МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ»

Кафедра автоматизированных систем управления (АСУ)

В.Г. Резник

СИСТЕМНЫЙ АНАЛИЗ, УПРАВЛЕНИЕ И ОБРАБОТКА ИНФОРМАЦИИ: ЧАСТЬ II

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ РАБОТ
И ОРГАНИЗАЦИИ САМОСТОЯТЕЛЬНОЙ РАБОТЫ
для аспирантов направления подготовки
09.06.01 – Информатика и вычислительная техника:
направленность (профиль) 05.13.01 – Системный анализ, управление и обработка информации (информация и информационные системы, экономика, энергетика, промышленность, образование)

Томск - 2018

Резник В.Г.

Системный анализ, управление и обработка информации. Часть II: Методические указания по практическим занятиям аспирантов по направлению «09.06.01 - Информатика и вычислительная техника» (профиль 05.13.01 «Системный анализ, управление и обработка информации (информация и информационные системы, экономика, энергетика, промышленность, образование»). – Томск, ТУСУР, 2018. – 146 с.

Методические указания определяют порядок выполнения практических работ и организацию самостоятельной работы аспирантов по дисциплине «Системный анализ, управление и обработка информации» для направленности (профиля) 05.13.01 – Системный анализ, управление и обработка информации (информация и информационные системы, экономика, энергетика, промышленность, образование) подготовки кадров высшей квалификации, осуществляемой в рамках направления подготовки 09.06.01 – Информатика и вычислительная техника. Практикум по части II этой дисциплины является продолжением практикума по части I. В части II изучаются компьютерные технологии обработки информации. Методические указания определяют также регламент самостоятельной работы при подготовке к выполнению практических работ. Для оценки степени освоения закрепленной за дисциплиной компетенции предложены тестовые задания.

Рассмотрено и утверждено на заседании кафедры автоматизированных систем управления (АСУ) факультета систем управления ТУСУР 26 апреля 2018 года, протокол № 5.

Оглавление

Введение

Данное учебно-методическое пособие предназначено для подготовки и выполнения аспирантами части II практических работ по дисциплине «Системный анализ, управление и обработка информации». Тематика этой части практикума – ***компьютерные технологии обработки информации***. Предполагается, что в предыдущем семестре аспирант прошел обучение по следующим трем разделам дисциплины: основные понятия и задачи системного анализа; модели и методы принятия решений и основы теории управления [1]. Также считается, что аспирант уже изучил дисциплину «Современные компьютерные технологии» [2-5].

Практические работы направлены на формирование у аспирантов следующей компетенции: способность разрабатывать, развивать и конкретизировать теоретические основы и методы системного анализа, управления и обработки информации в следующих областях профессиональной деятельности: информация и информационные системы, экономика, энергетика, промышленность, образование (ПК-3).

Практические работы выполняются индивидуально. Каждой работе предшествует подготовка к практическим занятиям и семинарам, а также самостоятельная работа, которая предназначена для оформления единого отчета по выполненной работе.

Основная рекомендуемая литература по каждой практической работе содержится в конце данного пособия или приведена в соответствующих методических указаниях. Дополнительно, работы, опирающиеся на конкренные реализации алгоритмов обработки информации, содержат локальный список использованных источников. Для сбора информации можно использовать опрос экспертов и других лиц, причастных к деятельности исследуемой системы; наблюдения, непосредственное участие в деятельности исследуемой системы; поиск информации в статистических сборниках, в литературных источниках и Интернете.

Каждая практическая работа выполняется в соответствии с методическими указаниями. Вначале работы аспирант выбирает вариант задания. По окончанию работы составляется отчет. Примеры заданий, порядок выполнения работы и содержание отчета описаны в методических указаниях.

Форма контроля выполнения практической работы – защита отчета. Для оценки степени освоения компетенции используются оценочные тестовые задания.

1 Информационные технологии

Первые три практических занятия посвящены информационным технологиям, в которых рассматриваются:

- композиция и декомпозиция данных в информационных технологиях;
- обработка данных в офисных технологиях;
- обработка данных в СУБД.

1.1 Практическая работа №1 «Методы композиции и декомпозиции в информационных технологиях»

Цель работы: получить практические навыки в компьютерной технологии обработки данных средствами средствами технологии ***zip***.

1.1.1 Самостоятельная работа

Определение и общая классификация видов информационных технологий. Модели, методы и средства сбора, хранения, коммуникации и обработки информации с использованием компьютеров.

Литература: [6].

1.1.2 Порядок выполнения работы

Методы композиции и декомпозиции различных объектов данных играют важнейшую роль в инфориационных технологиях. Например, в языках программирования широко используются структуры данных, объектные модули программ объединяются в различные библиотеки и как далее.

В данной практической работе рассматривается базовые основы технологии *zip*, которая наряду с ее широким применением мало отражена в учебной литературе.

1.1.2.1 История вопроса

В январе 1979 года был опубликован формат архиватора для ОС UNIX. Как отмечено в [1.1.1]: «**tar** (англ. *tape archive*) — формат битового потока или файла архива, а также название традиционной для Unix программы для работы с такими архивами. Программа tar была стандартизирована в POSIX.1-1998, а также позднее в POSIX.1-2001. Первоначально программа tar использовалась для создания

архивов на магнитной ленте, а в настоящее время tar используется для хранения нескольких файлов внутри одного файла, для распространения программного обеспечения, а также по прямому назначению — для создания архива файловой системы. Одним из преимуществ формата tar при создании архивов является то, что в архив записывается информация о структуре каталогов, о владельце и группе отдельных файлов, а также временны́е метки файлов.».

В 1989 году Фил Кац, основатель компании PKWARE, опубликовал формат архиватора ZIP [1.1.2]: «**ZIP** — популярный формат архивации файлов и сжатия данных без потерь. Архив ZIP может содержать один или несколько файлов и каталогов, которые могут быть сжаты разными алгоритмами. Наиболее часто в ZIP используется алгоритм сжатия Deflate. Формат был создан в 1989 году Филом Кацем и реализован в программе PKZIP компании PKWARE в качестве замены формату архивов ARC Тома Хендерсона. Формат ZIP поддерживается множеством программ, в том числе операционными системами Microsoft Windows (с 1998 года) и Apple Mac OS X (с версии 10.3). Многие свободные операционные системы также имеют встроенную поддержку ZIP-архивов.».

Широкую популярность технология zip получила с появлением в мае 1995 года языка Java. Корпорация Sun Microsystems в своем архиваторе Jar продемонстрировала широкие прикладные возможности этой технологии, создав достаточно универсальную инфораструктуру для хранения классов языка.

1.1.2.2 Структура zip-файла

В настоящее время, спецификации на формат zip-архивов являются общественным достоянием. Их можно найти в документе [1.1.3]. Сама архитектура zip-файла достаточно проста и может обрабатываться потоковыми алгоритмами чтения и анализа информации: «...

```
4.3.6 Overall .ZIP file format:

      [local file header 1]
      [encryption header 1]
      [file data 1]
      [data descriptor 1]
      .
      .
      .
      [local file header n]
      [encryption header n]
      [file data n]
      [data descriptor n]
      [archive decryption header]
      [archive extra data record]
      [central directory header 1]
      .
      .
```

```
.
[central directory header n]
[digital signature]
[zip64 end of central directory record]
[zip64 end of central directory locator]
[end of central directory record]
```

...».

Хорошо видно, что архитектура zip-файла имеет «плоскую» структуру, которая состоит из последовательности различных типов заголовков (header) и расположенных между ними сжатых (или не сжатых) данных.

Каждый заголовок (header) начинается с 4-х байтной сигнатуры, младшие два байта которой равны 0x4b50 (или «PK» - как строка). В таблице 1.1 представлены сигнатуры всех заголовков.

Таблица 1.1

Синатура заголвока структуры	***Имя структуры***
0x04034b50	`[local file header x]`
0x08074b50	`[data descriptor x]`
0x08064b50	`[archive extra data record]`
0x02014b50	`[central directory header x]`
0x05054b50	`[digital signature]`
0x06064b50	`[zip64 end of central directory record]`
0x07064b50	`[zip64 end of central directory locator]`
0x06054b50	`[end of central directory record]`

Таким образом, зная структуры заголовков, можно прочитать весь zip-архив и извлечь из него данные. В общем случае для этих целей используются специальные утилиты или отдельные графические приложения, но в частных случаях приложения сами могут работать с такими архивами. Покажем это на примере анализа общей структуры zip-архива.

1.1.2.3 Тестовая программа анализа zip-архива

Для анализа общей структуры zip-архива достаточно знание структур `[central directory header x]` и `[end of central directory record]`, которые качественно описаны в [1.1.3]. Но поскольку они не представлены в синтаксисе языка С, то воспользуемся готовыми обозначениями, приведенными в статье [1.1.4]. Дополнительно рекомендуется изучит также источники [1.1.5, 1.1.6].

Исходный текст программы, анализирующей zip-архив, представлен в листинге 1.1. Там же приводятся все необходимые обозначения и пояснения работы алгоритма.

Листинг 1.1 — Программа анализа zip-архива

```
/*
 ============================================================================
 Name        : zip_check.c
 Author      : Reznik V.G., 01.11.2018
 Version     :
 Copyright   : Your copyright notice
 Description : Анализ zip-файла, Ansi-style
 ============================================================================

 Программа zip_check.c написана специально для дисциплины "Системный анализ,
 управление и обработка информации. Часть II", преподаваемой на кафедре АСУ
 ТУСУР для аспирантов направления 09.06.01

 Назначение программы: анализ структуры zip-файла.

 Поскольку данная программа не анализирует содержимое архивированных (сжатых)
 файлов, то алгоритм ее работы - достаточно прост:

 1) сначала считывается статическая часть структуры
      [end of central directory record] (структура EOCD), расположенная в конце
      файла, из которой извлекается количество записей (структуры [central
      directory header x] - структура CentralDirectoryFileHeader) и смещение на
      первую из них (от начала zip-файла);

 2) в цикле счиываются структуры типа [central directory header x] и печатаются
      основные характеристики архивированного файла, включая смещение на структуру
      типа [local file header x] (структура LocalFileHeader), а также - имя файла;

 3) дополнительно, в цикле 2) печатаются данные структуры [local file header x],
      включая имя файла.
 */
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#include <stdint.h>

#include <sys/types.h>
```

```
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/mman.h>
#include <string.h>

/**
 * Описание статической части структуры [end of central directory record]
 * в zip-файле, за данными этой структуры расположен массив переменной длины:
 * (см. далее, после описания структуры)
 */
struct EOCD
{
    // Обязательная сигнатура, равна 0x06054b50
    uint32_t signature;
    // Номер диска
    uint16_t diskNumber;
    // Номер диска, где находится начало структуры
    // [end of central directory record]
    uint16_t startDiskNumber;
    // Количество записей структур типа
    // [central directory header x] на текущем диске
    uint16_t numberCentralDirectoryRecord;
    // Всего записей структур типа [central directory header x]
    uint16_t totalCentralDirectoryRecord;
    // Размер структуры [end of central directory record]
    uint32_t sizeOfCentralDirectory;
    // Смещение на первую структуру [central directory header 1]
    uint32_t centralDirectoryOffset;
    // Длина комментария, следующего за этой структурой.
    uint16_t commentLength;

    //Указываем, что выравнивание полей - недопустимо!!!
} __attribute__((packed));    //Для Visual Studio -
                              //читайте про #pragma push(pack/pop)

/**
 * В zip-файле указанные поля идут сразу за данными
 * структуры: EOCD
    // Комментарий (длиной commentLength)
    uint8_t *comment;
*/

/**
 * Описание статической части структуры [central directory header x]
 * в zip-файле, за данными этой структуры расположены массивы переменной длины:
 * (см. далее, после описания структуры)
 */
struct CentralDirectoryFileHeader
{
    // Обязательная сигнатура, равна 0x02014b50
    uint32_t signature;
```

```
    // Версия для создания
    uint16_t versionMadeBy;
    // Минимальная версия для распаковки
    uint16_t versionToExtract;
    // Битовый флаг
    uint16_t generalPurposeBitFlag;
    // Метод сжатия (0 - без сжатия, 8 - deflate)
    uint16_t compressionMethod;
    // Время модификации файла
    uint16_t modificationTime;
    // Дата модификации файла
    uint16_t modificationDate;
    // Контрольная сумма
    uint32_t crc32;
    // Сжатый размер
    uint32_t compressedSize;
    // Несжатый размер
    uint32_t uncompressedSize;
    // Длина название файла
    uint16_t filenameLength;
    // Длина поля с дополнительными данными
    uint16_t extraFieldLength;
    // Длина комментариев к файлу
    uint16_t fileCommentLength;
    // Номер диска
    uint16_t diskNumber;
    // Внутренние аттрибуты файла
    uint16_t internalFileAttributes;
    // Внешние аттрибуты файла
    uint32_t externalFileAttributes;
    // Смещение от начала zip-файла до
    // соответствующей структуры [local file header x]
    uint32_t localFileHeaderOffset;

    //Указываем, что выравнивание полей - недопустимо!!!
} __attribute__((packed));    //Для Visual Studio -
                              //читайте про #pragma push(pack/pop)

/**
 * В zip-файле указанные поля идут сразу за данными
 * структуры: CentralDirectoryFileHeader

    // Имя файла (длиной filenameLength)
    uint8_t *filename;

    // Дополнительные данные (длиной extraFieldLength)
    uint8_t *extraField;

    // Комментарий к файла (длиной fileCommentLength)
    uint8_t *fileComment;
 */
```

```
/**
 * Описание статической части структуры [local file header x]
 * в zip-файле, за данными этой структуры расположены массивы переменной длины:
 * (см. далее, после описания структуры)
 */
struct LocalFileHeader
{
    // Обязательная сигнатура, равна 0x04034b50
    uint32_t signature;
    // Минимальная версия для распаковки
    uint16_t versionToExtract;
    // Битовый флаг
    uint16_t generalPurposeBitFlag;
    // Метод сжатия (0 - без сжатия, 8 - deflate)
    uint16_t compressionMethod;
    // Время модификации файла
    uint16_t modificationTime;
    // Дата модификации файла
    uint16_t modificationDate;
    // Контрольная сумма
    uint32_t crc32;
    // Сжатый размер
    uint32_t compressedSize;
    // Несжатый размер
    uint32_t uncompressedSize;
    // Длина название файла
    uint16_t filenameLength;
    // Длина поля с дополнительными данными
    uint16_t extraFieldLength;

    //Указываем, что выравнивание полей - недопустимо!!!
} __attribute__((packed));    //Для Visual Studio -
                              //читайте про #pragma push(pack/pop)

/**
 * В zip-файле указанные поля идут сразу за данными
 * структуры: LocalFileHeader

    // Название файла (размером filenameLength)
    uint8_t *filename;

    // Дополнительные данные (размером extraFieldLength)
    uint8_t *extraField;
*/

/**
 * Головная программа, реализующая алгоритм анализа
 * zip-файла.
 */
int main(int argc, char *argv[]) {

      printf("=== start: %s ===\n", argv[0]);
```

```
/**
 * Проверяем наличие аргумента, который должен
 * содержать полный путь к файлу.
 */
assert(argc == 2);
printf("%s %s\n", argv[0], argv[1]);

/**
 * Открываю zip-файл только на чтение.
 */
int fd;
fd = open (argv[1], O_RDONLY);
if(fd < 0) {
      printf("Не могу открыть файл: %s\n", argv[1]);
      return -1;
}

/**
 * Определяю длину файла
 */
struct stat st;
if(stat(argv[1], &st)< 0) {
      printf("Не могу определить длину файла: %s\n", argv[1]);
      return -1;
}
size_t len = (size_t)st.st_size;
printf("Длина файла %s: %li байт\n\n", argv[1], len);

/**
 * Делаю mmap, Чтобы работать с файлом как с массивом байт.
 */
uint8_t *u8;
u8 = (uint8_t *)mmap(0, st.st_size, PROT_READ, MAP_PRIVATE, fd, 0);

if(u8 <= 0) {
      printf("Не могу сделать mmap файла: %s\n", argv[1]);
      close(fd);
      return -1;
}

/**
 * Часть 1.
 * Поиск начала структуры [end of central directory record] (структура EOCD)
 * по ее сигнатуре
 */
size_t il = 0;
uint32_t signature;
struct EOCD eocd;
uint8_t * p;
p = (uint8_t *)&eocd;
```

```
  for (size_t offset = len - sizeof(eocd); offset != 0; --offset)
  {
      // Считываем четыре байта сигнатуры
      strncpy((char *)&signature,(char *)&u8[offset], sizeof(signature));

      if (0x06054b50 == signature)
      {
          il = offset;
          printf("Адрес  структуры [end of central directory record]: %li\n",
                    il);
          break;
      }
  }
  if(il == 0){
        puts("Не найдено начало структуры "
             "[end of central directory record]!!!!");
        munmap(u8, len);
        close(fd);
        return -1;
  }
printf("Размер структуры EOCD: %li\n\n",
        sizeof(eocd));

  /**
   * Копируем содержимое структуры EOCD.
   */
for(int i=0; i<sizeof(eocd); i++)
  p[i] = u8[il + i];

  /**
   * Печатаем значения структуры EOCD.
   */
uint8_t buf[0x10000]; //Буфер для переменной части структур

puts("===== Параметы структуры [end of central directory record] =====");
printf("Сигнатура EOCD: 0x%x\n", eocd.signature);
printf("Номер диска   : %i\n", eocd.diskNumber);
printf("Номер диска для центальной директории: %i\n",
        eocd.startDiskNumber);
printf("Количество записей  [central directory header x]: %i\n",
        eocd.numberCentralDirectoryRecord);
printf("Всего записей       [central directory header x]: %i\n",
        eocd.totalCentralDirectoryRecord);
printf("Размер всех записей [central directory header x]: %i\n",
        eocd.sizeOfCentralDirectory);
printf("Смещение до         [central directory header 1]: %u\n",
        eocd.centralDirectoryOffset);

strncpy((char *)&buf,(char *)&u8[il + sizeof(eocd)],
        eocd.commentLength);
buf[il + sizeof(eocd) + eocd.commentLength] = 0;
```

```
printf("Длина коментария: %u\n", eocd.commentLength);
printf("Коментарий      : %s\n\n", (char *)&buf);

/**
 * Часть 2.
 * Определяем в zip-файле местоположение структуры:
 * [central directory header 1] - (CentralDirectoryFileHeader)
 */
  size_t kl = il;

  struct CentralDirectoryFileHeader cdfh;
  p = (uint8_t *)&cdfh;

  struct LocalFileHeader lfh;
  uint8_t *pp = (uint8_t *)&lfh;

  size_t offset = eocd.centralDirectoryOffset;

  /**
   * Цикл по чтению и печати структур [central directory header x]
   * и [local file header x].
   */
  for ( int n=0; n < eocd.numberCentralDirectoryRecord; n++)
  {

  // Копируем содержимое структуры CentralDirectoryFileHeader
    for(int i=0; i<sizeof(cdfh); i++)
        p[i] = u8[offset + i];

    //Проверяем сигнатуру [central directory header x]
      if (0x02014b50 == cdfh.signature)
      {
          printf("%i) [central directory header %i]: "
                 "%li ================\n",
                 n + 1, n + 1, offset);
          printf("Длина структуры       : %lu\n\n", sizeof(cdfh));
          kl = offset + sizeof(cdfh);
          /**
         * Печатаем значения структуры CentralDirectoryFileHeader
         */
          printf("Сигнатура             : 0x%x\n",
                    cdfh.signature);
          printf("Версия для создания   : %u\n", cdfh.versionMadeBy);
          printf("Версия для распаковки : %u\n", cdfh.versionToExtract);
          printf("Метод сжатия          : %u\n", cdfh.generalPurposeBitFlag);
          printf("Сжатый размер         : %u\n", cdfh.compressedSize);
          printf("Не сжатый размер      : %u\n", cdfh.uncompressedSize);
          printf("Длина названия файла  : %u\n", cdfh.filenameLength);
          printf("Длина extra field     : %u\n", cdfh.extraFieldLength);
          printf("Длина коментария      : %u\n", cdfh.fileCommentLength);
          printf("Смещение до [local file header %i]: %u\n",
                    n + 1, cdfh.localFileHeaderOffset);
```

```
  //Читаем имя файла.
  for(int i=0; i<cdfh.filenameLength; i++){
      buf[i] = u8[kl + i];
  }
  buf[cdfh.filenameLength] = 0;

  //Печатаем имя файла.
  if(cdfh.uncompressedSize == 0)
      printf("Имя директории        : %s\n\n", (char *)&buf[0]);
  else
      printf("Имя файла             : %s\n\n", (char *)&buf[0]);

  offset = kl + cdfh.filenameLength +
           cdfh.extraFieldLength +
           cdfh.fileCommentLength;

  kl = cdfh.localFileHeaderOffset;
  printf("[local file header %i]: %lu\n",
           n + 1, kl);
  printf("Длина структуры       : %lu\n\n", sizeof(lfh));

// Копируем содержимое структуры LocalFileHeader
for(int i=0; i<sizeof(lfh); i++)
      pp[i] = u8[kl + i];
//Проверяем сигнатуру [local file header x]
if (0x04034b50 == lfh.signature)
{
      /**
       * Печатаем значения структуры [local file header x]
       */
      printf("Сигнатура             : 0x%x\n",
                  lfh.signature);
      printf("Версия для распаковки : %u\n", lfh.versionToExtract);
      printf("Метод сжатия          : %u\n", lfh.generalPurposeBitFlag);
      printf("Сжатый размер         : %u\n", lfh.compressedSize);
      printf("Не сжатый размер      : %u\n", lfh.uncompressedSize);
      printf("Длина имени файла     : %u\n", lfh.filenameLength);
      printf("Длина extra field     : %u\n", lfh.extraFieldLength);

      //Читаем имя файла.
      kl += sizeof(lfh);
      for(int i=0; i<lfh.filenameLength; i++){
            buf[i] = u8[kl + i];
      }
      buf[lfh.filenameLength] = 0;

      //Печатаем имя файла.
      printf("Имя файла             : %s\n", (char *)&buf[0]);
      puts(" ");
 }else
      puts("Ошибочное значение сигнатуры LocalFileHeader !!!");
```

```
            }else{
              puts("Ошибочное значение сигнатуры CentralDirectoryFileHeader !!!");
              break;
            }
        }
        /**
         * Завершение работы программы
         */
        printf("=== exit: %s ===\n", argv[0]);
        munmap(u8, len);
        close(fd);

        return EXIT_SUCCESS;
}
```

