая база данных отслеживания ошибок (баг-трекера) Redmine.

Предметом исследования являются инструменты онтологии в распределении задач (багов) по исполнителям.

БД сайта имеет ограниченное количество участников и один проект – Redmine (рис.2.1). Следовательно, структура системы [21] не такая большая, что позволит быстрее моделировать разные варианты выборки и проверить качество распределения задач исполнителям.



Рис.2.1 БД исполнителей задач Redmine и количество задач каждого из них

Рассматривая Redmine, создатели совершают все распределения вручную, причем часто и не назначают никого для выполнения задания, а именно значения Assignee – пустые (рис.2.2). Поэтому такая выборка подойдет в качестве экспериментов с распределением при добавление новой не распределенной задачи, с реальными значениями из БД сайта.

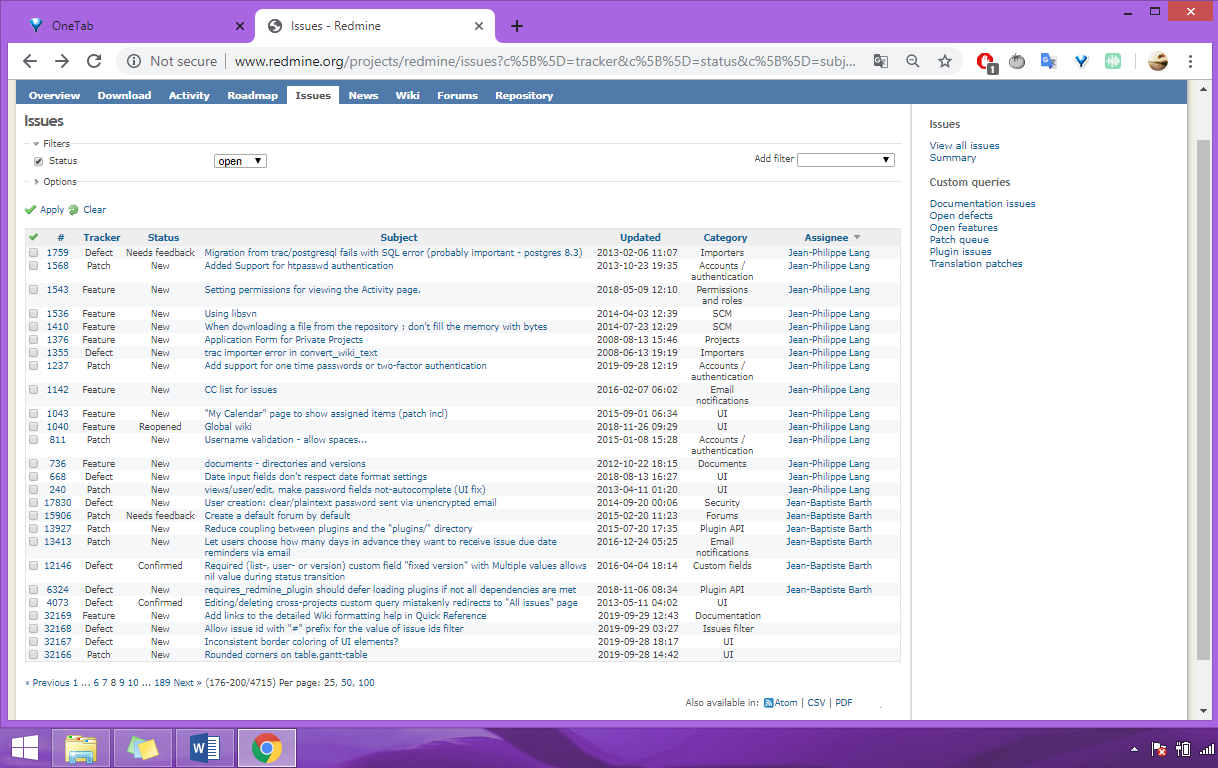


Рис.4.3. Баг-трекер Redmine и в конце списка не распределенные задачи

Уровни формирования онтологического знания:

* + - 1. метаонтологии общих категорий – контроль синтаксических понятий и конструкций предметных;
      2. онтология предметной области – набор понятий, используемых при решении интеллектуальных задач независимых от выбора метода решения;
      3. онтология задач – понятия, описывающие методы преобразования объектов предметной области, их сущность, последовательность выполнения и правила применения в конкретных ситуациях.

Следовательно, для решения задач интеллектуального анализа данных будет использоваться онтология предметной области.

Формализация онтологий происходит за счет построения семантических сетей, задающих отношения на множестве объектов знаний. Отношения представляют основные конструктивные элементы структуры знания. Далее в онтологии формируются аксиомы, правила и ограничения.

## Технико-экономическое обоснование проекта.

Не приводится.

## ТЗ с выбором конкретных методов решаемых задач интеллектуального анализа

### 1. Общие сведения

1.1. Полное наименование системы: Автоматизированная система распределений заданий по исполнителям на основе онтологии.

Условное обозначение системы: АИС «TasktrackerOnto».

1.2. Договор не заключался

1.3. Наименование предприятий разработчика: Иванова Н.П.

Наименование заказчика системы, их реквизиты: отсутствуют.

1.4. Перечень документов, на основании которых создается ИС: Техническое задание, задание к курсовому проекту.

1.5. Плановые сроки начала работ – начало семестра, и окончания работ – конец семестра.

1.6. Сведений об источниках и порядке финансирования работ нет, так как работы не финансируются.

1.7. Порядок оформления и предъявления заказчику результатов работ по созданию системы, её частей и отдельных средств не предъявляется.

### 2. Назначение и цели создания системы:

2.1. Разрабатываемая система предназначена для автоматизации работы менеджера при распределении заданий исполнителям с применением онтологии.

2.2. Цель создания системы: уменьшить трудоемкость системы управления задачами (распределить, выполнить) для сотрудников IT-компании.

### 3 Характеристика объекта автоматизации

3.1 Краткие сведения об объекте автоматизации

Данная система может использоваться для предприятий малого бизнеса в IT области. Она является частью системы управления задачами, ведет учет и распределение заданий на основе ограничений онтологии ПрО между пользователями, объединенными в группу.

Данные для онтологии используются из Административной БД сайта Redmine. БД сайта содержит задания по ошибкам (багам) во время разработки их веб-проекта.

3.2 Сведения об условиях эксплуатации и характеристиках окружающей среды:

– бесперебойная подача питания;

– защита от несанкционированного воздействия;

– доступ к сети Интернет (желательно).

Требования к процессам взаимодействия с ИС делается в соответствии со стандартами и законами.