Замечание

Программа *zip_check* использует только стандартные заголовочные файлы языка C, поэтому может быть компилирована на любой платформе ЭВМ. Дополнительно нужно лишь учесть, что должно быть отменено выравнивание полей в используемых структурах данных.

Как отмечено в [1.1.7]: Всемирная организация по стандартизации (ISO) предложила следующие ограничения на использование ZIP-формата, как контейнера файлов:

- Файлы в ZIP-архивах могут быть сохранены несжатыми или с использованием сжатия «смятия» (то есть метод сжатия может содержать значение «0» - просто сохранениее или «8» - сжатие по алгоритму *deflated*).
- Функции шифрования запрещены.
- Функции цифровой подписи запрещены.
- Функции «исправленных данных» запрещены.
- Архивы могут не охватывать несколько томов или быть сегментированными.

1.1.2.4 Анализ структуры архива

Аспиранту следует самостоятельно компилировать программу, приведенную на листинге 1.1 и применить ее к конкретным zip-файлам.

В качестве примера, в редакторе LibreOffice Writer создан файл с именем *test5.odt*, в котором содержится всего лишь одно слово «Тест5.odt».

При условии, что в конкретной директории содержатся файл *test5.odt* и программа *zip_check*, выполнена команда:

```
./zip_check test5.odt > ./test5.txt
```

В результате работы программы *zip_check*, создан файл, начальное содержимое которого показано на рисунке 1.1.

Аспиранту следует:

- выбрать произвольный файл формата *docx*;
- провести обработку его программной *zip_check*;
- провести анализ полученного результата;
- результаты анализа описать в личном отчете.

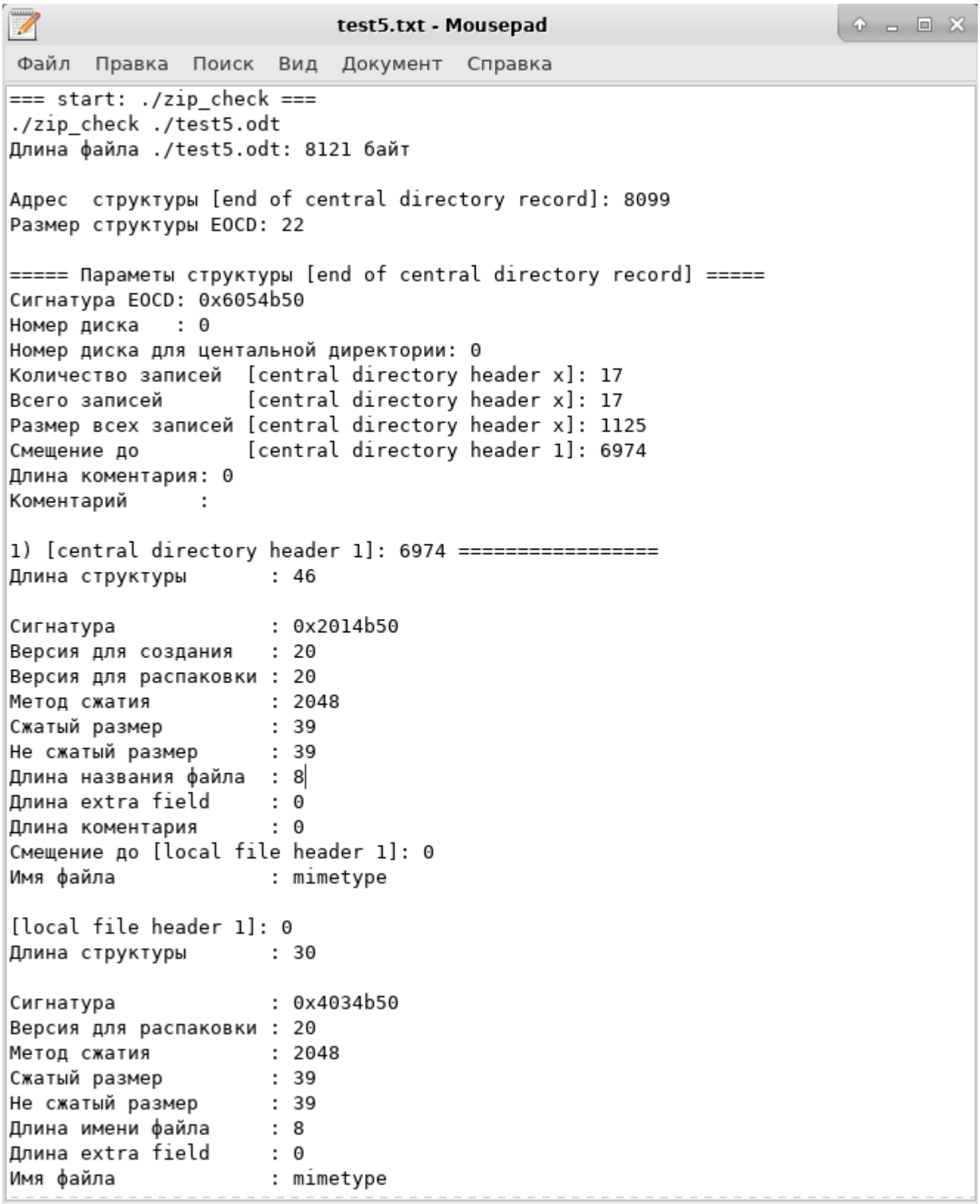

Рисунок 1.1 - Пример анализа структуры файла формата odt

1.1.3 Список использованных источников

1.1.1 tar - Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/Tar). Проверено: 30.10.2018.

1.1.2 ZIP - Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/ZIP). Проверено: 30.10.2018.

1.1.3 .ZIP File Format Specification: [Электронный документ]. - (https://pkware.cachefly.net/webdocs/casestudies/APPNOTE.TXT). Проверено: 30.10.2018.

1.1.4 Описание формата ZIP файла — Блог Евгения Жирнова: [Электронный документ]. - (https://blog2k.ru/archives/3391). Проверено: 30.10.2018.

1.1.5 Чтение ZIP файла — Блог Евгения Жирнова: [Электронный документ]. - (https://blog2k.ru/archives/3392). Проверено: 30.10.2018.

1.1.6 Запись ZIP файла — Блог Евгения Жирнова: [Электронный документ]. - (https://blog2k.ru/archives/3397). Проверено: 30.10.2018.

1.1.7 Zip (file format) - Wikipedia: [Электронный документ]. - (https://en.wikipedia.org/wiki/Zip_(file_format)). Проверено: 30.10.2018.

1.2 Практическая работа №2 «Офисные технологии»

Цель работы: изучение форматов представления данных в современных офисных приложениях.

1.2.1 Самостоятельная работа

Программно-технические средства реализации современных офисных технологий. Стандарты пользовательских интерфейсов. Создание и обработка текстовых файлов и документов с использованием текстовых редакторов и процессоров. Программные средства создания и обработки электронных таблиц. Программные средства создания графических объектов, графические процессоры (векторная и растровая графика).

Литература: [2, 7].

1.2.2 Порядок выполнения работы

Программно-технические средства реализации современных офисных технологий прошли свой технологический путь развития и приобрели достаточно стандартизированные формы представления и обработки информации.

Как аспирант уже изучил в курсе «Современные компьютерные технологии» [2]: «**Офисные технологии** — результат интеграции технологических достижений средств вычислительной техники применительно к прикладному направлению, связанному с *индивидуальной автоматизированной обработкой информации*. …

… офисные технологии рассматриваются в трех аспектах:

- как *набор приложений*;
- как *элемент системы документооборота*;
- как *элемент интеграции* со стационарными хранилищами информации. ...

Как правило, набор офисных пакетов содержит следующийе компоненты:

- *Текстовый процессор* — средство для создания сложных документов, содержащих текст, таблицы, графику и другое;
- *Табличный процессор* — средство для массовых табличных вычислений;
- *Программу подготовки и демонстрации презентаций* — позволяющую создавать красочные и впечатляющие электронные презентации;
- *Упрощенную СУБД* — позволяющую управлять базами данных или обеспечивать доступ к СУБД на уровне языка SQL;
- *Графическую программу* — позволяющую редактировать графические форматы файлов;
- *Редактор формул* — позволяющий создавать и редактировать математические формулы. ...».

В данной практической работе рассматривается основные форматы объектов, представляющих результаты обработки информации типовыми средствами офисных технологий.

1.2.2.1 История вопроса

Исторически, в России наиболе популярным офисным пакетом является Microsoft Office [1.2.1]: «**Microsoft Office** — офисный пакет приложений, созданных корпорацией Microsoft для операционных систем Microsoft Windows, Windows Phone, Android, OS X, iOS. В состав этого пакета входит программное обеспечение для работы с различными типами документов: текстами, электронными таблицами, базами данных и др. Microsoft Office является сервером OLE-объектов и его функции могут использоваться другими приложениями, а также самими приложениями Microsoft Office. Поддерживает скрипты и макросы, написанные на VBA.».

Первый выпуск этого пакета состоялся 19 ноября 1990 года и, как все аналогичные пакеты того времени, содержал проприоретарные бинарные форматы представления и хранения обрабатываемых данных.

Долгое время, серьезную конкуренцию MS Office составлял, появившийся в 1985 году, пакет StarOffice [1.2.2]: «**StarOffice** (Oracle OpenOffice) — проприоритарный офисный пакет, разработанный компанией StarDivision. Первый серьёзный аналог офисного пакета Microsoft Office. Стал основой для создания OpenOffice.org и других офисных пакетов на его основе.». Работа над этим пакетом была прекращена в декабре 2010 года.

В апреле 2002 года, преемником StarOffice стал свободно распространяемый пакет OpenOffice [1.2.3]: «**Apache OpenOffice** (ранее **OpenOffice.org**, **OO.org**, **OO.o**, **OOo**) — свободный пакет офисных приложений. Конкурирует с коммерческими офисными пакетами (в том числе Microsoft Office) как на уровне форматов, так и на уровне интерфейса пользователя. Одним из первых стал поддерживать новый открытый формат OpenDocument (ISO/IEC 26300). Официально поддерживается на платформах Linux, Microsoft Windows, macOS Intel/ PowerPC (поддержка оболочки Aqua находится в стадии альфа-тестирования) и раньше поддерживался Solaris SPARC/Intel. Существуют порты для OpenSolaris, FreeBSD, Linux PowerPC, OS/2 и Android.

Основан на коде StarOffice, который был приобретён, а затем выпущен с открытым исходным кодом фирмой Sun Microsystems. После покупки последней права на OO.o перешли к компании Oracle.».

Наконец, в октябре 2010 года появился пакет LibreOffice [1.2.4]: «**LibreOffice** — кроссплатформенный, свободно распространяемый офисный пакет с открытым исходным кодом, созданный как ответвление OpenOffice в 2010 году. Разрабатывается сообществом из более чем 480 программистов под эгидой некоммерческого фонда The Document Foundation за счёт пожертвований отдельных лиц и организаций.

Офисный пакет содержит в себе текстовый и табличный процессор, прог-

рамму для подготовки и просмотра презентаций, векторный графический редактор, систему управления базами данных и редактор формул. Основным форматом файлов, использующимся в приложении, является открытый международный формат OpenDocument (ODF, ISO/IEC 26300), но возможна работа и с другими популярными форматами, в том числе Office Open XML, DOC, XLS, PPT, CDR.

Офисный пакет распространяется под общественной лицензией GNU LGPL, поэтому может свободно устанавливаться и использоваться в бюджетных и коммерческих организациях, а также на домашних компьютерах и в учебных заведениях.».

1.2.2.2 Стандарты XML-форматов офисных документов

Наличие множества проприоритарных закрытых форматов естественным образом тормозит развитие и практическое использование офисных пакетов. Кроме того, возникают серьезные проблемы автоматизации документооборона на предприятиях, которым требуется достаточно универсальный формат электронного документа. В этих условиях, разработчики офисных пакетов стали обращать внимание на достаточно универсальный язык описания данных — XML. С целю решения основных, перечисленных выше проблем, был создан консорциум, известный в настоящее время как OASIS.

Как отмечено в [1.2.5]: «OASIS была основана под названием «SGML Open» в 1993 году. Она началась как консорциум поставщиков и пользователей, занимающихся разработкой руководящих принципов взаимодействия между продуктами, поддерживающими стандартный обобщенный язык разметки (SGML). В 1998 году консорциум изменил свое название на «OASIS» (Организация по совершенствованию стандартов структурированной информации), чтобы отразить расширенный объем технической работы.».

В июле 2006 года OASIS подготовила стандарт, который был опубликован как «ISO/IEC 26300:2006 - Information technology -- Open Document Format for Office Applications (OpenDocument) v1.0». В 2010 году этот стандарт был переведен и опубликован как «ГОСТ Р ИСО/МЭК 26300-2010» [7]. С 01 июня 2011 года стандарт был введен в силу.

Замечание

29 сентября 2011 OASIS опубликовала версию 1.2 документа «Open Document Format for Office Applications (OpenDocument)» [1.2.6]. ГОСТ на этот документ пока не опубликован.

Таким образом, к маю 2006 года был разработатн стандарт на универсальный формат офисных приложений, известный как формат .odf [1.2.7]: «**OpenDocument Format**, **ODF** (от англ. *OASIS Open Document Format for Office Application* — рус. *открытый формат документов для офисных приложений*) — открытый формат файлов документов для хранения и обмена редактируемыми офисными документами, в том числе текстовыми документами (такими как

заметки, отчёты и книги), электронными таблицами, рисунками, базами данных, презентациями.

Стандарт был разработан индустриальным сообществом OASIS и основан на XML-формате. 1 мая 2006 года принят как международный стандарт ISO/IEC 26300. В июле 2015 года стандартизован ODF версии 1.2.

Стандарт был совместно и публично разработан различными организациями, доступен для всех и может быть использован без ограничений. OpenDocument представляет собой альтернативу частным закрытым форматам, (включая Word (`.doc`), Exel (`.xls`) и PowerPoint (`.ppt`) — форматы, используемые в Microsoft Office 97—2003), а также формату Microsoft Office Open XML - .docx

Файл OpenDocument — ZIP-архив, включающий в себя файловую иерархию, содержащую XML-файл самого документа, файлы включений (например — картинок), вспомогательные файлы с метаинформацией, картинка-миниатюра страницы документа и тому подобное.».

В таблице 1.2 представлены форматы вариантов *.odf*, реализованные в офисном пакете LibreOffice. Полный перечень форматов следует смотреть в [1.2.7].

Таблица 1.2 (заимствовано из [1.2.4])

Модуль		***Назначение***	***Формат***	***Похожие приложения***
	Writer	Текстовый процессор и визуальный редактор HTML	***odt***	AbiWord, KWord, Microsoft Word, Pages, TextMaker
	Calc	Табличный процессор	***ods***	Gnumeric, KCells, Microsoft Excel, Numbers, PlanMaker
	Impress	Программа подготовки презентаций	***odp***	Keynote, KPresenter, Microsoft PowerPoint, SoftMaker Presentations
	Draw	Векторный графический редактор	***odg***	Adobe Illustrator, CorelDRAW, Dia, Inkscape, Kivio, Microsoft Visio
	Math	Редактор формул	***odf***	KFormula, MathType, Microsoft Equation Tools
	Base	Механизм подключения к внешним СУБД и встроенная СУБД HSQLDB	***odb***	Kexi, Microsoft Access

Следует также учесть, что пакет Microsoft Office придерживается своего собственного формата Open Office XML [1.2.8]: «**Office Open XML** (OOXML, DOCX, XLSX, PPTX, проект ISO/IEC IS 29500:2008) — серия форматов файлов для хранения электронных документов пакетов офисных приложений — в частности,

Microsoft Office. Формат представляет собой zip-архив, содержащий текст в виде XML, графику и другие данные, которые могут быть переведены в последовательность битов (сериализованы) с применением защищённых патентами двоичных форматов, спецификации которых были опубликованы Microsoft для пользователей OOXML на условиях Microsoft Open Specification Promise.

Первоначально формат создавался как замена прежнему двоичному формату документов, который использовали приложения Microsoft Office вплоть до версии Office 2003 включительно. В 2006 году формат Office Open XML был объявлен свободным и открытым форматом Ecma International. Он является форматом по умолчанию для приложений Microsoft Office 2007 и более поздних.

Две разные версии OOXML определены в ECMA-376 и в ISO 29500:2008. Полная поддержка формата ISO 29500 ожидалась (но не появилась) в Microsoft Office 2010.».

Что касается непосредственно форматов документов ODF, то, как отчечено в [1.2.9]: «Наиболее распространенными расширениями файлов, используемыми для документов OpenDocument, являются .odt для текстовых документов, .ods для электронных таблиц, .odp для презентационных программ и .odg для графики. Они легко запоминаются, рассматривая «.od» как краткое для «OpenDocument», а затем отмечая, что последняя буква указывает свой более специфический тип (например, t для текста). Ниже приведен полный список типов документов, показывающий тип файла, рекомендуемое расширение файла и тип MIME:

File type	Extension	MIME Type	ODF specification
Text	.odt	application/vnd.oasis.opendocument.text	1.0
Spreadsheet	.ods	application/vnd.oasis.opendocument.spreadsheet	1.0
Presentation	.odp	application/vnd.oasis.opendocument.presentation	1.0
Drawing	.odg	application/vnd.oasis.opendocument.graphics	1.0
Chart	.odc	application/vnd.oasis.opendocument.chart	1.0
Formula	.odf	application/vnd.oasis.opendocument.formula	1.0
Image	.odi	application/vnd.oasis.opendocument.image	1.0
Master Document	.odm	application/vnd.oasis.opendocument.text-master	1.0
Database	.odb	application/vnd.sun.xml.base[3][4]	not defined in ODF 1.0/1.1 specifications; used in OpenOffice.org 2.x
Database	.odb	application/vnd.oasis.opendocument.base	ODF 1.2; used in OpenOffice.org 3.x
Database	.odb	application/vnd.oasis.opendocument.database	defined in IANA registration
all OpenDocument single/flat XML files	not defined	text/xml	1.0

OpenDocument также поддерживает набор типов шаблонов. Шаблоны представляют собой информацию о форматировании (включая стили) для документов, без самого содержимого. Рекомендуемое расширение имени файла начинается с «.ot» (интерпретируемое как сокращение для «шаблона OpenDocument»), с последней буквой, указывающей, какой шаблон (например, «t» для текста). Поддерживаемый набор включает:

File type	Extension	MIME Type	ODF specification
Text	.ott	application/vnd.oasis.opendocument.text-template	1.0
Spreadsheet	.ots	application/vnd.oasis.opendocument.spreadsheet-template	1.0
Presentation	.otp	application/vnd.oasis.opendocument.presentation-template	1.0
Drawing	.otg	application/vnd.oasis.opendocument.graphics-template	1.0
Chart template	.otc	application/vnd.oasis.opendocument.chart-template	1.0
Formula template	.otf	application/vnd.oasis.opendocument.formula-template	1.0
Image template	.oti	application/vnd.oasis.opendocument.image-template	1.0
Web page template	.oth	application/vnd.oasis.opendocument.text-web	1.0

Файл OpenDocument обычно состоит из стандартного ZIP-архива (архив JAR), содержащего несколько файлов и каталогов; но файл OpenDocument также может состоять только из одного XML-документа. Файл OpenDocument обычно представляет собой набор из нескольких поддокументов в пакете (ZIP). Файл OpenDocument как один XML широко не используется. Согласно спецификации OpenDocument 1.0 спецификация файла ZIP определена в Info-ZIP Application Note 970311, 1997. Простой механизм сжатия, используемый для пакета, обычно делает файлы OpenDocument значительно меньше, чем эквивалентные файлы Microsoft .doc или .ppt. Этот меньший размер важен для организаций, которые хранят огромное количество документов в течение длительных периодов времени, и тем организациям, которые должны обмениваться документами по низкоскоростным соединениям. После распаковки, большинство данных содержится в простых текстовых XML-файлах, поэтому несжатое содержимое данных имеет возможность легкой модификации и обработки XML-файлов. Стандарт также позволяет создать единый XML-документ, который использует <office: document> в качестве корневого элемента для использования в обработке документов.

Стандарт позволяет включать каталоги для хранения изображений, анима-

ции non-SMIL и другие файлы, которые используются документом, но не могут быть выражены непосредственно в XML.

Из-за используемого открытого формата сжатия, пользователь может извлечь файл контейнера для ручного редактирования содержащихся файлов. Это позволяет восстанавливать поврежденный файл или манипулировать содержимым на нижнем уровне.

Зашифрованный набор файлов и каталогов включает в себя следующее:

- XML-файлы
    - content.xml
    - meta.xml
    - settings.xml
    - styles.xml
- Другие файлы
    - mimetype
- Директории
    - META-INF/
        - manifest.xml
    - Thumbnails/
        - thumbnail.png

Формат OpenDocument обеспечивает строгое разделение между контентом, макетом и метаданными. Наиболее значимые компоненты формата описаны в подразделах ниже. Файлы в формате XML дополнительно определяются с использованием языка RELAX NG для определения схем XML. RELAX NG сам определяется спецификацией OASIS, а также второй частью международного стандарта ISO/IEC 19757: Языки определения схемы документа (DSDL). ...».

Далее, приведем ряд определений, также заимствованных из [1.2.9]: «...

content.xml

content.xml, самый важный файл, включающий фактическое содержимое документа (за исключением двоичных данных, например изображений). Базовым форматом является HTML, и, хотя он намного сложнее, он должен быть вполне понятным для людей:

```
<text:h style-name="Heading_2">This is a title</text:h>
<text:p style-name="Text_body"/>
<text:p style-name="Text_body">
  This is a paragraph. The formatting information is
  in the Text_body style. The empty text:p tag above
  is a blank paragraph (an empty line).
</text:p>
```

styles.xml

styles.xml содержит информацию о стиле. OpenDocument строго использует

стили для форматирования и компоновки. Большая часть информации о стиле находится в этом файле (хотя некоторые из них находятся в content.xml). Типы стилей включают:

- Paragraph styles
- Page styles
- Character styles
- Frame styles
- List styles

Формат OpenDocument несколько необычен тем, что нельзя не использовать стили для форматирования. Даже «ручное» форматирование реализуется через стили (приложение динамически создает новые стили по мере необходимости).

meta.xml

meta.xml содержит метаданные файла. Например, автор, «Последнее изменение», дата последней модификации и т. д. Содержимое выглядит примерно так:

```
<meta:creation-date>2003-09-10T15:31:11</meta:creation-date>
<dc:creator>Daniel Carrera</dc:creator>
<dc:date>2005-06-29T22:02:06</dc:date>
<dc:language>es-ES</dc:language>
<meta:document-statistic
   table-count="6" object-count="0"
   page-count="59" paragraph-count="676"
   image-count="2" word-count="16701"
   character-count="98757"/>
```

Имена тегов <dc: ...> относятся к стандарту Dublin Core XML.

settings.xml

settings.xml включает такие параметры, как коэффициент масштабирования или положение курсора. Это свойства, которые не являются содержимым или макетом.

mimetype (file)

mimetype - это всего лишь однострочный файл с типом документа. Одним из следствий этого является то, что расширение файла фактически несущественно для формата. Расширение файла доступно только пользователю.

Thumbnails (directory)

Thumbnails - это отдельная директория для миниатюры документа. Эскиз должен быть сохранен как «thumbnail.png». Представление эскиза документа должно быть сгенерировано по умолчанию при сохранении файла. Это должно

быть представление первой страницы, первого листа и т. д. документа. Требуемый размер для эскизов - 128x128 пикселей. Чтобы соответствовать Стандарту управления эскизами (TMS) на сайте www.freedesktop.org, миниатюры должны быть сохранены как 8-битное, не чересстрочное PNG-изображение с полной альфа-прозрачностью.

META-INF (directory)

META-INF - отдельная директория. Информация о файлах, содержащихся в пакете OpenDocument, хранится в XML-файле, который называется файлом манифеста. Файл манифеста всегда хранится в имени пути META-INF/manifest.xml. Основные части информации, хранящиеся в манифесте:

- Список всех файлов в пакете.
- Тип медиафайла каждого файла в пакете.
- Если файл, хранящийся в пакете, зашифрован, информация, необходимая для дешифрования файла, сохраняется в манифесте.

Pictures (directory)

Pictures представляют собой отдельную директорию для изображений, включенных в документ. Эта директория не определена в спецификации OpenDocument. Файлы в этой папке могут использовать различные форматы изображений, в зависимости от формата вставленного файла. Хотя данные изображения могут иметь произвольный формат, рекомендуется, чтобы растровая графика хранилась в формате PNG, а векторная графика - в формате SVG.

Reuse of existing formats

По дизайну OpenDocument повторно использует существующие стандартные стандарты XML, когда они доступны, и создает новые теги только там, где ни один из существующих стандартов не может обеспечить необходимую функциональность. Таким образом, OpenDocument использует подмножество DublinCore для метаданных, MathML для отображаемых формул, SMIL для мультимедиа, XLink для гиперссылок и т. д.

Хотя OpenDocument не полностью повторно использует SVG для векторной графики, OpenDocument использует векторную графику, совместимую с SVG, в пространстве имен, специфичном для ODF-формата, но также включает в себя не-SVG-графику. ...».

1.2.2.3 Учебные задания

Теперь аспиранту следует закрепить представленный теоретический материал практическим исследованием. Для этого следует воспользоваться одним из свободно распространяемых офисных пакетов: LibreOffice или OpenOffice. Далее, рассмотрим использование LibreOffice.

Из таблицы 1.2 следует выбрать одну из программ и создать небольшой документ, а затем:

1. Провести обработку программной `zip_check`, как это делалось в предыдущем проктическом занятии.
2. Перенести документ в отдельную директорию и распаковать его любым архиватором: `unzip`, `rar`, `7-zip` , `WinZip` или другим.
3. Исследовать содержимое распакованных директорий и файлов.
4. Описать результаты исследования в личном отчете.
5. Желательно, внести предложение: каким образом можно использовать технологию zip-архива в конкретном прикладном проекте на языке C.

Замечание

Офисные пакеты LibreOffice и OpenOffice успешно работают и с файлами пакета Microsoft Office, хотя имеются и отдельные артефакты их воспроизведения.

Указанное выше исследование можно провести и файлами Microsoft Office, но следует учесть что структура их файлов отличается от структуры файлов формата ODF (см. источник [1.2.8]).

1.2.3 Список использованных источников

1.2.1 Microsoft Office — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/Microsoft_Office). Проверено: 06.11.2018.

1.2.2 StarOffice — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/StarOffice). Проверено: 06.11.2018.

1.2.3 OpenOffice — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/OpenOffice). Проверено: 06.11.2018.

1.2.4 OpenOffice — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/LibreOffice). Проверено: 06.11.2018.

1.2.5 About Us | OASIS: [Электронный документ]. - (https://www.oasis-open.org/org). Проверено: 06.11.2018.

1.2.6 OASIS Open Document Format for Office Applications (OpenDocument) Version 1.2 — OpenDocument-v1.2-os.pdf: [Электронный документ]. - (http://docs.oasis-open.org/office/v1.2/os/OpenDocument-v1.2-os.pdf). Проверено: 06.11.2018.

1.2.7 OpenDocument - Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/OpenDocument). Проверено: 06.11.2018.

1.2.8 Office Open XML - Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/Office_Open_XML). Проверено: 06.11.2018.

1.2.9 OpenDocument technical specification - Wikipedia: [Электронный документ]. - (https://en.wikipedia.org/wiki/OpenDocument_technical_specification). Проверено: 06.11.2018.

1.3 Практическая работа №3 «Технологии хранения данных»

Цель работы: практическое освоение технологии хранения и обработки данных на основе использование встроенной СУБД SQLite.

1.3.1 Самостоятельная работа

Понятие информационной системы, банки и базы данных. Логическая и физическая организация баз данных. Модели представления данных, архитектура и основные функции СУБД. Распределенные БД. Принципиальные особенности и сравнительные характеристики файл-серверной, клиент-серверной и интранет технологий распределенной обработки данных. Реляционный подход к организации БД. Базисные средства манипулирования реляционными данными. Методы проектирования реляционных баз данных (нормализация, семантическое моделирование данных, ER-диаграммы). Языки программирования в СУБД, их классификация и особенности. Стандартный язык баз данных SQL. Перспективные концепции построения СУБД (ненормализованные реляционные БД, объектно-ориентированные базы данных и др.).

Литература: [2, 8 или 9].

1.3.2 Порядок выполнения работы

Основы технологии хранения данных аспирант начал изучать на уровне бакалавриата в дисциплине «Базы данных». Более широко эти технологии были изучены в дисциплине «Современные компьютерные технологии» (см. [2], раздел 3). Наконец, по материалу предыдущего подраздела, на примере пакета LibreOffice, мы видим, что технологии хранения информации стали частью офисных технологий (см. таблицу 1.2).

В целом, можно отметить, что современные приложения становятся все более и более информационно насыщенными, а также - инфраструктурно сложными. На практике это требует все более сложных и разнообразных вариантов их использования, которые по традиции отражаются в разлиных параметрах настройки приложений. Стремление делать приложения все более универсальными и гибкими только увеличивает количество настроечных параметров, нелинейно влияющих на качество их работы.

В такой ситуации возникает настоятельная потребность в специальных средствах хранения конфигурационных файлов и сопутствующих настроек приложений. И такие средства уже созданы в достаточном количестве. Фактически каждый, пусть даже не очень большой проект, имеет свои средства конфигурирования и часто — уникальные. Следовательно, актуальным является вопрос об использовании универсальных хранилищ данных, какими являются СУБД.

1.3.2.1 История вопроса

Наиболее известными в настоящее время компаниями, реализующими технологии хранения информации уровня предприятия, являются Oracle и Microsoft.

Исторически, общепризнанным лидером создания средств СУБД является корпорация Oracle, основанная 1977 году [1.3.1]: «**Oracle** (*Oracle Corporation*) — американская корпорация, второй по величине доходов производитель программного обеспечения (после Microsoft), крупнейший производитель программного обеспечения для организаций, крупный поставщик серверного оборудования.

Компания специализируется на выпуске систем управления базами данных, связующего программного обеспечения и бизнес-приложений (ERP- и CRM-систем, специализированных отраслевых приложений). Наиболее известный продукт компании — Oracle Database, который компания выпускает с момента своего основания. С 2008 года корпорация освоила выпуск интегрированных аппаратно-программных комплексов, а с 2009 года в результате поглощения Sun Microsystems стала производителем серверного оборудования, до этого компания выпускала исключительно программное обеспечение. ...».

В апреле 1989 года корпорация Microsoft создала свою реляционную СУБД [1.3.2]: «**Microsoft SQL Server** — система управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов — Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка. ...».

В 1995 году Apache Software Foudation выпустила свой знаменитый Веб-сервер: Apache HTTP Server [1.3.3]. В качестве СУБД в ней использовался продукт MySQL [1.3.4]: «**MySQL** — свободная реляционная система управления базами данных. Разработку и поддержку MySQL осуществляет корпорация Oracle, получившая права на торговую марку вместе с поглощённой Sun Microsystems, которая ранее приобрела шведскую компанию MySQL AB. Продукт распространяется как под GNU General Public License, так и под собственной коммерческой лицензией. Помимо этого, разработчики создают функциональность по заказу лицензионных пользователей. Именно благодаря такому заказу почти в самых ранних версиях появился механизм репликации. ...». Позднее, MySQL стал заменяться СУБД MariaDB [1.3.5]: «**MariaDB** — ответвление от системы управления базами данных MySQL, разрабатываемое сообществом под лицензией GNU GPL. Разработку и поддержку MariaDB осуществляет компания MariaDB Corporation Ab и фонд MariaDB Foundation.

Толчком к созданию стала необходимость обеспечения свободного статуса, в противовес политике лицензирования MySQL компанией Oracle. Основателями проекта выступили первоначальные разработчики MySQL. ...».

На этом примере хорошо прослеживаются две тенденции:

1) появляются программные системы, использующие СУБД, которым не требуются столь мощные средства как Oracle или MS SQL Server;
2) основным доводом для выбора СУБД становятся не качества продукта, а лицензионные ограничения на его использование.

В 2003 году появляется международный стандарт ISO/IEC TR 10032-2003 на эталонную модель управления данными [1.3.5]. В 2007 году появляется ГОСТ Р ИСО/МЭК 10032-2007 - Эталонная модель управления данными [8 или 9]. Эти документы стали определять современные тенденции развития СУБД.

В России большой популярностью пользуется PostgreSQL [1.3.6], которой в 2015 году был посвящен цикл статей [1.3.7, 1.3.8]. Общая тема связана с импортозамещением программного обеспечения. Как конкурент, рассматривается СУБД Firebird [1.3.9]: «**Firebird** (FirebirdSQL) — свободная кроссплатформенная реляционная система управления базами данных (РСУБД), работающая на macOS, Linux, Microsoft Windows и разнообразных Unix платформах.