### 4. Требования к системе

4.1. Требования к системе в целом

Требования к структуре и функционированию полноценной системы:

* подсистема распределения задач (распределить задачу по исполнителям, проверить ограничения по онтологии)
* подсистема выполнения задачи (отображение списка задач, выбор задачи из соответствующего списка, уведомление об исполнении задачи);
* подсистема хранилища (данные о пользователях и задачах и их взаимосвязи в виде свойств).

Описанные выше последние два пункта будут реализованы в данном проекте.

Требования к персоналу: владеть навыками пользования ПК и браузера на базовом уровне.

Требования к надёжности:

* обеспечить контроль и обработку ошибок пользователя;
* обеспечить возможность восстановления данных с внешнего накопителя после восстановления активного накопителя;
* БД системы необходимо резервировать минимум раз в месяц.

Требования к эргономике и технической эстетике:

* интерфейс системы должен быть удобным и простым;
* пакет программного обеспечения включает все необходимые библиотеки.

4.2. Требования к функциям:

* Учёт существующих задач: Менеджер может изменять список текущих задач организации (добавление, удаление). Необходимо обеспечить корректный отклик на соответствующий запрос.

Входные данные: список задач.

В результате на странице должны отображаться задачи с отметкой статуса исполнения, который определяется пользователем.

Действующие средства сбора, передачи и обработки информации: информация о текущих задачах заносится администратором в БД и передаётся исполнителям.

* Распределения актуальных задач среди пользователей: Формирование индивидов для объектов «задача», «организация», «сотрудник» и формирование свойств типов данных и объектных свойств для распределения актуальных задач.

Входные данные: класс задач и класс исполнителей.

В результате ограничений онтологии задач выявляют актуальные задачи, задачи со статусом исполнения.

Действующие средства сбора, передачи и обработки информации: информация о текущих задачах заносится администратором в БД и передаётся исполнителям.

4.3. Требования к видам обеспечения:

* Требования к информационному обеспечению: данные хранить в онтологии ПрО, реализовать многопользовательский режим доступа к данным.
* Требования к алгоритмическому обеспечению: разработать алгоритмы для суперпользователя, для пользователя с низкими правами доступа работа не меняется.
* Требования к программному обеспечению: вид предполагается desktop-приложения. Такие приложения также называют «оконными». В них используется графический интерфейс и ввод информации осуществляется при помощи клавиатуры и мыши. Данные приложения позволяют создавать практически любые по сложности решения, онтология в формате OWL.
* Требования к лингвистическому обеспечению: язык интерфейса – русский, язык программирования – python.
* Требования к техническому обеспечению: Процессор – Intel(R) Core 1.5ГГц и выше, Объём оперативной памяти – 1 Гб и выше, Объём HDD – 20 Гб и выше.

### 5. Состав и содержание работ по созданию системы.

5.1. Перечень стадий и этапов работ, сроки исполнения ведутся соответственно ГОСТ 34.201.

5.2. Вид и порядок проведения экспертизы технической документации выявить в ходе лабораторных работ по подготовке курсового проекта.

5.3 Программа обеспечения надежности предусматривает на стадиях разработки, изготовления и эксплуатации контроль частого резервного копирования проекта.

### 6 Порядок контроля и приемки системы

Не предъявляется.

### 7. Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие

7.1 Преобразование входной информации к машиночитаемому виду путем ввода данных с клавиатуры.

7.2 Сроки и порядок комплектования и обучения персонала в течение недели до опытной эксплуатации. За это время необходимо изучить руководство пользователя и практически освоить работу Системы.

# Проектирование системы

## Анализ аналогов решения обозначенных в предыдущем разделе задач.

В работе [[8](#_Список_используемых_источников)] для распределений заданий IT-компании hh.ru применяется машинное обучение на языке программирования python. Автор подчеркивает важность предобработки текста и предлагает следующие методы для создания выборки:

1. Программа [MyStem](https://tech.yandex.ru/mystem/) производит морфологический анализ текста на русском языке. Она умеет строить гипотетические разборы для слов, не входящих в словарь.

2. [НЛТК-data](http://promise.site.uottawa.ca/SERepository/datasets/cocomonasa.arff) (Стоп-слова, лемматизация та же)

Рассматриваемые автором модели машинного обучения умеют выделять признаки, оказавшие наибольшее влияние на распознавание, по имени. Однако есть и недостаток: уменьшение точности при распознавании сочетании имен.

Несмотря на популярность предметной области данного курсового проекта, больше аналогов не было найдено. Большинство систем с применением онтологии используются в задачах базы знаний для создания крупных порталов или чаще в контексте Text Mining, а именно автоматического аннотирования. Из описанных особенностей аналога связи между именами объектов как раз будут учитываться в данной системой средствами онтологии.

## Системное и математическое описание конкретного метода (методов) ИАД для решения поставленных задач.

Для задачи распределения будут использоваться инструменты онтологии.

Задачи будут иметь структуру:

Z = <D, T, P> при Z = {z1,... ,zn}, n–количество заданий

где

D – множество доменов, к которым относится задание (продукты компании, компоненты);

Т = {T0, Tend} – множество длительности исполнения задания, где

T0 – множество дат при постановке задания на исполнение,

Tend – множество дат после выполнения задания;

Р – множество оценок сложности задания.

Исполнители (assignee) будут иметь следующую структуру:

A = <D, T, P> при Z = {z1,... ,zn}, n–количество заданий

где

D – множество области специализации (продукты компании, компоненты, аналогично);

Т = {T0, Tend} – множество длительности занятости исполнителя, где

T0 – множество дат начала занятости,

Tend – множество дат после выполнения задания;

Р – множество уровня квалификации.

Меры точности и адекватности применяемых методов определяются в тестовом множестве. Популярные меры будут применены в проекте:

1. Accuracy – количество верно классифицированных объектов относительно общего количества всех объектов

Accuracy = T / N, t – true, n – number

1. F-мера стремится к нулю, если точность или полнота стремится к нулю.

F=2Precision×Recall/(Precision+Recall)

* 1. Точность – сколько объектов модель правильно определила к данному классу из всего множества объектов, которые она отнесла к этому классу:

Precision = TP / (TP+FP),

TP – true-positive, FP – false-positive

* 1. Полнота – сколько объектов модель правильно определила к данному классу из всего множества объектов, которые принадлежат этому классу в тестовой выборке (в экспертной системе):

Recall = TP / (TP+FN), FN – false-negative

## Описание контрольного примера для каждого метода интеллектуального анализа.

На вход будет поступать новая задача и информация о ней соответственно кортежу, описанному в следующем пункте.