Firebird используется в различных промышленных системах (складские и хозяйственные, финансовый и государственный сектора) с 2001 г. Это коммерчески независимый проект C и C++ программистов, технических советников. ...».

В связи с высокой переносимостью ПО на языке Java, большой популярностью пользуются СУБД Apache Derby [1.3.10]: «**Apache Derby** — реляционная СУБД, написанная на Java, предназначенная для встраивания в Java-приложения или обработки транзакций в реальном времени. Занимает 2 MB на диске. Распространяется на условиях лицензии Apache 2.0. Ранее известна как **IBM Cloudscape**. Oracle распространяет те же бинарные файлы под именем **Java DB**. ...».

В рамках проекта OpenDocument Format используется СУБД HSQLDB [1.3.11]: «**HSQLDB** — реляционная СУБД с открытым исходным кодом. Распространяется по собственной лицензии, близкой к лицензии BSD. Поддерживает стандарты SQL-92, SQL:1999, SQL:2003 и SQL:2008.

HSQLDB полностью написана на Java и отличается небольшим размером (размер около 1100 кБ для версии 2.0). Может использоваться и как отдельный сервер с поддержкой сетевых соединений по JDBC, и в виде библиотеки для использования непосредственно в коде программы.

HSQLDB используется во многих известных программных продуктах, в частности, в LibreOffice, OpenOffice.org, Jboss, Openfire, JAMWiki. ...».

Кроме реляционных СУБД набирают популярность проекты, поддерживающие древовидные модели данных XML. Одним из таких проектов является СУБД Sedna [1.3.12]: «**Sedna** — система управления базами данных, изначально спроектированная для хранения и обработки XML-данных. Разработана и развивается Отделом управления данными и информационных систем Института системного программирования РАН. Система распространяется в открытых исходных текстах. Существуют версии под Windows, Linux, Mac OS и FreeBSD.

СУБД Sedna поддерживает древовидную модель данных (хранимых в

двоичном виде), которые загружаются и извлекаются в виде XML-документов. Данные оптимизируются и индексируются для рационального хранения и быстрого доступа.

Прирождённые XML-СУБД в настоящее время активно развиваются — в ряде применений они начинают конкурировать с традиционными реляционными СУБД. СУБД Sedna выглядит достойно в сравнении с другими XML-СУБД: во-первых, за счёт эффективных внутренних механизмов (например, собственного 64-разрядного диспетчера памяти, адресации и подкачки), во-вторых, из-за полного соответствия стандарту на язык запросов XQuery, в третьих, за счёт возможности интеграции в XML БД наследованных реляционных источников данных ... ».

В современных программных проектах все чаще начинают использовать встроенные СУБД, обеспечивающие не только поддержку обрабатываемых данных самого приложения, но и данные, относящиеся к инфраструктуре прикладного использования этого приложения. Примером такого решения является использование СУБД SQLite [1.3.13]: «**SQLite** — компактная встраиваемая СУБД. Исходный код библиотеки передан в общественное достояние. В 2005 году проект получил награду Google-O'Reilly Open Source Awards. …

Слово «встраиваемый» (embedded) означает, что SQLite не использует парадигму клиент-сервер, то есть движок SQLite не является отдельно работающим процессом, с которым взаимодействует программа, а представляет собой библиотеку, с которой программа компонуется, и движок становится составной частью программы. Таким образом, в качестве протокола обмена используются вызовы функций (API) библиотеки SQLite. Такой подход уменьшает накладные расходы, время отклика и упрощает программу. SQLite хранит всю базу данных (включая определения, таблицы, индексы и данные) в единственном стандартном файле на том компьютере, на котором исполняется программа. Простота реализации достигается за счёт того, что перед началом исполнения транзакции записи весь файл, хранящий базу данных, блокируется; ACID-функции достигаются в том числе за счёт создания файла журнала.

Несколько процессов или потоков могут одновременно без каких-либо проблем читать данные из одной базы. Запись в базу можно осуществить только в том случае, если никаких других запросов в данный момент не обслуживается; в противном случае попытка записи оканчивается неудачей, и в программу возвращается код ошибки. Другим вариантом развития событий является автоматическое повторение попыток записи в течение заданного интервала времени.

В комплекте поставки идёт также функциональная клиентская часть в виде исполняемого файла `sqlite3`, с помощью которого демонстрируется реализация функций основной библиотеки. Клиентская часть является кроссплатформенной утилитой командной строки. ...».

СУБД SQLite используется во многих проектах, например, Mozilla Fiefox (браузер), Mozilla Thunderbird (почтовый клиент), Skype и других. Таким образом, аспиранту крайне необходимо освоить работу с ней.

1.3.2.2 Встраеваемая СУБД проекта SQLite

Для полного и подробного изучения инструментов СУБД SQLite можно воспользоваться учебным интернет-пособием [1.3.14]. Аспиранту, для краткого изучения и выполнения индивидуального практического задания, рекомендуется интернет-источник [1.3.15], кратко объясняющий и демонстрирующий работу с SQLite.

На практическом занятии разберем ситуацию, когда:

- база данных проекта создается заранее, используя инструментальные средства СУБД;
- программа (написанная на языке С) читает данные из базы данных и использует их по своему назначению.

Конкретный пример:

- таблица ***Students*** содержит общий список фамилий студентов вуза;
- таблица ***Themes*** содержит общий список предметов, преподаваемых в вузе;
- таблица ***Exams*** содержит записи о назначенных студенту предметов, результаты экзаменов и коментарии к ним;
- необходимо созать базу данных с именем ***test3.db***, содержащую указанные таблицы;
- написать программу на языке С, котоорая обращается к базе данных и выводит результаты аттестации конкретного студента.

Реализацию данного примера проведем в три этапа:

- краткое изучение инструментальных средств СУБД SQLite;
- создание базы данных ***test3.db***;
- написание, отладка и исследование с промощью программы ***test3_exams***.

Этап 1. Инструментальные средства SQLite.

СУБД SQLite имеет утилиту командной строки ***sqlite3***, с помощью которой осуществляются все административные операции с базами данных. Но, прежде чем воспользоваться ей, создадим описание базы данных, воспользовавшись типами данных СУБД SQLite, приведенных в таблице 1.3. Для этого, откроем в текстовом редакторе файл ***test3.sql*** и наберем в нем текст, как показано на листинге 1.2.

Замечание

Все, приведенные в данном примере действия, желательно проводить в отдельной директории. Это не только повысит наглядность представления результатов, но и защитит файлы от случайного их изменения, вследствие возможного совпадения имен.

Таблица 1.3 (заимствованная из [1.3.14]: https://www.tutorialspoint.com/sqlite/sqlite_data_types.htm)

Sr.No.	*Storage Class & Description*
1	**NULL** Значение NULL.
2	**INTEGER** Значение представляет собой целое число со знаком, сохраненное в 1, 2, 3, 4, 6 или 8 байтах в зависимости от величины значения.
3	**REAL** Значение представляет собой значение с плавающей запятой, которое хранится как 8-байтовое число с плавающей точкой IEEE.
4	**TEXT** Значение представляет собой текстовую строку, хранящуюся с использованием кодировки базы данных (UTF-8, UTF-16BE или UTF-16LE)
5	**BLOB** Значение представляет собой блок данных, который хранится точно так же, как он был введен.

Листинг 1.2 — Содержимое файла test3.sql

```
-- --------------------------
-- Пример sql-файла для третьего практического занятия
-- по дисциплине САУиОИ, аспиратура направления 09.06.01
-- Резник В.Г., 13.11.2018
-- --------------------------
PRAGMA encoding = "UTF-8";
PRAGMA foreign_keys = ON;

-- Начинаем транзакцию
BEGIN TRANSACTION;

-- Сначала удаляем таблицы, если они существуют.
DROP TABLE IF EXISTS Exams;
DROP TABLE IF EXISTS Students;
DROP TABLE IF EXISTS Themes;

-- Создание таблицы Students
CREATE TABLE [Students]
(
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      family TEXT NOT NULL UNIQUE
);
INSERT INTO Students VALUES(NULL, 'Петров');
INSERT INTO Students VALUES(NULL, 'Сидоров');
INSERT INTO Students VALUES(NULL, 'Андреев');
INSERT INTO Students VALUES(NULL, 'Никишин');
INSERT INTO Students VALUES(NULL, 'Бусоедов');
INSERT INTO Students VALUES(NULL, 'Корсаков');

-- Создание таблицы Themes
```

```
CREATE TABLE [Themes]
(
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      name TEXT NOT NULL UNIQUE
);
INSERT INTO Themes VALUES(NULL, 'Операционные системы');
INSERT INTO Themes VALUES(NULL, 'Базы данных');
INSERT INTO Themes VALUES(NULL, 'Современные компьютерные технологии');
INSERT INTO Themes VALUES(NULL, 'Иностранный язык');
INSERT INTO Themes VALUES(NULL, 'Теория систем и системный анализ');
INSERT INTO Themes VALUES(NULL, 'Методы планирования эксперимента');

-- Создание таблицы Themes с двумя внешними ключами:
-- на таблицы Students и Themes
CREATE TABLE [Exams]
(
      id_st INTEGER NOT NULL,
      id_th INTEGER NOT NULL,
      value INTEGER NOT NULL default 0,
      comment TEXT NOT NULL default 'Не атестован',
      foreign key(id_st) references Students(id)
            on delete cascade,
      foreign key(id_th) references Themes(id)
            on delete cascade,
      primary key(id_st, id_th)
);
INSERT INTO Exams VALUES(1, 1, 0, 'Не атестован');
INSERT INTO Exams VALUES(1, 2, 0, 'Не атестован');
INSERT INTO Exams VALUES(1, 5, 0, 'Не атестован');

INSERT INTO Exams VALUES(2, 2, 0, 'Не атестован');
INSERT INTO Exams VALUES(2, 3, 0, 'Не атестован');
INSERT INTO Exams VALUES(2, 4, 0, 'Не атестован');

-- Завершаем транзакцию
COMMIT;

-- Тест таблиц
SELECT ' ' AS " ";
SELECT 'Таблица Students:' AS "=== Тестирование таблиц ===";
SELECT * FROM Students;

SELECT 'Таблица Themes:' AS "";
SELECT * FROM Themes;

--SELECT 'Таблица Exams:' AS "";
--SELECT * FROM Exams;
```

Конец листинга 1.2

Теперь для того, чтобы удобней было работать с СУБД SQLite, создадим файл *.sqliterc*, содержимое которого показано на рисунке 1.2, и поместим в домашнюю директорию пользователя.

Далее, изучим основные команды утилиты *sqlite3*, представленые в таблице 1.4.

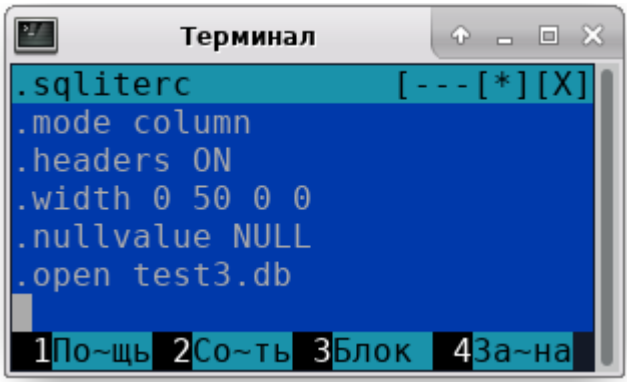

Рисунок 1.2 — Файл конфигурации утилиты sqlite3

Таблица 1.4 (заимствовано из [1.3.5])

Команда	***Краткое описание***
.databases	Вывод списка имен всех подключенных в текущем сеансе баз данных и соответствующих им файлов.
.dump ?TABLES?:	Дамп всех SQL инструкций использованных в создании БД или отдельной таблицы в текстовом формате
.echo ON|OFF	ВКЛ | ВЫКЛ эхо введенных команд
.exit	Выход из программы
.explain ON|OFF	Управляет режимом вывода виртуальных машинных команд. Используется при выполнении SQL запроса EXPLAIN.
.header(s) ON|OFF	ВКЛ | ВЫКЛ показ заголовков столбцов
.import FILE TABLE	Импорт данных из файла FILE в таблицу TABLE
.indices TABLE	Показывает имена всех индексов таблицы
.mode MODE	Установка режима вывода: line(s), column(s), insert, list, html и других
.nullvalue STRING	Напечатает строку STRING вместо NULL данных при выводе SQL запроса SELECT
.output FILENAME	Послать весь вывод в файл FILENAME
.output stdout	Послать весь вывод на экран
.promt MAIN COTINUE	Изменить стандартную строку подсказки
.quit	Выход их программы
.read FILENAME	Выполнение SQL инструкций из файла FILENAME

Команда	***Краткое описание***
.schema ?TABLE?	Покажет текст SQL инструкции CREATE для всех таблиц или указанной таблицы
.separator STRING	Изменить строку разделитель колонок, используется при выводе SQL запроса SELECT и команды .import
.show	Показать значения установленных переменных
.tables ?PATTERN?	Вывод списка имен таблиц БД (возможно по шаблону)
.timeout MS	Блокирование открытия таблиц на число миллисекунд MS
.width NUM NUM :	Установка ширины столбцов при выводе в режиме column

Этап 2. Создание базы данных ***test3.db***.

Изучив основные команды утилиты *`sqlite3`*, перейдем в директорию, где был размещен файл ***test3.sql***, и выполним команды:

```
sqlite3
.read test3.sql
.quit
```

как это показано на рисунке 1.3.

В результате, будет создана база данных ***test3.db***, в чем можно убедиться, исследовав содержимое директории.

Протестируем ее запросом, представленным на листинге 1.3, результат которого показан на рисунке 1.4.

Листинг 1.3 — Содержимое файла query3.sql

```
-- --------------------------
-- Пример sql-запроса для третьего практического занятия
-- по дисциплине САУиОИ, аспиратура направления 09.06.01
-- Резник В.Г., 13.11.2018
-- --------------------------
PRAGMA encoding = "UTF-8";
PRAGMA foreign_keys = ON;

SELECT Students.family,
       Exams.value, Exams.comment,
       Themes.name
FROM
       Students, Exams, Themes
WHERE
       Students.family = 'Петров' AND
       Students.id   = Exams.id_st AND
       Exams.id_th   = Themes.id ;
```

Конец листинга 1.3

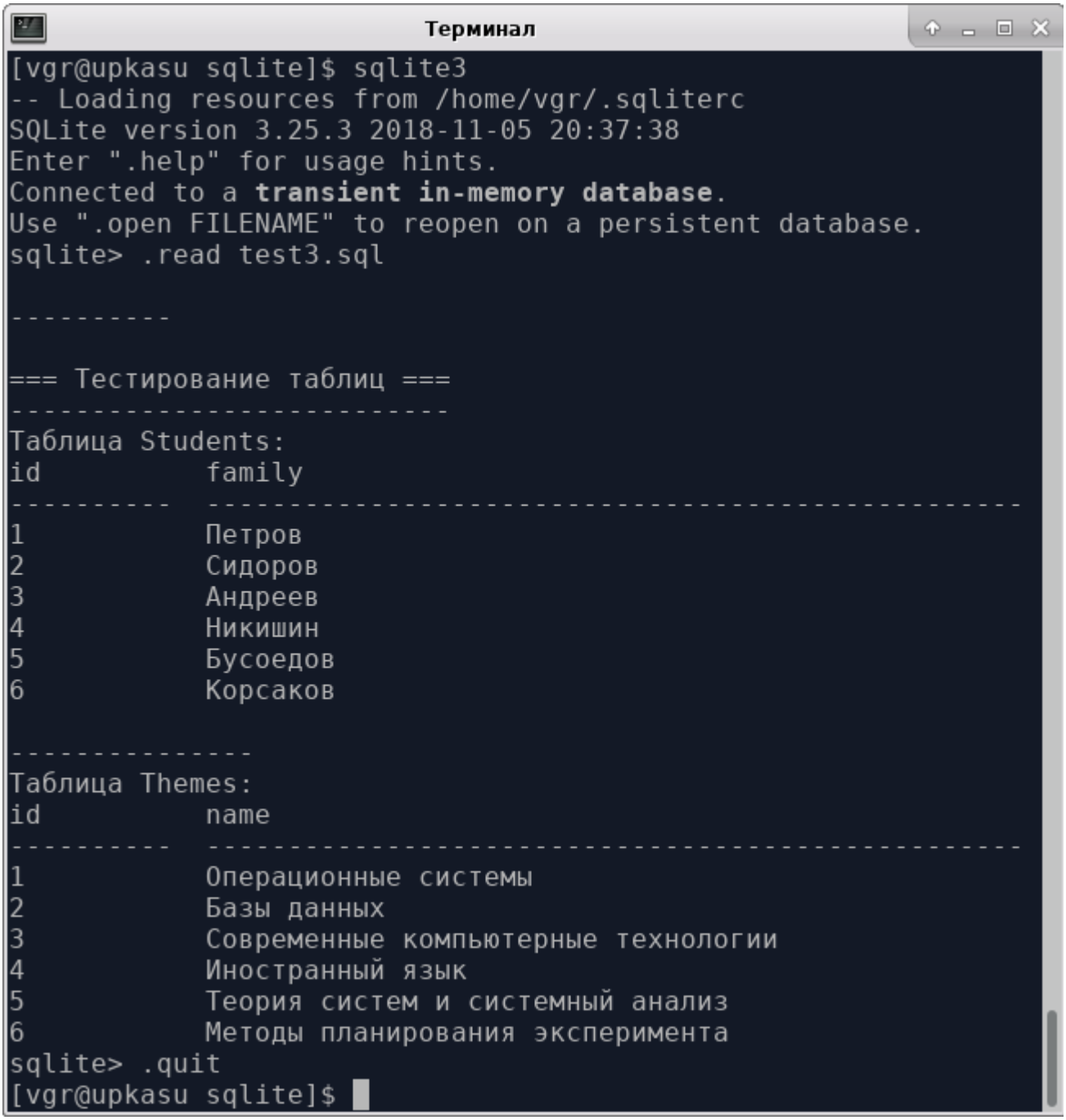

Рисунок 1.3 — Создание таблицы test3.db

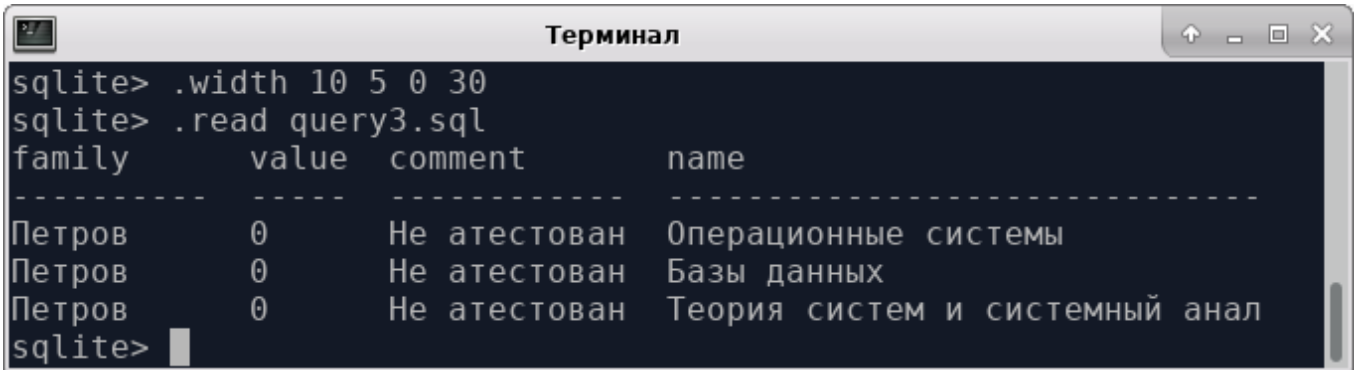

Рисунок 1.4 — Тестирование базы данных запросом query3.sql

Таким образом, мы создали нужную базу данных ***test3.db*** и прототип запроса, который следует реализовать с помощью программы на языке С.

Этап 3. Реализация программы ***test3_exams***.

Исходный текст программны ***test3_exams***, со всеми необходимыми комментариями по ее работе, представлен на листинге 1.4.

Листинг 1.4 — Исходный текст программы test3_exam

```
/*
 ============================================================================
 Name        : test3_exams.c
 Author      : Reznik V.G., 14.11.2018
 Version     :
 Copyright   : Your copyright notice
 Description : Запрос к базе данных test3.db, Ansi-style
 ============================================================================
 */

#include <stdio.h>
#include <stdlib.h>

#include <sqlite3.h>

/**   Функция обратного вызова из функции sqlite_exec():

      Здесь происходит обработка результата SQL запроса.
      Эта функция будет вызвана для каждой строки
      результата выполняемого запроса.
*/
static int callback(void *pArg, int coln, char **rows, char **colnm)
{
      /** Аргументы функции callback():
       *
       * pArg – указатель на произвольную информацию, передаваемую в callback()
       * при вызове ее из функции sqlite_exec(); здесь используется для подсчета
       * числа обработанных строк;
       *
       * coln  - число столбцов в запрашиваемой таблице;
       * rows  - массив значений в записи;
       * colnm - массив имен столбцов;
      */
      int i;
      static int b = 1;
      int *kk;
      kk = (int *)pArg;
      *kk = *kk + 1;
      // Печаем названия столбцов
      if (b) // Печатаем только один раз
```

```
	{
		for(i=0; i<coln; i++)
			printf("%s\t", colnm[i]);
		printf("\n");
		b = 0;
		puts("--------------------------------------------------");
	}
	/**
	 * Печать значений строки БД
	 */
	for(i=0; i<coln; i++)
		printf("%s\t", rows[i]);
	printf("\n");
	return 0;
} // end callback

/**
 * Главная функция вызова.
 */
int main(int argc, char *argv[])
{
	int rc;
	sqlite3 *db;	// Указатель на открытую базу данных.
	char * errmsg;	// Сообщение об ошибке.
	int k = 0;	// Количество строк в ответе на запрос.

	if(argc != 2){
		puts("Используй запуск: test3_exams ФИО_студента");
		exit(1);
	}
	printf("Оценки студента: %s\n", argv[1]);
	puts("--------------------------------------------------");

	/**
	 * SQL запрос к базе данных test3.db
	 */
	char sql[BUFSIZ]; // Буфер запроса, отсылаемого к БД.
	char const * sform = "SELECT Exams.value, Exams.comment, Themes.name "
			"FROM Students, Exams, Themes "
			"WHERE	Students.family = '%s' AND "
			"Students.id	= Exams.id_st AND "
			"Exams.id_th	= Themes.id ;";

	sprintf(sql, sform, argv[1]);
	//puts(sql);

	/**
	 * Открываю созданную БД.
	 * У аспиранта путь к БД будет наверно другой!!!
	 */
	rc = sqlite3_open("/home/vgr/sqlite/test3.db", &db);
	if (rc)
```

```
{
      // Если ошибка при открытии БД
      errmsg = (char*) sqlite3_errmsg(db);
      printf("%s\n", errmsg);
      sqlite3_close(db);
      exit(1);
}
/**
 * Выполнение SQL запроса...
 * Доступ к базе данных осуществляется с помощью
 * функций sqlite3_exec():
 *
 * int sqlite3_exec (
            //    Описатель базы данных полученный из функции
            //    sqlite_open().

            sqlite * db,

            //    Строка SQL запроса, возможно определить
            //    несколько запросов в одной строке.

            const char * sql,

            //    Указатель на функцию повторного вызова,
            //    которая будет вызвана для каждой строки
            //    результата выполнения запроса SELECT.
            //    Можно указать пустое значение NULL.

            int (*xCallback) (void*, int, char **, char **),

            //    Значение переданное функции повторного
            //    вызова, в качестве ее первого аргумента.

            void * pArg,

            //    Строка сообщения об ошибке.

            char ** errmsg
   );
 */
rc = sqlite3_exec(db, sql, callback, &k, &errmsg);
if (rc != SQLITE_OK)
{
      // Если ошибка при выполнении запроса
      printf("%s\n", errmsg);
      sqlite3_close(db);
      exit(1);
}

puts("----------------------------------");
printf("Обработано %i строк(и)...\n", k);
/**
```

```
        * Закрытие БД
        */
       sqlite3_close(db);

} // end main
```

Конец листинга 1.4

Замечание

Для создания исполняемой программы, работающей с базой данных SQLite, необходимо подключить библиотеку libsqlite3.so (libsqlite3.dll). Кроме того, при обращении к базе данных необходимо *правильно указать путь до нее!!!*

На рисунке 1.5 приведены варианты запуска программы ***test3_exams***.

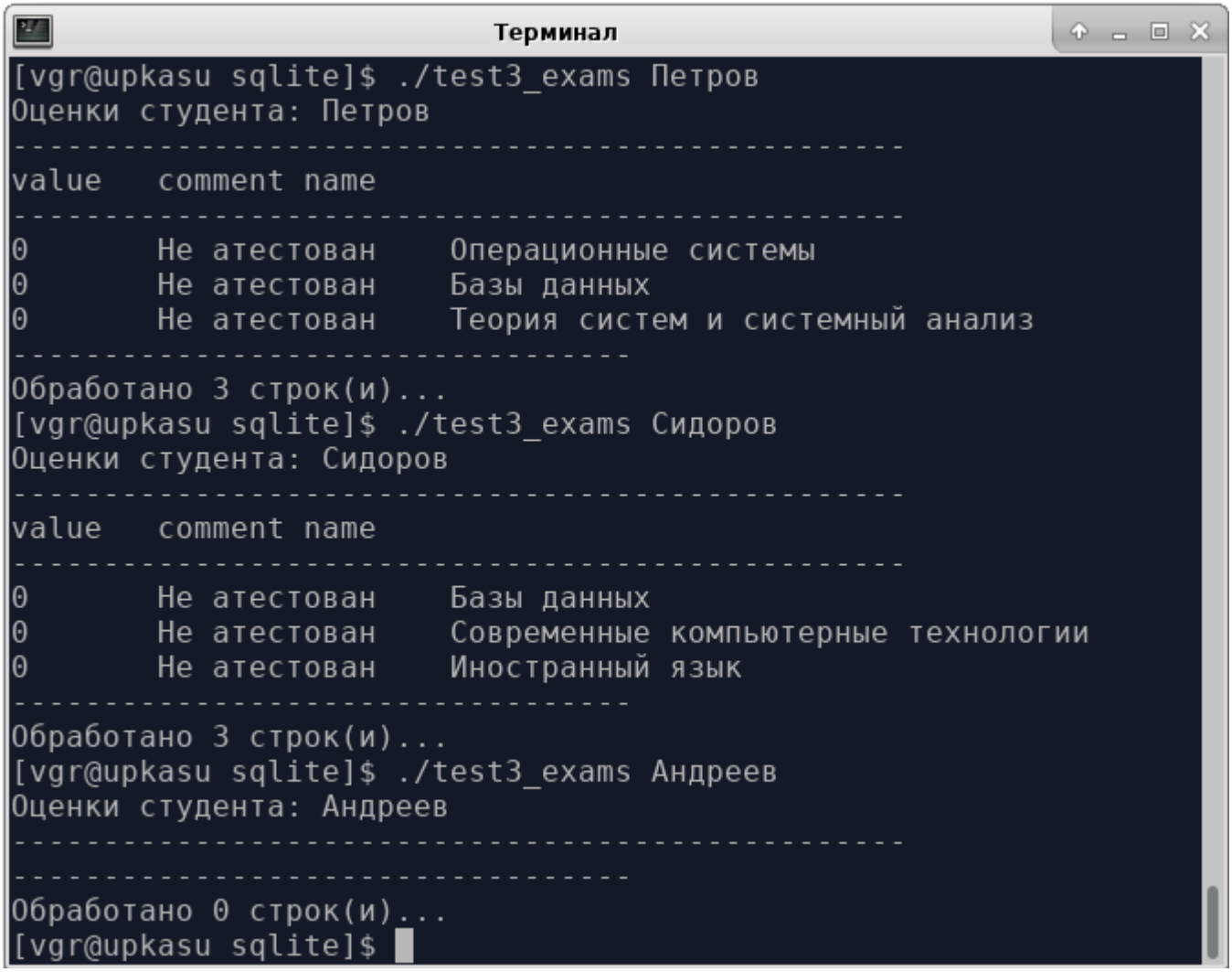

Рисунок 1.5 — Результаты запуска программы test3_exams

1.3.2.3 Учебные задания

Аспиранту следует самостоятельно придумать проект, в котором он мог бы применить СУБД SQLite.

Далее, необходимо провести разработку наподобие той, что представлена в пункте 1.3.2.2, и отразить результаты работы в личном отчете.

Замечание

Выбранный аспирантом индивидуальный проект не должен быть слишком сложным — не более 4-х таблиц.

При разработке проекта следует учесть, что:

- преимущество использования СУБД состоит в том, что само создание базы данных проходит синтаксическую проверку, поэтому значительно уменьшает число возможных логических ошибок при ее использовании;
- при создании и модификации базы данных может быть задействован достаточно мощный инструмент утилиты sqlite3, обеспечивающий контроль правильности проведения операций и вывод сообщений об ошибках;
- работа с базой данных может выполняться и из сценария Shell (см. листинг 1.5).

Если аспирант желает реализовать программы модифицирующие данные, то рекомендуется воспользоваться примерами, реализованными в [1.3.15].

Листинг 1.3 — Файл сценария test3.sh

```
#!/bin/bash

echo ==============================
sqlite3 <<EOF

select * from Students;
.quit

EOF
echo ==============================
```

Конец листинга 1.5

1.3.3 Список использованных источников

1.3.1 Oracle — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/Oracle). Проверено: 11.11.2018.

1.3.2 Microsoft Microsoft SQL Server — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/Microsoft_SQL_Server). Проверено: 11.11.2018.

1.3.3 Apache HTTP Server — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/Apache_HTTP_Server). Проверено: 11.11.2018.

1.3.4 MySQL — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/MySQL). Проверено: 11.11.2018.

1.3.5 ISO/IEC TR 10032-2003 - Information technology -- Reference Model of Data Management: [Электронный документ]. - (https://www.iso.org/standard/38607.html). Проверено: 11.11.2018.

1.3.6 PostgreSQL — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/PostgreSQL). Проверено: 11.11.2018.

1.3.7 LXF85:PostgreSQL — Linuxformat: [Электронный документ]. - (http://wiki.linuxformat.ru/wiki/LXF85:PostgreSQL). Проверено: 11.11.2018.

1.3.8 PostgreSQL: вчера, сегодня, завтра | Открытые системы. СУБД: [Электронный документ]. - (https://www.osp.ru/os/2015/03/13046900/). Проверено: 11.11.2018.

1.3.9 Firebird — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/Firebird). Проверено: 11.11.2018.

1.3.10 Apache Derby — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/Apache_Derby). Проверено: 11.11.2018.

1.3.11 HSQLDB — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/HSQLDB). Проверено: 11.11.2018.

1.3.12 Sedna — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/Sedna). Проверено: 11.11.2018.

1.3.13 SQLite — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/SQLite). Проверено: 11.11.2018.

1.3.14 SQLite Tutorial: [Электронный документ]. - (http://www.tutorialspoint.com/sqlite/). Проверено: 11.11.2018.

1.3.15 Двигатель базы данных "SQLite": [Электронный документ]. - (http://www.codenet.ru/db/other/sqlite/). Проверено: 11.11.2018.

2 Сетевые технологии

В данном разделе рассматриваются:

- сетевые концепции на основе технологии протокола SCTP;
- технологии Internet на примере протокола WebSocket;
- технологии мультимедиа на примере векторной графики в формата SVG.

2.1 Практическая работа №4 «Основные сетевые концепции»

Цель работы: Освоение технологии сокетов на примере усовершенствованного протокола SCTP.

2.1.1 Самостоятельная работа

Основные сетевые концепции. Глобальные, территориальные и локальные сети. Проблемы стандартизации. Сетевая модель OSI. Модели взаимодействия компьютеров в сети. Среда передачи данных. Преобразование сообщений в электрические сигналы, их виды и параметры. Проводные и беспроводные каналы передачи данных. Локальные сети. Протоколы, базовые схемы пакетов сообщений и топологии локальных сетей. Сетевое оборудование ЛВС. Глобальные сети. Основные понятия и определения. Сети с коммутацией пакетов и ячеек, схемотехника и протоколы. Принципы межсетевого взаимодействия и организации пользовательского доступа. Методы и средства защиты информации в сетях. Базовые технологии безопасности. Сетевые операционные системы. Архитектура сетевой операционной системы: сетевые оболочки и встроенные средства. Обзор и сравнительный анализ популярных семейств сетевых ОС.

Литература: [10, 11].

2.1.2 Порядок выполнения работы

Основы сетевых технологий аспирант начал изучать на третьем курсе бакалавриата в дисциплине «Сети и телекоммуникации». В более широком аспекте эти технологии изучены в дисциплине «Современные компьютерные технологии» (см. [2], раздел 7 «Технологии взаимодействия открытых систем»).

В общем случае, все сетевые технологии предполагают условное разделение программного обеспечения на две большие категории: серверное ПО и клиентское ПО, реализуя тем самым технологию «клиент-сервер».

Более конкретно, сетевые технологии уточняются набором используемых протоколов и спецификой их применения в разрабатываемых приложениях. В этом

аспекте достаточно четко выделяются системный и прикладной уровни, которые предполагают, что:

- системный уровень сетевого ПО реализован как часть программного обеспечения используемой ОС;
- прикладной уровень сетевого ПО обеспечивает надежный транспорт данных.

2.1.2.1 История вопроса

Сетевые технологии имеют не только достаточно длинную историю, но и множество реализованных и успешно работающих методов. Общие вопросы можно изучить по источнику [10], а более профессиональные вопросы уже стандартизированы. Основная концептуальная идея сформулирована в модели OSI (Open Systems Interconnection), что отражено в серии соответствующих ГОСТов, начиная с [11].

Применительно к программному обеспечению, сетевые технологии опираются на технологию сокетов [2.1.1]: «**Со́кет** (англ. *socket* — разъём) — название программного интерфейса для обеспечения обмена данными между процессами. Процессы при таком обмене могут исполняться как на одной ЭВМ, так и на различных ЭВМ, связанных между собой сетью. Сокет — абстрактный объект, представляющий конечную точку соединения.

Следует различать **клиентские** и **серверные сокеты**. Клиентские сокеты грубо можно сравнить с конечными аппаратами телефонной сети, а серверные — с коммутаторами. Клиентское приложение (например, браузер) использует только клиентские сокеты, а серверное (например, веб-сервер, которому браузер посылает запросы) — как клиентские, так и серверные сокеты.

Интерфейс сокетов впервые появился в BSD Unix. Программный интерфейс сокетов описан в стандарте POSIX.1 и в той или иной мере поддерживается *всеми* современными операционными системами. ...».

В практике сетевого программирования, наиболшее распространение получили «Сокеты Беркли» [2.1.2]: «**Сокеты Беркли** — интерфейс программирования приложений (API), представляющий собой библиотеку для разработки приложений на языке C с поддержкой межпроцессного взаимодействия (IPC), часто применяемый в компьютерных сетях.

Сокеты Беркли (также известные как API сокетов BSD) впервые появились как API в операционной системе 4.1BSD Unix (выпущенной в 1982 году). Тем не менее, только в 1989 году Калифорнийский университет в Беркли смог начать выпускать версии операционной системы и сетевой библиотеки без лицензионных ограничений AT&T, действующих в защищённой авторским правом Unix.

API сокетов Беркли сформировал фактически стандарт абстракции для сетевых сокетов. Большинство прочих языков программирования используют интерфейс, схожий с API языка Си.

API Интерфейса транспортного уровня (TLI), основанный на STREAMS, представляет собой альтернативу сокетному API. Тем не менее, API сокетов Беркли значительно преобладает в популярности и количестве реализаций. ...».

Сам интерфейс сокетов настолько популярен, что широко используется как интерфейс взаимодействия внутри отдельной ЭВМ [2.1.3]: «**Сокет домена Unix** (англ. *Unix domain socket*, UDS) или **IPC-сокет** (сокет межпроцессного взаимодействия) — конечная точка обмена данными, подобная Интернет-сокету, но не использующая сетевого протокола для взаимодействия (обмена данными). Используется в операционных системах, поддерживающих стандарт POSIX, для межпроцессного взаимодействия. Корректным термином стандарта POSIX является **POSIX Local IPC Sockets**. Подобно TCP-сокетам, эти сокеты поддерживают надёжную потоковую передачу (макрос SOCK_STREAM). Также они могут работать в режимах передачи дайтаграмм: упорядоченной и надёжной передачи (SOCK_SEQPACKET) или неупорядоченной и ненадёжной (SOCK_DGRAM). Подробное описание Unix сокетов и API содержится в странице man с названием unix из раздела 7. ...».