В результате работы онтологии будет выбран исполнитель. Описание примеров представлено на этапе реализации в инструкции пользователя.

## Проектирование и описание системы хранения данных и систем знаний.

Источники с исходными данными будут, как формироваться вручную, так и извлекаться в виде csv, xml формата представления из открытой базы данных сайта Redmine (рис.3.1).

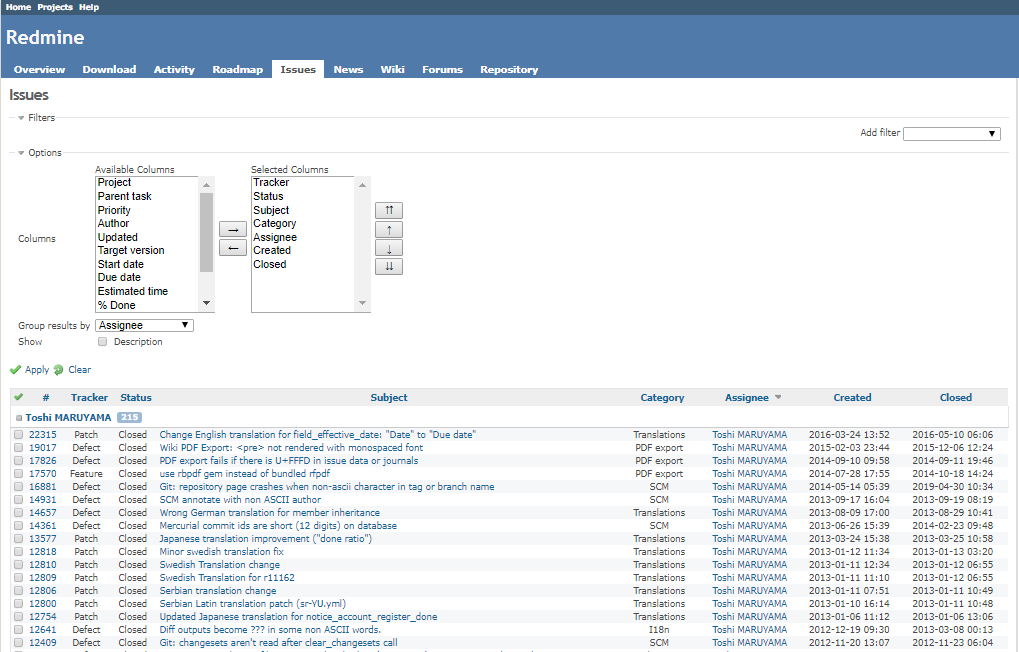


Рис.3.1. Страница отчетов сайта Redmine

В соответствие с ТЗ п.4.3 хранилище знаний – онтология в формате OWL. Онтология будет построена вручную с учетом малого опыта в использовании редакторов онтологий автором проекта.

В дальнейшем предполагается использовать другие способы ее построения:

* автоматизированное, где синтаксический анализ разбивается на несколько шагов:

1. выделение лексем;
2. морфологический анализ;
3. выделение простых вспомогательных структур;
4. выделение структур согласования, управления и объектов;
5. выделение набора жестких структур, описывающих отношения между понятиями.

* автоматическое аннотирование с JAVA.

Создание онтологий задач IT-специалистов вручную предполагает создание классов и добавлении ограничений. Так, задания представлены в виде кортежа в описаниях методов ИАД данного этапа проектирования.

## Проектирование архитектуры программной системы.

### 1.Концептуальная архитектура системы

Архитектура системы как модуль поведения описывается в нотации IDEF0. Такое функциональное описание бизнес-процессов представлено на рис.3.3.

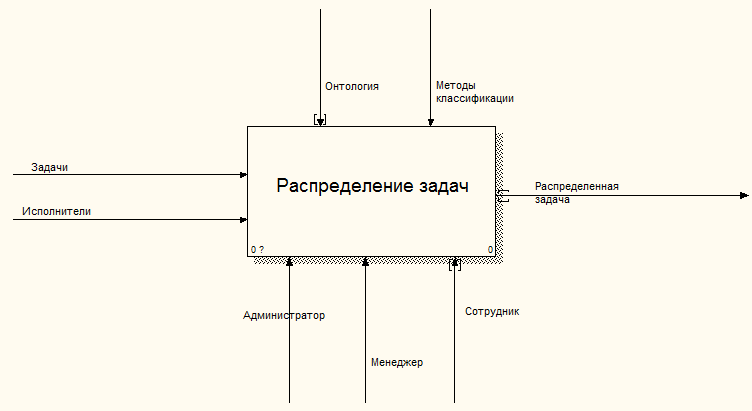


Рис.3.3 Диаграмма архитектуры системы

На вход получаем данные задачи и исполнителей. Они преобразуются с управляющими элементами: инструменты. Архитектура “черный ящик” снизу управляется пользователями. Далее их роли описываются подробнее.

### 2. Диаграмма вариантов использования

Диаграмма вариантов использования (рис.3.4) описывает то, что система будет делать в процессе ее функционирования. Она отображает работу следующих актёров[[4]](#footnote-4):

* сотрудник, который обладает прецедентами подсистемы выполнения задачи, которая в свою очередь связывается с подсистемой отсчета времени;
* менеджер, который в подсистеме статистики обладает прецедентом, имеющий связь расширения (extend) из подсистемы выполнения задач, чтобы иметь данные статуса задач. Также он имеет прецеденты подсистемы управления задачами, чтобы непосредственно эти задачи были созданы.

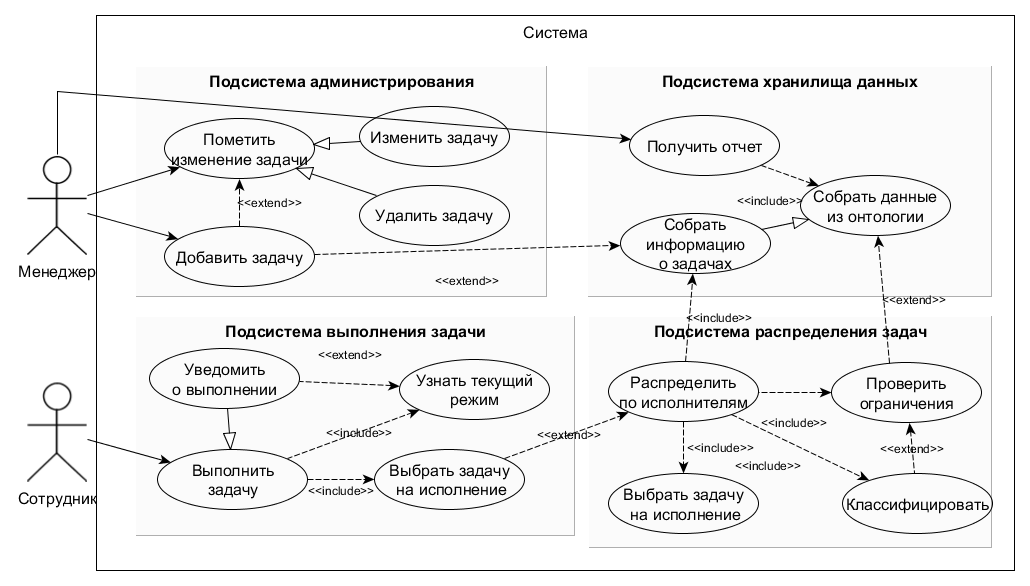


Рис. 3.4 UML-диаграмма прецедентов системы

### 3. Диаграмма классов

Диаграмма классов детально описывает структуру модели системы с использованием терминологии классов объектно-ориентированного программирования. В данном случае в диаграмме (рис.3.5) указано отношение композиции для пользователей (Users) для сотрудников (Workers) и руководителей (Organisators). Это отношение служит для выделения специальной формы отношения "часть-целое", при которой составляющие части в некотором смысле находятся внутри целого.

Пользователь должен быть авторизован, чтобы иметь доступ к классам задач (Tasks) и отделы (Departments), для этого используется отношение зависимости с проверкой разрешения (CheckPerms) к этим классам. Такие отношения чаще всего использованы в данной диаграмме.

Отношение обобщения описывает иерархическое строение классов и наследование их свойств и поведения. Так, класс выполнения задач (Doing\_Task) является потомком класса задач (Task), который наследует его свойства, которые являются открытыми, изображенные знаком «+».

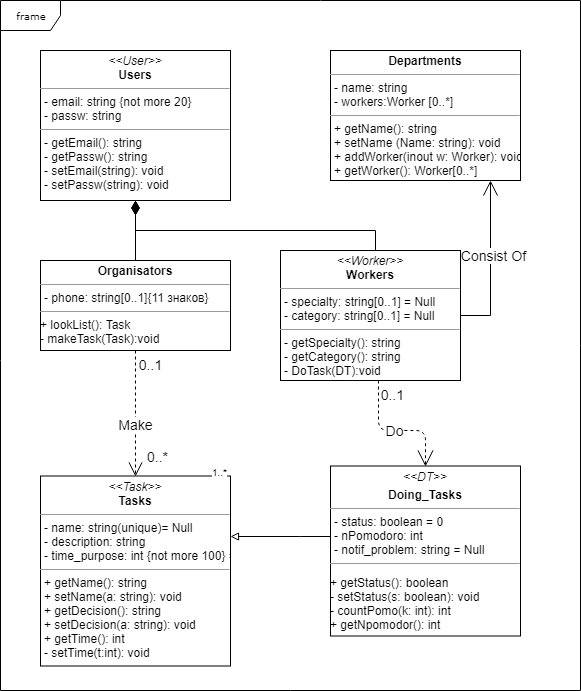


Рис.3.5 Диаграмма классов UML

### 4. Диаграмма состояний

Диаграмма состояний представляет собой состояние, соединенные переходами, характеризует жизненный цикл, который начинается при запуске. Диаграмма (рис.3.6) описывает работу распределения задачи с учетом создаваемой системы. Указаны роли «Менеджер» и «Сотрудник». «Система» выполняет большую часть задач, а значит, автоматизирует работу обеих сторон.

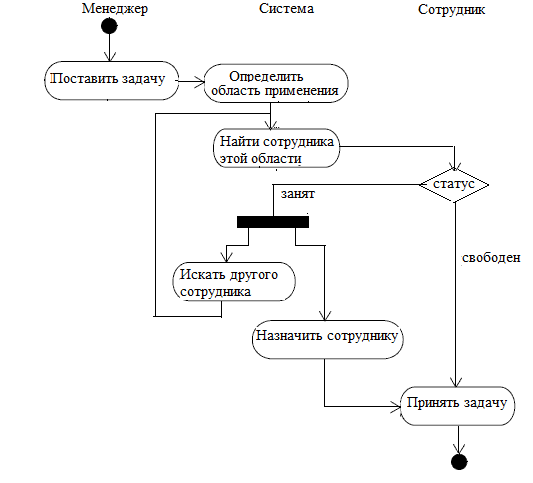


Рис.3.6 Диаграмма состояний

### 5. Диаграмма последовательности

Диаграмма последовательности (рис.3.7) отражает временные особенности передачи сообщений между частями программы, а также показывает время жизни классов, которые представлены в диаграмме классов (рис.3.5).

В данной диаграмме частями программы являются основные модули программной разработки: главный модуль, объединяющий в себе работу модуля связи с онтологией и модуля классификации, оптимизирующего распределение. В рамках данного курсового проекта эта часть не будет реализована.

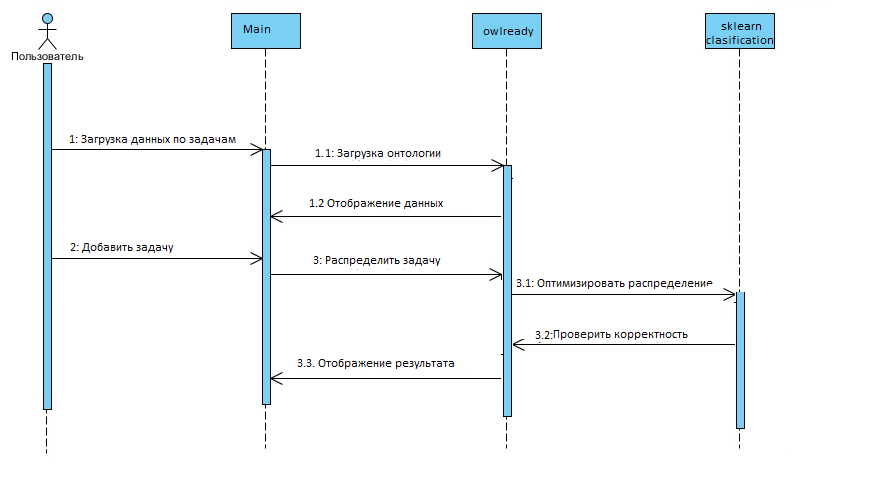


Рис.3.7 Диаграмма последовательности

### 

# Реализация программного обеспечения

## Описание инструментальных средств.

Инструментальные средства используются из требований к программному обеспечению: desktop-приложение в соответствии с ТЗ, онтология в формате OWL.