В целом, популярность технологии сокетов обусловлена не столько особыми ее преимуществами перед технологией «Интерфейса транспортного уровня (TLI)», который является частью модели OSI, сколько традицией программирования на языке C, рассматривающей сокет как разновидность файла со многими типичными для файловой системы ОС UNIX операциями, например, open(), close(), read(), write() и другими.

Другим базовым понятием, применительно к программному обеспечению сетевых технологий, является протокол, с помощью которого клиенты и сервера обмениваются между собой данными. Технология сокетов предоставляет достаточно простые и стандартизированные средства доступа к транспортному уровню наиболее популярному в настоящее время стеку протоколов TCP/IP [2.1.4]: «**TCP/IP** — сетевая модель передачи данных, представленных в цифровом виде. Модель описывает способ передачи данных от источника информации к получателю. В модели предполагается прохождение информации через четыре уровня, каждый из которых описывается правилом (протоколом передачи). Наборы правил, решающих задачу по передаче данных, составляют стек протоколов передачи данных, на которых базируется Интернет. Название TCP/IP происходит из двух важнейших протоколов семейства — Transmission Control Protocol (TCP) и Internet Protocol (IP), которые первыми были разработаны и описаны в данном стандарте. Также изредка упоминается как модель DOD (Department of Defense) в связи с историческим происхождением от сети ARPANET из 1970-х годов (под управлением DARPA, Министерства обороны США ...».

Хотя стек протоколов TCP/IP охватывает только четыре уровня, из семи уровней модели OSI, этого достаточно для реализации огромного количества сетевых приложений, эффективно использующих ресурсы ЭВМ и операционных систем. Это подтверждается неугасающим интересом к усовершенствованию как модели всего стека, например, разработкой и внедрением протокола IPv6, так и ряда отдельных модификаций, например, усовершенствование протокола TCP в виде SCTP (Stream Control Transmission Protocol) [2.1.5].

Как отмечено в [2.1.6]: «**SCTP** (англ. *Stream Control Transmission Protocol* — «протокол передачи с управлением потоком») — протокол транспортного уровня в

компьютерных сетях, появившийся в 2000 году в IETF. RFC 4960 описывает этот протокол, а RFC 3286 содержит техническое вступление к нему.

Как и любой другой протокол передачи данных транспортного уровня, SCTP работает аналогично TCP или UDP. Будучи более новым протоколом, SCTP имеет несколько нововведений, таких как многопоточность, защита от DDoS атак, синхронное соединение между двумя хостами по двум и более независимым физическим каналам (multi-homing). ...».

Наиболее важным усовершенствованием является устранение уязвимостей протокола TCP, как это показано на рисунке 2.1 [2.1.6]: «Протокол TCP имеет потенциальную уязвимость, обусловленную тем, что нарушитель, устанавливая фальшивые IP-адреса отправителя, может послать серверу множество пакетов SYN. При получении пакета SYN сервер выделяет часть своих ресурсов для установления нового соединения. Обработка множества пакетов SYN рано или поздно затребует все ресурсы сервера и сделает невозможной обработку новых запросов. Такой вид атак называется «SYN-флуд» (SYN flood).

Протокол SCTP защищён от подобных атак с помощью механизма четырёхэтапного квитирования (four-way handshake) и вводом маркера (cookie). По протоколу SCTP клиент начинает процедуру установления соединения, посылая пакет INIT. В ответ сервер посылает пакет INIT-ACK, который содержит маркер (уникальный ключ, идентифицирующий новое соединение). Затем клиент отвечает посылкой пакета COOKIE-ECHO, в котором содержится маркер, полученный от сервера. Только после этого сервер выделяет свои ресурсы новому подключению и подтверждает это отправкой клиенту пакета COOKIE-ACK. ...».

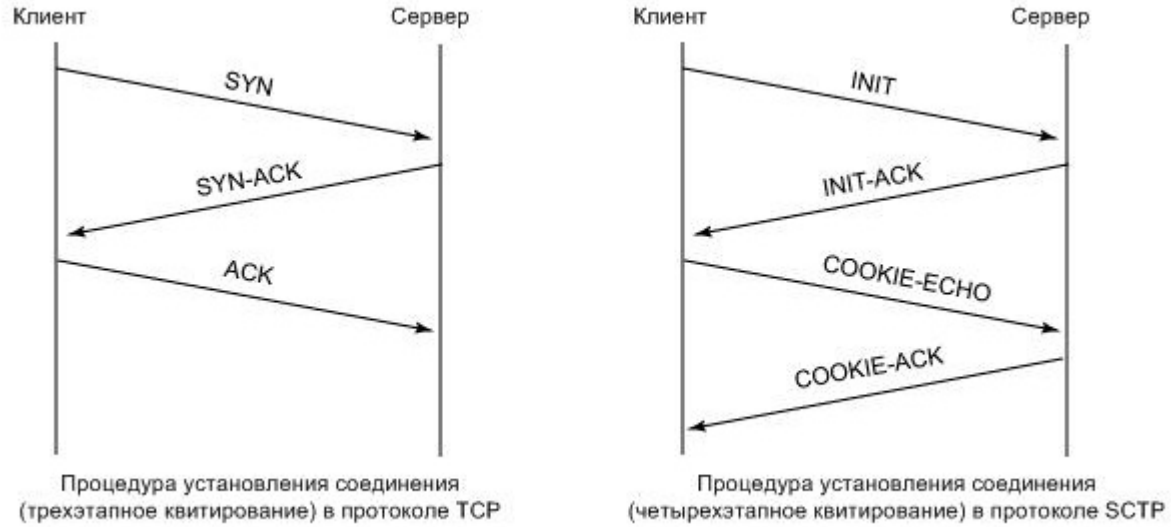

Рисунок 2.1 — Различия процедур установления соединения в протоколах TCP и SCTP (заимствовано из источника [2.1.6])

В таблице 2.1 представлены основные различия в свойствах протоколов UDP, TCP и SCTP.

Таблица 2.1 (заимствовано из источника [2.1.6])

Параметр	***UDP***	***TCP***	***SCTP***
Установка соединения	Нет	Да	Да
Надёжная передача	Нет	Да	Да
Сохранение границ сообщения	Да	Нет	Да
Упорядоченная доставка	Нет	Да	Да
Неупорядоченная доставка	Да	Нет	Да
Контрольные суммы данных	Да	Да	Да
Размер контрольной суммы (бит)	16	16	32
Путь MTU	Нет	Да	Да
Управление накоплением	Нет	Да	Да
Многопоточность	Нет	Нет	Да
Поддержка множественных интерфейсов	Нет	Нет	Да
Связка потоков	Нет	Да	Да

Одним из замечательных свойств протокола SCTP является разделение отдельных записей в потоке данных, как это показано на рисунке 2.2. Указанное свойство освобождает программиста от самостоятельной реализации алгоритмов разделения передаваемых и принимаемых пакетов на уровне своего приложения.

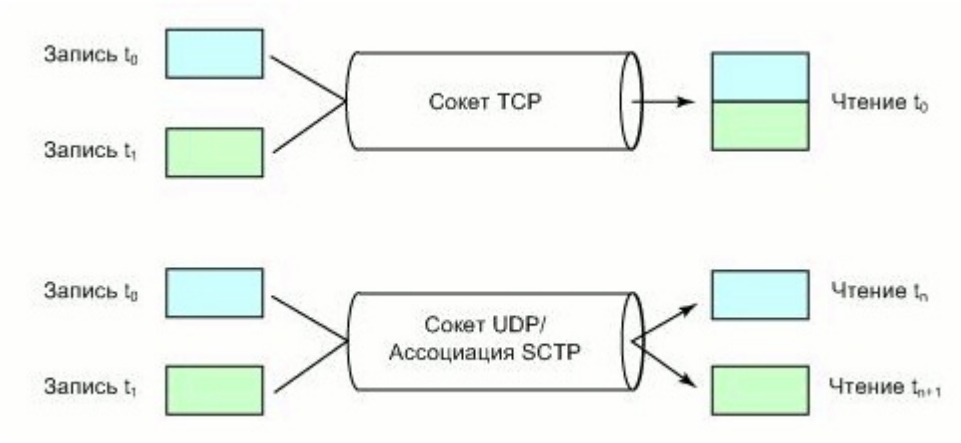

Рисунок 2.2 — Разделение записей в протоколе SCTP (заимствовано из источника [2.1.6])

Сказанное выше, является серьезным основанием для необходимости изучения аспирантом, технологии сетевого программирования с применением протокола SCTP.

2.1.2.2 Технология сокетов на примере протокола SCTP

Для практического освоения и совершенствования работы стехнологией

сокетов рекомендуется воспользоваться серией статей, которые начинаются источником «Интерфейс сетевого программирования Socket API, Часть 1: Создание собственного сервера:» [2.1.7].

Непосредственно для демонстрации работы протокола SCTP будет использован источник «Интерфейс сетевого программирования Socket API, Часть 5: SCTP» [2.1.8].

Дополнительно, как справочное руководство по работе с сокетами, рекомендуется публикация Федорчука В.Г. «TCP/IP Protocols / Федорчук В.Г., МФТИ им. Баумана, кафедра САПР» [2.1.9].

Традиционный пример, используемый в данной практической работе, демонстрирует взаимодействие клиента и сервера по протоколу SCTP:

- сервер запускается и ожидает соединение по порту 29008;
- когда приходит запрос на соединение от клиента, сервер посылает ему три сообщения по трем отдельным каналам, а затем ждет новое соединение;
- клиент соединяется с сервером по заданному в качестве аргумента адресу и порту 29008, а потом читает три посланных ему сообщения.

На листинге 2.1 представлен исходный код программы сервера.

Листинг 2.1 — Исходный текст программы sctp_server.c

```
/*
 ============================================================================
 Name        : sctp_server.c
 Author      : Reznik V.G., 24.11.2018
 Version     :
 Copyright   : Your copyright notice
 Description : Пример SCTP-сервера, Ansi-style
 ============================================================================

 Сервер ожидает соединения и передает сообщение клиенту
 по трем разным каналам.
 */

#include <stdio.h>             // Определены общие заголовки используемых функций.
#include <string.h>
#include <time.h>
#include <sys/types.h>
#include <unistd.h>

#include <sys/socket.h>  // Базовые функции сокетов BSD и структуры данных.
#include <netinet/in.h>  // Семейства адресов/протоколов PF_INET и PF_INET6.
#include <arpa/inet.h>   // Функции для работы с числовыми IP-адресами.

#include <netinet/sctp.h>// Дополнительные функции для протокола SCTP.
```

```
#define MAX_BUFFER  1024 // Размер используемого буфера сообщений.

int main()
{
    int sfd, cfd; // Дескрипторы сокетов.
    int len, i;

    struct sockaddr_in saddr; // Серверный адрес сокета, формируется сервером.
    struct sockaddr_in caddr; // Адрес сокета, формирующийся на основе
                              // запроса на соединение.

    struct sctp_initmsg initmsg; // См. /usr/include/netinet/sctp.h
    char buff[INET_ADDRSTRLEN];  // Рабочий буфер.

    char *msgs[] = {"Сообщение №1",       // Сообщение для канала 0
                    "Сообщение №2",       // Сообщение для канала 1
                    "Сообщение №3"};      // Сообщение для канала 2

    /**
     * Открытие сокета и получение его дескриптора.
     */
    sfd = socket( AF_INET, SOCK_STREAM, IPPROTO_SCTP );

    /**
     * Очищаем и заполняем адрес серверного сокета.
     */
    bzero( (void *)&saddr, sizeof(saddr) );
    saddr.sin_family = AF_INET;
    saddr.sin_addr.s_addr = htonl( INADDR_ANY );
    /**
     * Используем порт № 29008
     * Клиент должен делать соединение по этому же порту.
     */
    saddr.sin_port = htons(29008);

    /**
     * Привязываем адрес сервера к открытому сокету
     * (по его дескриптору).
     */
    bind( sfd, (struct sockaddr *)&saddr, sizeof(saddr) );

    /**
     * Очищаем и устанавливаем опции структуры sctp_initmsg,
     * определенной в /usr/include/netinet/sctp.h:
     *
     struct sctp_initmsg {
            __u16 sinit_num_ostreams;
            __u16 sinit_max_instreams;
            __u16 sinit_max_attempts;
            __u16 sinit_max_init_timeo;
       };
```

```
 */
memset( &initmsg, 0, sizeof(initmsg) );
initmsg.sinit_num_ostreams = 3; // Количество выходных потоков передачи.
initmsg.sinit_max_instreams = 3;// Количество входных  потоков приема сообщ.
initmsg.sinit_max_attempts = 2; // Максимальное количество делаемых попыток.
/**
 * Устанавливаем опции протокола с помощью функции:

  int setsockopt( int sockfd,         // Дескриптор сокета.
                  int level,          // Используемый протокол <sys/socets.h>.
                  int optname,        // Модификация протокола <sys/socets.h>.
                  const void *optval, // Адрес структуры struct sctp_initmsg.
                  socklen_t optlen);  // Длина структуры.
 */
setsockopt( sfd, IPPROTO_SCTP, SCTP_INITMSG,
        &initmsg, sizeof(initmsg) );

/**
 * Запускаем "слушатель" сервера с очередью
 * на 5 запросов на соединение.
 */
listen( sfd, 5 );

/**
 * В цикле обрабатываем каждое соединение.
 */
for(;;) {
    printf("\nСервер запустился и ожидает соединения...\n\n");

    len=sizeof(caddr);
    /**
     * Ожидаем запрос, порождая клиентский дескриптор
     * для обмена информацией с клиентом.
     */
    cfd=accept(sfd, (struct sockaddr *)&caddr, (socklen_t *)&len);

    printf("Connected to %s\n",
        inet_ntop(AF_INET, &caddr.sin_addr, buff, sizeof(buff)));

    /**
     * Передаем клиенту три сообщения по разным каналам.
     */
    for(i=0; i< 3; i++) {

        /**
         * Посылаем клиенту сообщения (по одному на каждый из трех каналов)
         * с помощью специальной функции (см. man):

         int sctp_sendmsg (int sd,          // Дескриптор сокета.
                           const void *msg, // Адрес передаваемого сообщения.
                           size_t len,      // Длина сообщения.
```

```
                                struct sockaddr *to,// Структура адреса назначения.
                                socklen_t tolen, // Длина структуры.
                                uint32_t ppid,   //
                                uint32_t flags,  //
                                uint16_t stream_no, // Номер потока (i).
                                uint32_t timetolive,// Время жизни канала.
                                uint32_t context);//
                     */
                    sctp_sendmsg( cfd, (void *)msgs[i], (size_t)strlen(msgs[i]),
                            NULL, 0, 0, 0, i /* stream */, 0, 0 );
                    printf("Послал клиенту по каналу %d: %s\n", i, msgs[i]);
                }

                /**
                 * Закрываем дескриптор клиентского сокема и
                 * переходим на обслуживание следующего запроса.
                 */
                close( cfd );
            }
            return 0;
}
```

Конец листинга 2.1

После компиляции программы сервера, она запускается в отдельном терминале, как показано на рисунке 2.3.

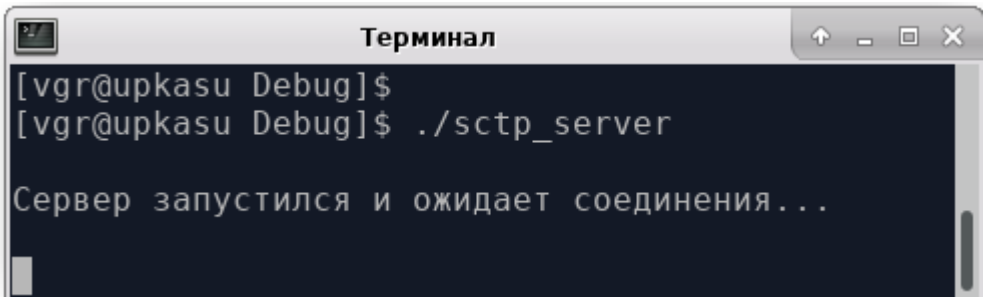

Рисунок 2.3 — Результат запуска программы SCTP-сервера

Замечание

Для создания программы SCTP-сервера требуется библиотека *libsctp.so - для ОС Linux* (или *libsctp.dll* - для ОС MS Windows).

Исходный код программы SCTP-клиента показан на листинге 2.2.

Листинг 2.2 — Исходный текст программы sctp_client.c

```
/*
 ============================================================================
```

```
 Name        : sctp_client.c
 Author      : Reznik V.G., 24.11.2018
 Version     :
 Copyright   : Your copyright notice
 Description : Пример SCTP-клиента, Ansi-style
 ============================================================================

 Клиент выполняет соединение с сервером по заданному адресу и порту 29008,
 а затем читает сообщения, полученные по трем обслуживаемым каналам.
 */

#include <stdio.h>       // Определены общие заголовки используемых функций.
#include <string.h>
#include <time.h>
#include <sys/types.h>
#include <unistd.h>

#include <sys/socket.h>  // Базовые функции сокетов BSD и структуры данных.
#include <netinet/in.h>  // Семейства адресов/протоколов PF_INET и PF_INET6.
#include <arpa/inet.h>   // Функции для работы с числовыми IP-адресами.

#include <netinet/sctp.h>// Дополнительные функции для протокола SCTP.

#define MAX_BUFFER  1024

int main(int argc, char **argv)
{
    int cfd; // Дескриптор сокета.
    int flags, i;

    struct sockaddr_in saddr;    // Серверный адрес сокета, формируется сервером.
    struct sctp_initmsg initmsg; // См. /usr/include/netinet/sctp.h

    /**
     * Структуры для обработки событий и получения информации
     * о принятых данных.
     * См. /usr/include/netinet/sctp.h
     */
    struct sctp_sndrcvinfo sndrcvinfo;
    struct sctp_event_subscribe events;

    char buffer[MAX_BUFFER+1];  // Буфер приема сообщений.

    if(argc!=2) {
        printf("Используй: %s ipaddress\n", argv[0]);
        return -1;
    }

    /**
     * Открытие сокета и получение его дескриптора.
     */
    cfd = socket( AF_INET, SOCK_STREAM, IPPROTO_SCTP );
```

```
/**
 * Заполнение опций, как и для программы-сервера.
 */
memset( &initmsg, 0, sizeof(initmsg) );
initmsg.sinit_num_ostreams = 3;
initmsg.sinit_max_instreams = 3;
initmsg.sinit_max_attempts = 2;
setsockopt( cfd, IPPROTO_SCTP, SCTP_INITMSG,
        &initmsg, sizeof(initmsg) );

/**
 * Формируем структуру адреса и порта соединения.
 */
bzero( (void *)&saddr, sizeof(saddr) );
saddr.sin_family = AF_INET;
inet_pton(AF_INET, argv[1], &saddr.sin_addr);
saddr.sin_port = htons(29008);

/**
 * Осуществляем само соединение.
 */
connect( cfd, (struct sockaddr *)&saddr, sizeof(saddr) );

/**
 * Формируем опции сокета для обработки событий.
 */
memset( (void *)&events, 0, sizeof(events) );
events.sctp_data_io_event = 1;
setsockopt( cfd, SOL_SCTP, SCTP_EVENTS,
    (const void *)&events, sizeof(events) );


/**
 * Обрабатываем прием только трех сообщений.
 */

for (i=0; i<3; i++) {
  /**
   * Чистим буфер приема сообщения.
   */
    bzero( (void *)&buffer, sizeof(buffer) );

    /**
     * Сообщения принимаются по одному
     * с помощью специальной функции (см. man):

        int sctp_recvmsg( int sd,     // Дескриптор сокета.
                          void * msg, // Адрес приемного буфера.
                          size_t len, // Длина приемного буфера.
                          struct sockaddr * from, // Структура адреса источника.
                          socklen_t * fromlen,    // Длина этой структуры.
```

```
                                struct sctp_sndrcvinfo * sinfo,// Адрес
информационной структуры.
                                int * msg_flags); // Указатель на флаги сообщения.
         */
        sctp_recvmsg( cfd, (void *)buffer, sizeof(buffer),
                        (struct sockaddr *)NULL, 0, &sndrcvinfo, &flags );

        /**
         * Печать принятого сообщения.
         */
        printf("Клиент принял данные по каналу %d: ", sndrcvinfo.sinfo_stream);

        if(strlen(buffer) > 0)
            printf("%s\n", buffer);
        else printf("Нет данных...\n");

    }

    /**
     * Закрываем сокет и завершаем работу клиента.
     */
    close(cfd);

    return 0;
}
```

Конец листинга 2.2

Замечание

Для создания программы SCTP-клиента также требуется библиотека *libsctp.so - для ОС Linux* (или *libsctp.dll - для* ОС MS Windows).