Операционная система подойдет любая, если включить модуль для поддержания соответствующих исполняемых файлов.

Желательно подключение к сети.

К ПК должны быть подключены клавиатура и мышь для взаимодействия с интерфейсом.

## Характеристики программной системы.

С учетом ТЗ п.4.3 язык программирования написан на python. Редактор для работы с ним официальной версии – IDLE 3.7 (64 bit).

Для проверки и визуализации построенной онтологии используется популярный редактор Protege.

Для вычисления объема программы применяется метрика Холстеда:

V = (N1+N2)log2(n1+n2) = (60+18)log2(40 + 8) = 131,

где n1 – число уникальных операторов программы, включая символы-разделители, имена процедур и знаки операций (словарь операторов);

n2 – число уникальных операндов программы (словарь операндов);

N1 – общее число операторов в программе (команды);

N2 – общее число операндов в программе (переменные).

Особенностей и сложностей в установке программного обеспечения, благодаря кроссплатформенности языка разработки.

## Разработка и описание пользовательского интерфейса.

Система предназначена для автоматизации работы менеджера при распределении заданий исполнителям с применением онтологии.

В соответствии с требованиями разработан многооконный интерфейс на языке python. Основные функции в интерфейсе:

* Создание задачи с параметрами;
* Распределение созданной задачи по исполнителям.

## Описание используемых компонент системы.

Как уже описывалось онтология спроектирована в редакторе Protégé (рис.4.1).

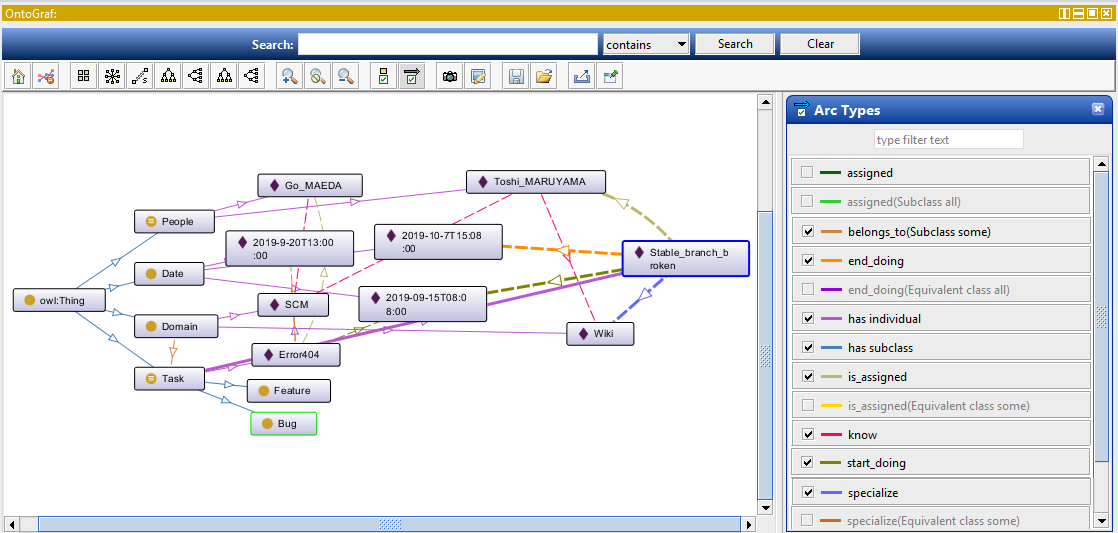


Рис.4.1. Онтология задач с указанием связей классов и индивидуумов

Онтология содержит классы с ограничениями кардинальности, что шире понятий домена и ранга (рис.4.2). Класс Задач (Task) имеет связи назначения (is assigned) Человеку (People), специализации (specialized) по Области применения (Domain), дату создания (start\_doing) по Дате (Date).

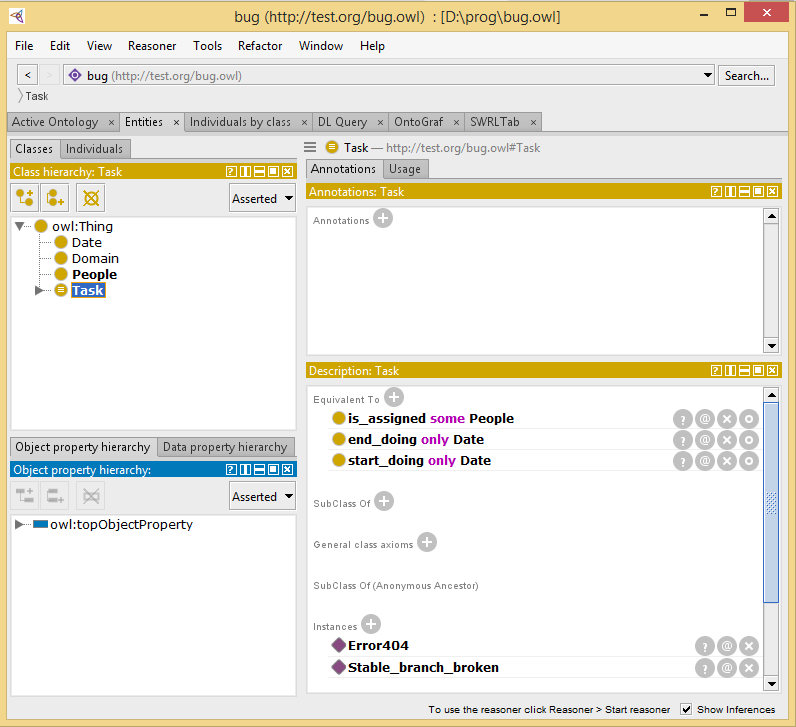


Рис.4.2 Описание класса Задач в онтологии

Библиотека owlready [16] используется для работы с онтологией. Имеется официальное руководство обо всех ее возможностях с примером основных реализаций. Изначально создание и проектирование онтологии осуществлялось программным способом с помощью данной библиотеки. На рис.4.3 отображена загрузка онтологии и вывод классов с их индивидуумами. Для каждой задач также отображены связи соответственно.

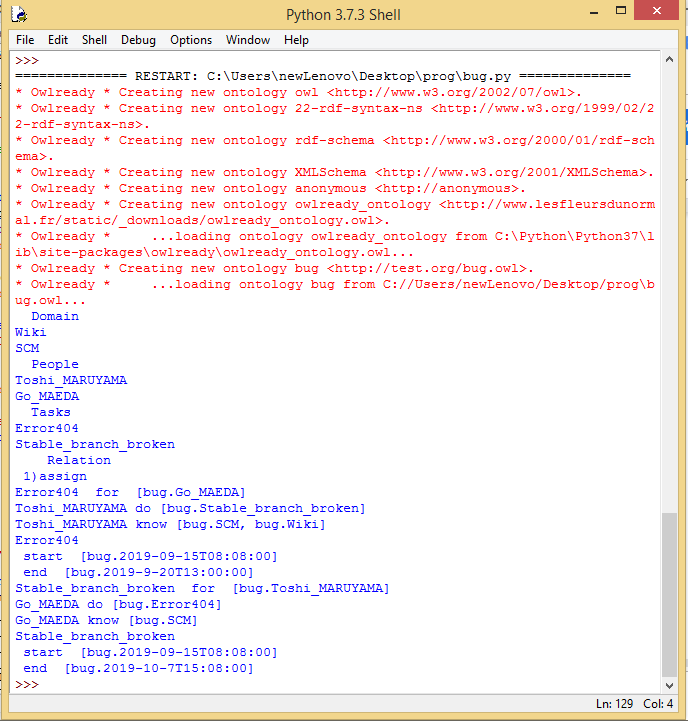


Рис.4.3 Консольное приложение вывода онтологии

Библиотека sys предоставляет доступ к переменным и функциям интерпретатора python. В программе используется для работы с файлами.

Библиотека types интерпретирует стандартные типы Python, а именно для создания нового объекта. А datetime работает с форматом даты и время.

Модуль PyInstaller интерпретирует код программы в исполняемый формат под любую ОС. Следовательно, обеспечивает кросс-платформенность.

# Инструкция пользователю

Для работы программы необходимо запустить файл с соответствующим ОС расширением. Пример на основе Windows для \*.exe файла.

Если у ПК нет доступа к интернету, следует нажать в верхнем меню Import файла онтологии формата owl (рис.5.1). Иначе загружается автоматически по ссылке URI, имеющейся у каждой онтологии.

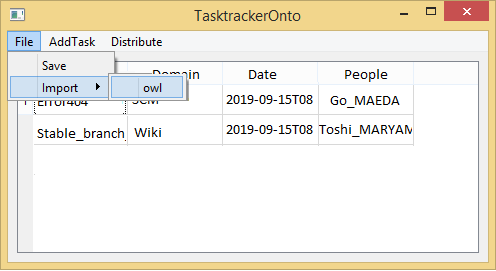


Рис.5.1 Импорт файла онтологии задач

Для добавления новой задачи нужно нажать на соответствующий пункт меню AddTask. Появится новое поле в таблице с возможностью редактирования. Как только значение установлено, активируется запрет на дальнейшие изменения.

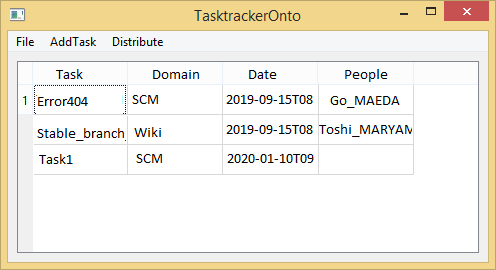


Рис.5.2 Результат работы операции AddTask

Выполнить основную задачу программы и проекта, а именно распределение новой задачи по исполнителям, возможно по нажатию пункта Distribute. Если у каких-то задач нет исполнителя, то можно выбрать данную задачу и повторить предыдущее действие. Результатом данной операции будет обновление таблицы с назначенным исполнителем (рис.5.3).

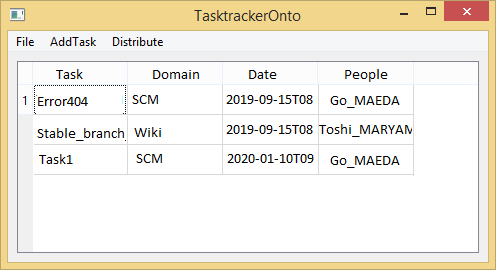


Рис.5.3 Результат работы операции Distribute

Новые данные важно сохранить через пункт меню File > Save (рис.5.1), чтобы при следующей загрузке онтология сохранила новые значения и связи.

# Заключение

Разработанная система выполняет цель: уменьшить трудоемкость системы управления задачами для сотрудников IT-компании. А именно система сопоставляет активные задачи исполнителям с применением онтологии и прецедентного подхода.

В результате выполнения курсового проекта изучены материалы по онтологии в бизнес-процессах по управлению задачами, рассмотрены аналоги решения распределения задач как с применением онтологии, так и другими инструментами интеллектуального анализа. Выбранная среда разработки python IDLE и множество поддерживаемых библиотек, ориентированных на интеллектуальный анализ, помогли реализовать desktop-приложение распределения задачи по группам исполнителей.

Дополнительная формализация параметров задачи позволила управлять онтологией. Ограничения онтологии ПрО через связи и кардинальности сузили набор для распределения. Реализованные функции включают в себя простой старт и удобный интерфейс, визуализацию онтологии IT-задач в виде таблицы, отмеченных в аналогах.

Для анализа качества работы распределения достаточно использовать дедукцию, а именно построить небольшую часть системы в онтологии, как это было сделано из ресурсов Redmine, и рассчитывать качество с изменением небольшого числа параметров.

При дальнейшем исследовании планируется экспериментировать методами классификации для выбора оптимального алгоритма. Так же будет использоваться менее формализованный язык, чем OWL. В высоко-формализованном представлении не оказалось смысла. Однако необходим так же повторный выбор инструментов для интеграции онтологии с программой.

# Список используемых источников

1. В.Г.Паршков [Как правильно распределить обязанности между сотрудниками и обеспечить их разумную загрузку?](https://wiseeconomist.ru/poleznoe/7175-pravilno-raspredelit-obyazannosti-mezhdu-sotrudnikami-obespechit-razumnuyu)//Паршков В.Г. - "Руководитель автономного учреждения". – 2012. – NN7, 8, 9

Эта и другие темы, касающиеся разработки и оптимизации организационной структуры, легли в основу проекта автора "Скорая помощь для вашего бизнеса OnLine" ([www.pplus.ru](http://www.pplus.ru)).

1. Дмитрий Клешевников. Мы обозрели: выбираем систему управления задачами для web-студий//HABR. – 18.01.2016 [Электронный доступ] <https://m.habr.com/ru/company/ptysh/blog/297824/>

SEO-специалисты одной web-студии протестировали множество систем управления проектами и задачами (таск-менеджеры), чтобы выбрать хорошие идеи для собственной разработки.