Программа SCTP-клиента запускается в отдельном терминале, как это показано на рисунке 2.4.

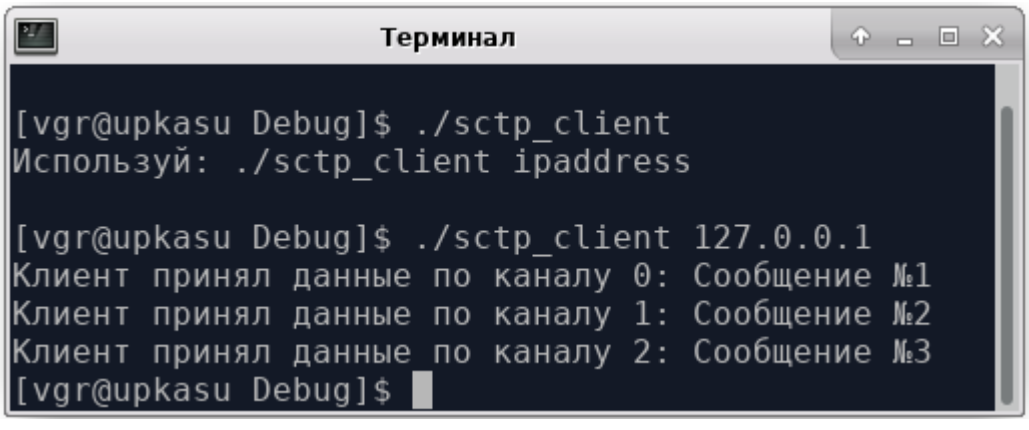

Рисунок 2.4 — Результат запуска программы SCTP-клиента

После получения (или попыток получения) трех сообщений, программа SCTP-клиента завершает свою работу, а программа SCTP-сервера продолжает работу, как это показано на рисунке 2.5.

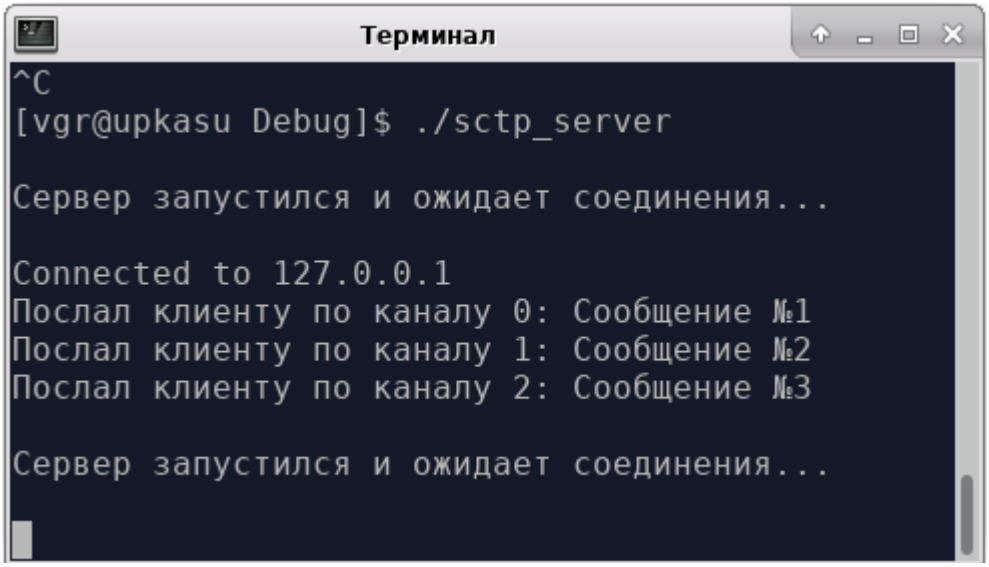

Рисунок 2.5 — Результат запуска программы SCTP-сервера

2.1.2.3 Учебные задания

Аспиранту следует разобраться, откомпилировать и исследовать работу программ SCTP клиента и сервера. Полученные результаты отразить в личном отчете. Затем, провести модификацию имеющихся программ и последующее их исследование.

Модификацию программ необходимо провести в следующих направлениях:

- изменение и дополнение прикладного назначения клиентской и серверной программ;
- дополнение алгоритма программ средствами обработки ошибок, возникающих на различных этапах взаимодействия клиента и сервера;
- изучение описания функций, представленных в файле `sctp.h`, с целью использования новых алгоритмов реализации клиентских и серверных программ.

По результатам исследования, в отчете отразить свое мнение о технологии sctp-сокетов, а также возможные направления их прикладного использования.

2.1.3 Список использованных источников

2.1.1 Сокет (программный интерфейс) — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/Сокет_(программный интерфейс)). Проверено: 18.11.2018.

2.1.2 Сокеты Беркли — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/Сокеты Беркли). Проверено: 18.11.2018.

2.1.3 Сокет домена UNIX — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/Сокет домена UNIX). Проверено: 18.11.2018.

2.1.4 TCP/IP — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/TCP/IP). Проверено: 18.11.2018.

2.1.5 RFC 4960 — Stream Control Transmission Protocol: [Электронный документ]. - (https://tools.ietf.org/html/rfc4960). Проверено: 18.11.2018.

2.1.6 SCTP — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/SCTP). Проверено: 18.11.2018.

2.1.7 Интерфейс сетевого программирования Socket API, Часть 1: Создание собственного сервера: [Электронный документ]. - (http://rus-linux.net/MyLDP/algol/sokets-API-1.html). Проверено: 18.11.2018.

2.1.8 Интерфейс сетевого программирования Socket API, Часть 5: SCTP: [Электронный документ]. - (http://rus-linux.net/MyLDP/algol/sokets-API-5.html). Проверено: 18.11.2018.

2.1.9 TCP/IP Protocols / Федорчук В.Г., МФТИ им. Баумана, кафедра САПР: [Электронный документ]. - (http://www.opennet.ru/docs/RUS/tcpip/). Проверено: 18.11.2018.

2.2 Практическая работа №5 «Технологии Internet»

Цель работы: Освоение технологии Internet на примере протокола WebSocket.

2.2.1 Самостоятельная работа

Принципы функционирования Internet, типовые информационные объекты и ресурсы. Ключевые аспекты WWW-технологии. Адресация в сети Internet. Методы и средства поиска информации в Internet, информационно-поисковые системы. Языки и средства программирования Internet приложений. Язык гипертекстовой разметки HTML, основные конструкции, средства подготовки гипертекста (редакторы и конверторы). Базовые понятия VRML. Организация сценариев отображения и просмотра HTML документов с использованием объектно-ориентированных языков программирования.

Литература: [2, 10, 11].

2.2.2 Порядок выполнения работы

Основы технологий Internet аспирант начал изучать на третьем курсе бакалавриата в дисциплине «Сети и телекоммуникации». В более широком аспекте эти технологии изучены в дисциплине «Современные компьютерные технологии» (см. [2], раздел 8 «Сервисные технологии»).

Излишне напоминать, что технологии Internet опираются на сетевые технологии, рассмотренные в предыдущем практическом занятии, и на нижнем уровне используют стек протоколов TCP/IP. С другой стороны, в основе этой технологии лежит язык раметки гипертекста, что указывает на сильную связь с офисными технологиями, особенно в плане формирования электронных документов.

2.2.2.1 История вопроса

Обобщенное (суммарное) обозначение всех технологий Internet получило название «Всемирная паутина» [2.2.1]: «**Всеми́рная паути́на** (англ. *World Wide Web*) — распределенная система, предоставляющая доступ к связанным между собой документам, расположенным на различных компьютерах, подключённых к сети Интернет. Для обозначения Всемирной паутины также используют слово **веб** (англ. *web* «паутина») и аббревиатуру **WWW**.

Всемирную паутину образуют сотни миллионов веб-серверов. Большинство ресурсов Всемирной паутины основано на технологии гипертекста. Гипертекстовые документы, размещаемые во Всемирной паутине, называются веб-страницами. Несколько веб-страниц, объединённых общей темой, дизайном, а также связанных

между собой ссылками и обычно находящихся на одном и том же веб-сервере, называются веб-сайтом. Для загрузки и просмотра веб-страниц используются специальные программы — браузеры (англ. *browser*). ...».

Визитной карточной технологий Internet является понятие гипертекста [2.2.2]: «**Гипертекст** (англ. *hypertext*) — термин, обозначающий систему из текстовых страниц, имеющих перекрестные ссылки.

Примерами гипертекста являются энциклопедии, компьютерные сети, веб-сайты, в которых можно переходить с одной страницы на другую и выполнять поиск по ключевым словам.

В компьютерной терминологии гипертекст — это текст, сформированный с помощью языка разметки (например, HTML) с расчетом на использование гиперссылок. ...».

Что касается языка разметки HTML, он был разработан британским учёным Тимом Бернсом-Ли, ориентировочно в 1986-1991 годах в стенах организации ЦЕРН (г. Женева в Швейцарии) [2.2.3]: «**HTML** (от (англ. *HyperText Markup Language* — «язык гипертекстовой разметки разметки») — стандартизированный язык разметки документов во Всемирной паутине. Большинство веб-страниц содержат описание разметки на языке HTML (или XHTML). Язык HTML интерпретируется браузерами; полученный в результате интерпретации форматированный текст отображается на экране монитора компьютера или мобильного устройства.

Язык HTML до 5-й версии определялся как приложение SGML (стандартного обобщённого языка разметки по стандарту ISO 8879). Спецификации HTML5 формулируются в терминах DOM (объектной модели документа).

Язык XHTML является более строгим вариантом HTML, он следует синтаксису XML и является приложением языка XML в области разметки гипертекста.

Во всемирной паутине HTML-страницы, как правило, передаются браузерам от сервера по протоколам HTTP или HTTPS, в виде простого текста или с использованием шифрования. ...».

Первоначально ориентированные на представление статических документов, содержащих текст, растровые изображения и простейшие средства для представления форм, технологии Internet постоянно развивались в двух направлениях:

- направление развития *серверных средств* подготовки html-документов, первоначально определенные через интерфейс CGI;
- направление развития *клиентских средств* манипулирования отражением веб-страниц в браузерах.

Интерфейс CGI заложил основу протокола взаимодействия серверной и клиентской сторон программного обеспечения [2.2.4]: «**CGI** (от англ. *Common Gateway Interface* — «общий интерфейс шлюза») — стандарт интерфейса, используемого для связи внешней программы с веб-сервером. Программу, которая работает по такому интерфейсу совместно с веб-сервером, принято называть шлюзом, хотя многие предпочитают названия «скрипт» (сценарий) или «CGI-программа». По сути позволяет использовать консоль ввода и вывода для взаимодействия с

клиентом.

Сам интерфейс разработан таким образом, чтобы можно было использовать любой язык программирования, который может работать со стандартными устройствами ввода-вывода. Такими возможностями обладают даже скрипты для встроенных крмандных интерпретаторов операционных систем, поэтому в простых случаях могут использоваться даже командные скрипты. ...».

В начале 1995 года был разработан свободный сервер Apache, который дал интенсивный толчек серверным средствам подготовки html-документов [2.2.5]: «**Apache HTTP-сервер** (… назван именем группы племён североамериканских индейцев апачей; кроме того, является искажённым сокращением от англ. *a patchy server*; среди русских пользователей общепринято переводное *апа́ч*) — свободный веб-сервер.

Apache является кроссплатформенным ПО, поддерживает операционные системы Linux, BSD, Mac OS, Microsoft Windows, Novell NetWare, BeOS.

Основными достоинствами Apache считаются надёжность и гибкость конфигурации. Он позволяет подключать внешние модули для предоставления данных, использовать СУБД для аудентификации пользователей, модифицировать сообщения об ошибках и т.д. ...».

Именно этот сервер обеспечил интенсивное развитие и внедрение языка динамического формирования html-страниц на стороне сервера — PHP [2.2.6]: «**PHP** (англ. *PHP: Hypertext Preprocessor* — «PHP: препроцессор гипертекста»; первоначально *Personal Home Page Tools* — «Инструменты для создания персональных веб-страниц») — скриптовый язык общего назначения, интенсивно применяемый для разработки веб-приложений. В настоящее время поддерживается подавляющим большинством хостинг-провайдеров и является одним из лидеров среди языков, применяющихся для создания динамических веб-сайтов. ...».

Что казается стороны клиентов, то здесь особую популярность получил язык JavaScript [2.2.7]: «**JavaScript** (аббр. **JS**) — мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. Является реализацией языка ECMAScript (стандарт ECMA-262).

JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам. ...».

С момента своего появления, технологии Internet всегда претендовали на нечто большее, чем просто публичное распрострпнение документов, представленных в гипертексте. Обладая достаточно универсальными средствами отображения информации на стороне клиента (браузерами), технологии Internet обеспечивали парадигму «тонкого клиента» [2.2.8]: «**Тонкий клиент** (англ. *thin client*) в компьютерных технологиях — компьютер или программа-клиент в сетях с клиент-серверной или терминальной архитектурой, который переносит все или большую часть задач по обработке информации на сервер. Примером тонкого клиента может служить компьютер с браузером, использующийся для работы с веб-приложениями. Данным термином может также называться P2P-клиент, использующий в качестве

сервера другие узлы сети. ...».

Серьезным заказчиком на «тонкий клиент» является промышленное производство, но технологии Internet, на уровне обработки информации языками сценариев, не удовлетворяют многим его требованиям: быстродействию, надежности, защищенности, стоимости и другим.

В 1997 году корпорация Sun Microsystems кадинально изменила ситуацию предложив технологию *сервлетов* и *апплетов* на основе языка Java.

Сервлеты Java кардинально изменили качество работы серверов приложений [2.2.9]: «**Сервлет** является интерфейсом Java, реализация которого расширяет функциональные возможности сервера. Сервлет взаимодействует с клиентами посредством принципа запрос-ответ.

Хотя сервлеты могут обслуживать любые запросы, они обычно используются для расширения веб-серверов. Для таких приложений технология Java Servlet определяет HTTP-специфичные сервлет классы. ...». Кроме того, являясь компилируемым языком, Java освобождает серверы от необходимости проводить синтаксический анализ программного обеспечения сервера, ответственного за формирование html-страницы для клиента. Также обеспечивается кэширование повторяющихся запросов, что невозможно при использовании языка PHP.

В целом, технология Java сервлетов реализована в проекте Apache Tomcat [2.2.10]: «**Tomcat** (в старых версиях — **Catalina**) — контейнер сервлетов с открытым исходным кодом, разрабатываемый Apache Software Foundation. Реализует спецификацию сервлетов, спецификацию JavaServer Pages (JSP) и JavaServer Face (JSF). Написан на языке Java.

Tomcat позволяет запускать веб-приложения, содержит ряд программ для самоконфигурирования.

Tomcat используется в качестве самостоятельного веб-сервера, в качестве сервера контента в сочетании с веб-сервером Apache HTTP Server, а также в качестве контейнера сервлетов в серверах приложений JBoss и GlassFish. ...».

Для реализации «тонкого клиента» компания Sun Microsystems предложила Java-апплеты [2.2.11]: «**Java-апплет** — прикладная программа, чаще всего написан-ная на языке программирования Java в форме байт-кода. Java-апплеты выполняются в веб-обозревателе с использованием виртуальной Java машины (JVM), или в Sun's AppletViewer, автономном средстве для испытания апплетов. Java-апплеты были внедрены в первой версии языка Java в 1995 году. Java-апплеты обычно пишутся на языке программирования Java, но могут быть написаны и на других языках, которые компилируются в байт-код Java, таких, как Jython.

Апплеты используются для предоставления интерактивных возможностей веб-приложений, которые не могут быть предоставлены HTML. Так как байт-код Java платформо-независим, то Java-апплеты могут выполняться с помощью плагинов браузерами многих платформ, включая Microsoft Windows, UNIX, Apple Mac OS и GNU Linux. Такие программы с открытым исходным кодом, как applet2app, могут быть использованы для преобразования апплета в самостоятельные программы на Java или исполняемые файлы Linux и Windows. ...».

Как показала практика, получив полезный первоначальный успех, апплеты стали вытесняться скриптовыми языками, такими как JavaScript, и сейчас используются очень редко, хотя поддержка их в браузерах является обязательной.