1. 43 полезных сервиса для управления проектами. Без эпитетов. //HABR, 9.02.2016 [Электронный доступ] https://m.habr.com/ru/post/276873/

Краткое описание сервисов для управления проектами

1. Мошкин В.С. Онтологический подход к анализу временных рядов / Н.Г. Ярушкина, В.С.Мошкин// Интегрированные модели и мягкие вычисления в искусственном интеллекте. Сб. научных трудов VII-й Международной научно-практической конференции. – Коломна – 2013. – № 2. – С. 529–537
2. Мошкин В.С. Онтология как основа семантического анализа данных в задачах автоматизированного проектирования / Н.Г. Ярушкина, В.С.Мошкин// Нечеткие системы и мягкие вычисления (НСМВ-2014): труды Шестой всероссийской научно – практической конференции. – Т.1. –СПб. : Политехникасервис, 2014. – С. 174-182.
3. Карпов Л.Е. Методы добычи данных при построении локальной метрики в системах вывода по прецедентам /Л. Е. Карпов, В. Н. Юдин//Препринты Института системного программирования РАН, Препринт 18, 2006 г. <http://citforum.ru/consulting/BI/data_mining/>
4. Малахова А.И., Черняховская Л.Р., Старцева Е. Б., Владимирова И. П. Правила принятия решений на основе онтологии задач с помощью правил SWRL <https://cyberleninka.ru/article/n/upravlenie-prinyatiem-resheniy-v-organizatsionnom-upravlenii-s-primeneniem-pravil/viewer>
5. Никулина Н. О., Черняховская Л. Р. Модели информационной поддержки принятия решений с использованием систем управления электронным документооборотом: учеб. пособие. Уфа: УГАТУ, 2009. 134с.
6. Александр Дмитриев. Автоматическое назначение задач в Jira с помощью ML //HABR, 24.06.2019 [Электронный доступ] <https://m.habr.com/ru/company/hh/blog/457418/>
7. Создание семантического веб-приложения. – Учебный проект. – 2011 [Электронный доступ] <https://habr.com/ru/post/123612/>
8. Tiller: тайм-менеджмент оффлайн. //Блог о гаджетах и технологиях / Гаджеты / InspectorGadgets.ru, 17.04.2018 [Электронный доступ] <https://www.inspectorgadgets.ru/post/tiller-time-tracker-kickstarter>
9. Устройство трекер Tiller для учета времени, потраченного на каждую задачу.
10. Загорулько Ю.А., Онтологии и их практическое применение в системах, основанных на знаниях/ Ю.А. Загорулько, Г.Б. Загорулько// Цифровой репозиторий DSpace ИВТ СО РАН

1. [Лапшин](https://rsdn.org/article/philosophy/what-is-onto.xml) В.А., Роль онтологий в современной компьютерной науке / В.А. Лапшин. – RSDN Magazine #4. – 2009
2. Онтология для управления проектами: обследование. / International Journal of Information Processing and Management. – ноябрь 2014 [Электронный доступ] <https://www.researchgate.net/publication/270281571_Ontologies_for_Project_Management_Survey/download>
3. ВласовД.Ю. Автоматизация извлечения отношений между понятиями из текстов естественного языка /Д.Ю. Власов,Д.Е. Пальчунов, П.А. Степанов // Вестник Новосибирского государственного университета. Серия: Информационные технологии. – 2010 [Электронный доступ] https://cyberleninka.ru/article/n/avtomatizatsiya-izvlecheniya-otnosheniy-mezhdu-ponyatiyami-iz-tekstov-estestvennogo-yazyka
4. ЛеХоай. Семантическое аннотирование документов в электронных библиотеках/ ЛеХоай, А.Ф. Тузовский // Известия Томского политехнического университета. Инжиниринг георесурсов. – 2013 [Электронный доступ] https://cyberleninka.ru/article/n/semanticheskoe-annotirovanie-dokumentov-v-elektronnyh-bibliotekah
5. Клименко С.В., Золотарев О.В., Шарнин М.М. Использование онтологического подхода для анализа текстов естественного языка / С.В. Клименко, О.В. Золотарев, М.М. Шарнин // Вестник Российского нового университета. -2017. -№1. – С. 67-71
6. Святогор Л., Гладун В. Семантический анализ текстов естественного языка: цели и средства. – 2009. - Knowledge-Dialogue-Solution
7. Jean-Philippe Lang. Сайт Redmine. Developer’s – 2006-2014 [Электронный доступ] <https://www.redmine.org/projects/redmine/issues/report>
8. Тушканова О.Н., Городецкий В.И. Ассоциативная классификация: аналитический обзор. Часть 1 // Труды СПИИРАН. 2015. № 38 (1). C. 183-203. [Электронный доступ] https://doi.org/10.15622/sp.38.10

# Приложение 1

## Листинг программы

**from** owlready **import** \*  
**import** types  
*#------------------pycharm*onto\_path.append(**"C://Users/newLenovo/Desktop/prog"**)  
*#bug\_onto = Ontology("http://test.org/bug.owl")*bug\_onto = get\_ontology(**"http://test.org/bug.owl"**).load()  
  
*#print("have type %s %inst\_Bug.do())  
#----------Create classes---------*myPeople = types.new\_class(**"People"**, (Thing,), kwds = { **"ontology"** : bug\_onto })  
myDomain = types.new\_class(**"Domain"**, (Thing,), kwds = { **"ontology"** : bug\_onto })  
myDate = types.new\_class(**"Data"**, (Thing,), kwds = { **"ontology"** : bug\_onto })  
**class** Task(Thing):  
 ontology = bug\_onto  
  
  
*#sync.reasoner - ч/з протеже - для проверки противоречивости  
#----------Create Properties-------***class** is\_assigned(Property):  
 ontology = bug\_onto  
   
**class** assigned(Property):  
 ontology = bug\_onto  
 domain = [myPeople]  
 range = [Task]  
 inverse\_property = is\_assigned  
   
**class** know(Property):  
 ontology = bug\_onto  
 domain = [myPeople]  
 range = [myDomain]  
   
**class** start\_doing(Property):  
 ontology = bug\_onto  
 domain = [myData]  
 range = [Task]  
  
 *#MIN, 2, ActivePrinciple  
#----------Create Instances ---------*inst\_domain = []  
inst\_people = []  
inst\_task = []  
print (**" Domain"**)  
**for** i **in** bug\_onto.Domain.instances():  
 print(i)  
 inst\_domain.append(i)  
print (**" People"**)  
**for** i **in** bug\_onto.People.instances():  
 print(i)  
 inst\_people.append(i)  
print (**" Tasks"**)  
**for** i **in** bug\_onto.Task.instances():  
 print(i)  
 inst\_task.append(i)  
  
print(**" Relation \n 1)assign"**)  
  
**for** i **in** range(2):*#len(inst\_task)):  
 #inst\_task[i].is\_assigned.append(inst\_people[i])* print(inst\_task[i],**" for "**,inst\_task[i].is\_assigned)  
 print(inst\_people[i],**"do"**,inst\_people[i].assigned)  
 print(inst\_people[i],**"know"**,inst\_people[i].know)  
 print(inst\_task[i],**"\n start "**,inst\_task[i].start\_doing,  
 **"\n end "**,inst\_task[i].end\_doing)  
  
bug\_onto.save()  
*#get an OWL string for a given ontology*print(to\_owl(bug\_onto))

**from** PyQt5.QtCore **import** \**#QSize, Qt***from** PyQt5.QtWidgets **import** QApplication, QMainWindow, QGridLayout, QWidget, QTableWidgetItem, QMenu, QAction  
   
**import** sys  
**from** widgets **import** CustomTableWidget  
   
   
*# Наследование от QMainWindow***class** MainWindow(QMainWindow):  
 *# Overriding the class constructor* **def** \_\_init\_\_(self):  
 *# Обязательно вызовите метод суперкласса* QMainWindow.\_\_init\_\_(self)  
  
 self.myMenuBar()  
   
 central\_widget = QWidget(self) *# Создать центральный виджет* self.setCentralWidget(central\_widget) *# Установить центральный виджет* grid\_layout = QGridLayout() *# Создать QGridLayout* central\_widget.setLayout(grid\_layout) *# Установите это размещение в центральном виджете* table = CustomTableWidget(self) *# Создать таблицу* table.setColumnCount(5) *# Мы устанавливаем три колонки  
 #table.setRowCount(1) # и один ряд в таблице  
   
 # Установить заголовки таблицы* table.setHorizontalHeaderLabels([**"ID"**,**"Task"**, **"Domain"**, **"Time\_start"**, **"Executor"**])  
 **'''  
 # Установить подсказки для заголовков  
 table.horizontalHeaderItem(0).setToolTip("Column 1 ")  
 table.horizontalHeaderItem(1).setToolTip("Column 2 ")  
 table.horizontalHeaderItem(2).setToolTip("Column 3 ")  
 '''** *# Установите выравнивание к заголовкам* **for** i **in** range(5):  
 table.horizontalHeaderItem(i).setTextAlignment(Qt.AlignLeft)  
 *#table.horizontalHeaderItem(1).setTextAlignment(Qt.AlignHCenter)  
 #table.horizontalHeaderItem(2).setTextAlignment(Qt.AlignRight)  
   
 # заполните первую строку* **for** i **in** range(5):  
 row\_position = self.table.rowCount()  
 table.insertRow(row\_position)  
 table.setItem(0, 0, self.createItem(**"Text in column 1"**, Qt.ItemIsSelectable | Qt.ItemIsEnabled))  
 table.setItem(0, 1, self.createItem(**"Text in column 2"**, Qt.ItemIsSelectable | Qt.ItemIsEnabled | Qt.ItemIsEditable))  
 table.setItem(0, 2, self.createItem(**"Text in column 3"**, Qt.ItemIsSelectable | Qt.ItemIsEnabled | Qt.ItemIsEditable))  
   
 *# изменить размер столбца по содержимому* table.resizeColumnsToContents()  
   
 grid\_layout.addWidget(table, 0, 0) *# Add a table to the grid* **def** createItem(self, text, flags):  
 tableWidgetItem = QTableWidgetItem(text)  
 tableWidgetItem.setFlags(flags)  
 **return** tableWidgetItem  
   
   
 **def** myMenuBar(self):   
   
 menubar = self.menuBar()  
 fileMenu = menubar.addMenu(**'File'**)  
   
 impMenu = QMenu(**'Import'**, self)  
 impAct = QAction(**'owl'**, self)   
 impMenu.addAction(impAct)  
   
 newAct = QAction(**'Save'**, self)   
   
 fileMenu.addAction(newAct)  
 fileMenu.addMenu(impMenu)  
   
 taskMenu = menubar.addMenu(**'AddTask'**)  
 doMenu = menubar.addMenu(**'Distribute'**)  
 *#<->, v^, width, length  
 #self.setGeometry(500, 200, 500, 300)* self.setWindowTitle(**'TasktrackerOnto'**)   
 *#self.show()* self.setMinimumSize(QSize(480, 80)) *# Установить размеры  
 #self.setWindowTitle("Работа с QTableWidget") # Установите заголовок окна* taskMenu.triggered.connect(self.actionClicked)  
  
 *#@QtCore.pyqtSlot()* **def** actionClicked(self):  
 action = self.sender()  
 print(**'Action: '**, action.text())  
 *#self.statusBar().showMessage(sender.text() + ' was pressed')***if** \_\_name\_\_ == **"\_\_main\_\_"**:  
   
 app = QApplication(sys.argv)  
 mw = MainWindow()  
 mw.show()  
 sys.exit(app.exec())

**from** PyQt5.QtCore **import** Qt  
**from** PyQt5.QtGui **import** QKeyEvent  
**from** PyQt5.QtWidgets **import** QTableWidget  
   
   
*# Мы создаем собственный QTableWidget для реагирования на события нажатия клавиш***class** CustomTableWidget(QTableWidget):  
   
 *# переопределяем keyPressEvent* **def** keyPressEvent(self, e: QKeyEvent) -> **None**:  
 **if** e.key() == Qt.Key\_Enter:  
 print(**"Key enter was pressed"**)  
 **elif** e.key() == Qt.Key\_Return:  
 print(**"Key return was pressed"**)

1. "Бритва Оккама". (Проще = лучше) Более простым теориям следует отдавать предпочтение перед сложными, если и те и другие в равной степени согласуются с эмпирическими, опытными данными. [↑](#footnote-ref-1)
2. принцип Родена: отсекать все лишнее. [↑](#footnote-ref-2)
3. Принцип наименьшего усилия - небольшая доля причин, вкладываемых средств или прилагаемых усилий отвечает за большую долю результатов, получаемой продукции или заработанного вознаграждения.

   * закон фокуса: меньше - значит больше;
   * закон прогресса: можно создавать большее посредством меньшего.

   [↑](#footnote-ref-3)
4. Актёр – элемент, обозначающий роли пользователя, который взаимодействует с определенной сущностью [↑](#footnote-ref-4)