Начиная с 2005 года, на стороне клиента особую популярность получила технология AJAX [2.2.12]: «**AJAX**, Ajax (от англ. *Asynchronous Javascript and XML* — «асинхронный JavaScript и XML») — подход к построению интерактивных пользовательских интерфейсов веб-приложений, заключающийся в «фоновом» обмене данными браузера с веб-сервером. В результате, при обновлении данных веб-страница не перезагружается полностью, и веб-приложения становятся быстрее и удобнее. По-русски иногда произносится по существующей аналогии у имени Ajax — Аякс, но у аббревиатуры AJAX нет устоявшегося варианта на кириллице. ...». На рисунке 2.6 схематически представлено сравнение классической технологии HTTP и ее модификации с помощью AJAX.

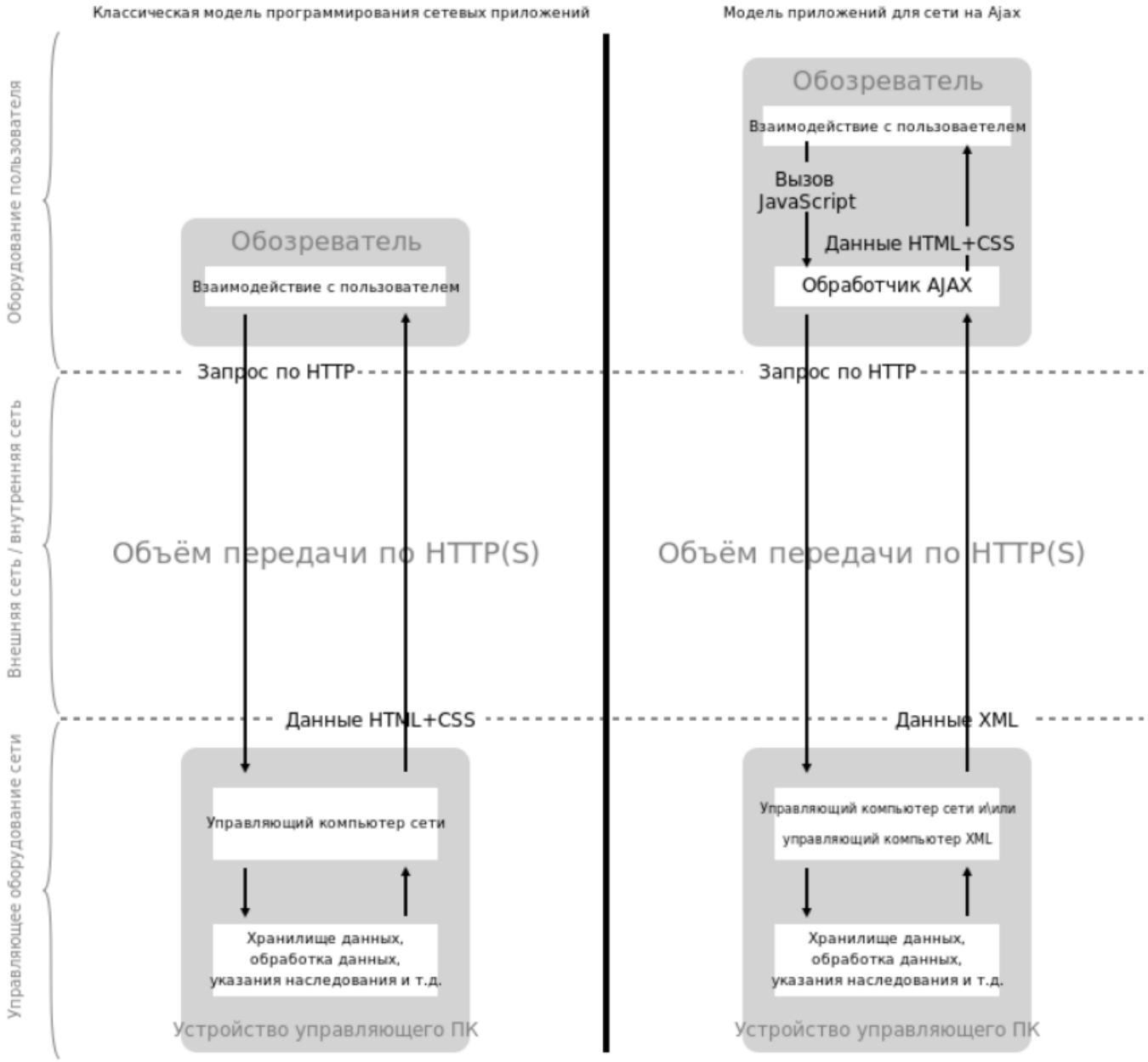

Рисунок 2.6 — Технологии HTTP и AJAX (заимствовано из источника [2.2.12])

Хорошо видно, что современные браузеры, построенные по объектно-ориентированной технологии, имеют специальный «Обработчик AJAX», который способен самостоятельно устанавливать с серверами соединения, скачивать с них данные, обычно в формате XML, и использовать эти данные для изменения уже загруженной ранее html-станицы.

Преимущества такого подхода состоят в том, что:

- нет необходимости, при каждом изменении данных, обращаться к веб-серверу за новой страницей, которую тот еще должен и построить;
- нет необходимости заново перерисовывать всю страницу.

В качестве недостатков технологии AJAX следует отметить сильное изменение самой html-страницы, в которой язык HTML начинает играть второстепенную роль:

- необходимость управления данными на стороне клиента требует организации специальных управляющих структур, которые реализуются вставками скриптовых сценариев в саму страницу;
- необходимость обеспечения коммуникаций с серверами, хранение и обработка данных порождает создание специальных библиотек функций на языке JavaScript, которые хотя и хранятся на сервере, но загружаются вместе с загрузкой html-страниц и расширяют языковый набор средств разметки самой сраницы.

В современных технологиях Internet такие технологии как AJAX занимают ведущее место, в частности, в современном языке разметки HTML5 [2.2.13], но несмотря на различные усовершенствования, AJAX обеспечивает только классическое синхронное взаимодействие клиента и сервера: при необходимости, «Обработчик AJAX» устанавливает соединение с сервером, делает запрос, ждет подготовку данных, скачивает данные и разрывает соединение с сервером.

В декабре 2011 года был опубликован новый асинхронный протокол взаимодействия клиента и сервера, названный WebSocket [2.2.14]. Википедия дает ему такое описание [2.2.15]: «**WebSocket** — протокол связи поверх TCP-соединения, предназначенный для обмена сообщениями между браузером и веб-сервером в режиме реального времени.

В настоящее время в W3C осуществляется стандартизация API Web Sockets. Черновой вариант стандарта этого протокола утверждён IETF.

WebSocket разработан для воплощения в веб-браузерах и веб-серверах, но он может быть использован для любого клиентского или серверного приложения. Протокол WebSocket — это независимый протокол, основанный на протоколе TCP. Он делает возможным более тесное взаимодействие между браузером и веб-сайтом, способствуя распространению интерактивного содержимого и созданию приложений реального времени. ...».

Учитывая большое количество вариантов реализации систем построенных на базе протокола WebSocket, рассморим пример, реализующий эту технологию на языке Java.

2.2.2.2 Технологии Internet на примере протокола WebSocket

Возможности технологии WebSocket будет проведено на примере использования Apache Tomcat [2.2.10]. В качестве программной реализации воспользуемся простейшим примером реализованным в руководстве «Apache Tomcat WebSocket Tutorial» [2.2.16] и модифицированном для целей обучения, в основном в плане коментариев.

Замечание

Публикация [2.2.16] демонстрирует пример в среде ОС Microsoft Windows.
Аспиранту рекомендуется использовать уже известную ему обучающую среду кафедры АСУ, описанную в руководстве [5], и методическое руководство, использованное им при выполнении практических работ [3].

В соотвествии с основной идеей технологии WebSocket, хорошо описанной в [2.2.5], на стороне клиента и сервера необходимо реализовать четыре метода:

- метод реакции на установление соединения;
- метод реакции на закрытие соединения;
- метод реакции на передачу серверу сообщения клиента;
- метод реакции на ошибку.

Проведем все этапы реализации простого примера, разделив основные этапы реализации на условные шаги.

Шаг 1. Запустим среду разработки Eclipse EE и создадим проект типа «Dynamic Web Project» с именем WsServer, как показано на рисунке 2.7.

Далее, завершим создание проекта нажатием на кнопку «Finish».

Шаг 2. Создадим в проекте WsServer java-класс, который будет ослуживать WebSocket на стороне сервера. Назовем этот класс именем WsSocket и обеспечим его настройками как показано на рисунке 2.8.

Нажатием на кнопки «Finish» завершим настройку класса и перейдем к созданию и редактированию исходного теста WsSocket.java.

Шаг 3. Создание исходного текста программы (java-класса) WsSocket.java.

Исходный текст представляет реализацию java-класса WsSocket, который имеет четыре стандартных метода для обслуживания соединения по интерфейсу WebSocket.

Содержимое программы представлено на листинге 2.3.

Комментарии к тесту листинга дают полное представление о работе методов этого класса.

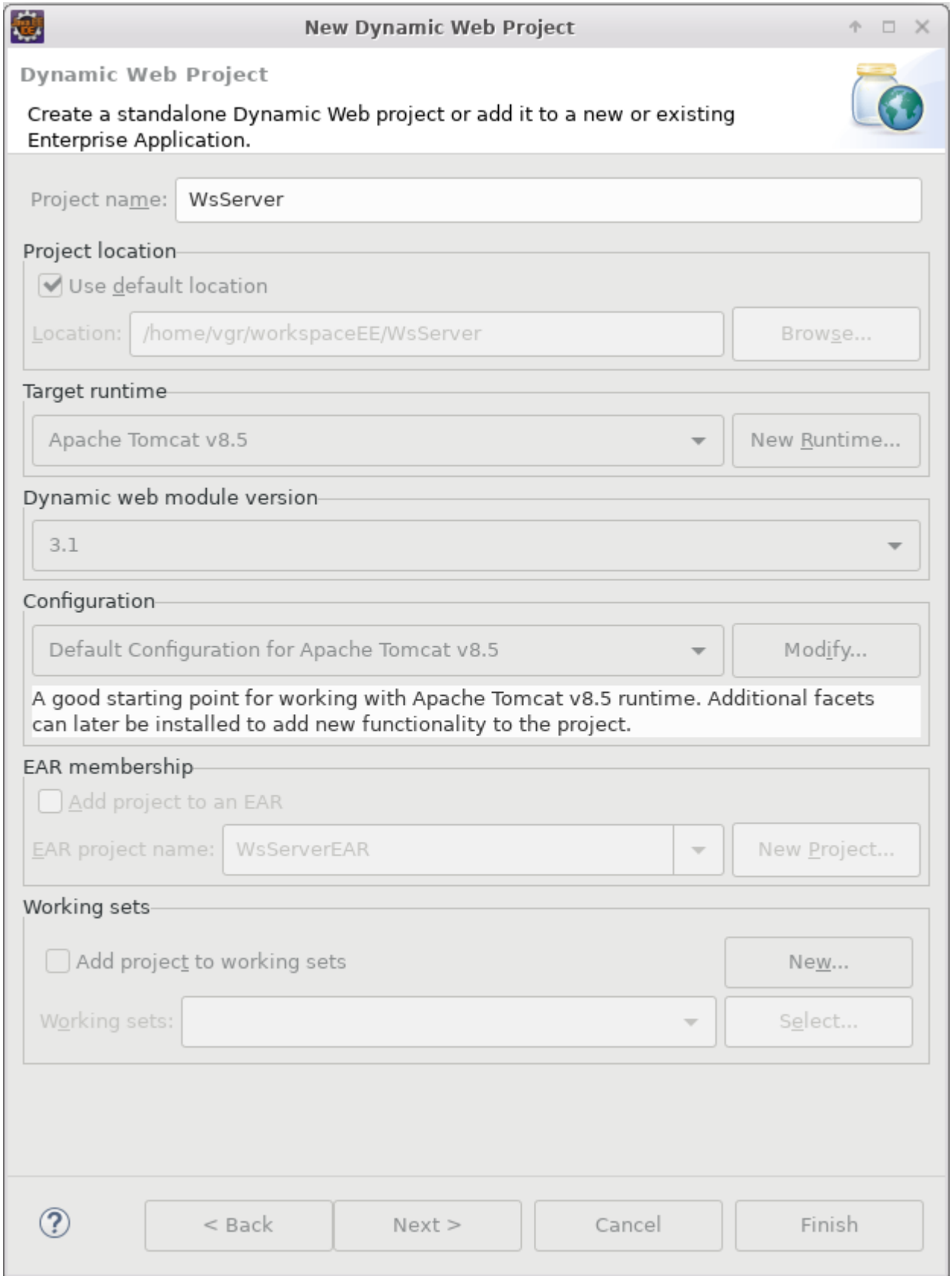

Рисунок 2.7 — Основные настройки среды разработки Eclipse EE при создании проекта типа Dynamic Web Project с именем WsServer

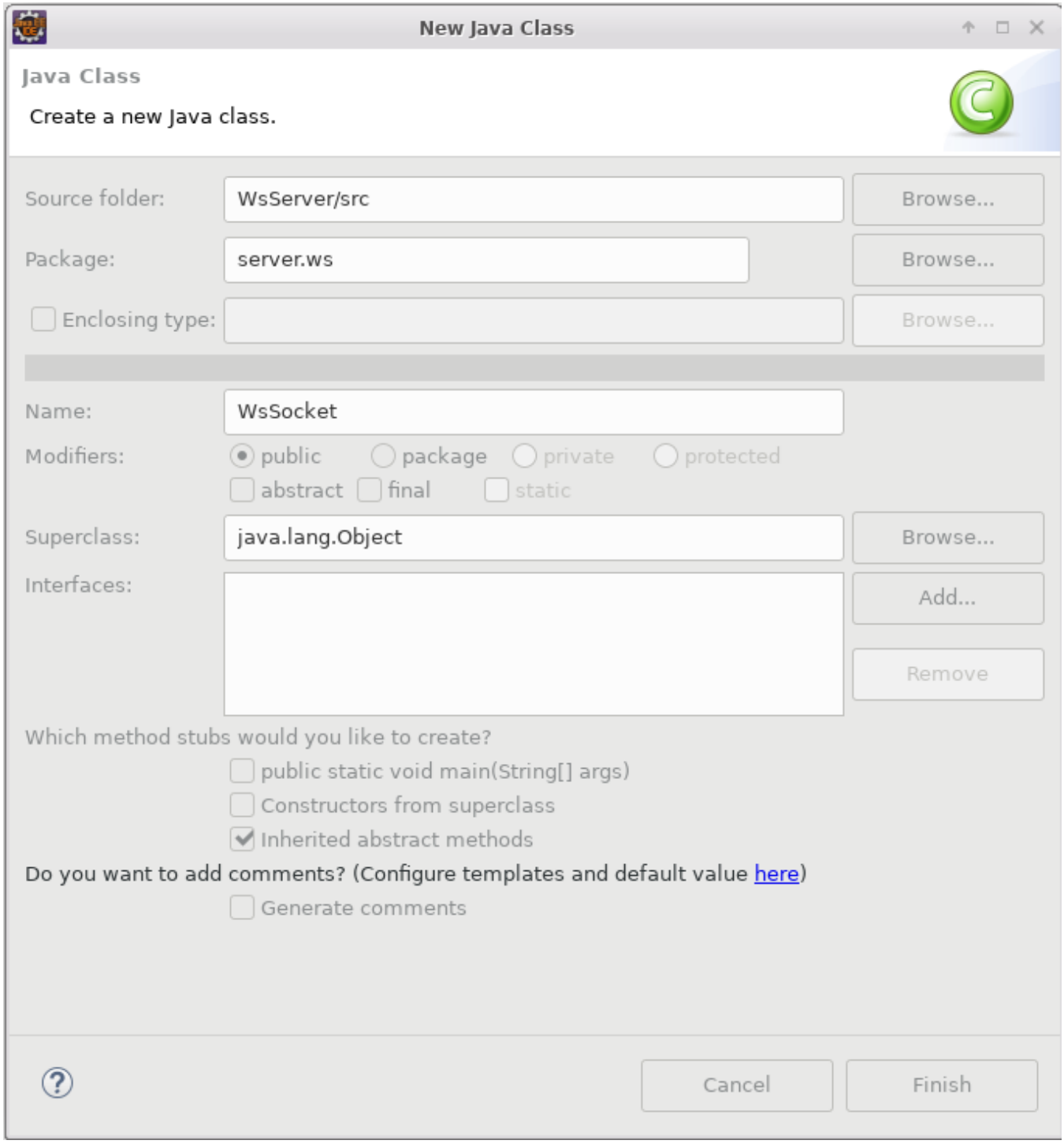

Рисунок 2.8 — Основные настройки класса WsSocket

Листинг 2.3 — Исходный текст программы WsSocket.java

```
/**
 * Пример реализации эхо-сервера, обеспечивающего
 * интерфейс WebSocket/
 */
package server.ws;      // Произвольный вариант пакета.
/**
 * Набор необходимых импортируемых классов,
 * реализующих основу интерфейса WebSocket.
```

```
 */
import javax.websocket.OnClose;
import javax.websocket.OnError;
import javax.websocket.OnMessage;
import javax.websocket.OnOpen;
import javax.websocket.server.ServerEndpoint;

/**
 * Точка входа в проект для соединения
 * с интерфейсом WebSocket.
 */
@ServerEndpoint("/wspoint")
public class WsSocket {
          // Реакция на установление соединения.
          @OnOpen
          public void onOpen(){
              System.out.println("Open Connection ...");
          }

          // Реакция на закрытие соединения.
          @OnClose
          public void onClose(){
              System.out.println("Close Connection ...");
          }

          // Реакция на получение сообщения от клиента.
          // Отправляем сообщение назад клиенту.
          @OnMessage
          public String onMessage(String message){
              System.out.println("Сообщение от клиента: " + message);
              String echoMsg = "Я - сервер: " + message;
              return echoMsg;
          }

          // Реакция на установление соединения.
          @OnError
          public void onError(Throwable e){
              e.printStackTrace();
          }
}
```

Конец листинга 2.3

Шаг 4. Создание исходных тестов html-страниц.

Известно, Apache Tomcat имеет средства для подготовки html-страниц с помощью технологии JSP.

С целью минимизации затрат на разработку проекта и желания сосредоточить свое внимание на предмете изучения, в данной практической работе используются статические варианты html-страниц: *`WsClient.html`* и *`other.html`*.

Html-страница *`WsClient.html`* предназначена для работы с WebSocket,

который реализуется классом сервера WsSocket.java. Для ее открытия необходимо в проекте выделить директорию WebConent, правой кнопкой мыши ативизировать контекстное меню и выбрать шаблон «HTML File». Далее, заполнить появившуюся форму, как показано на рисунке 2.9, и нажать кнопку «Finish».

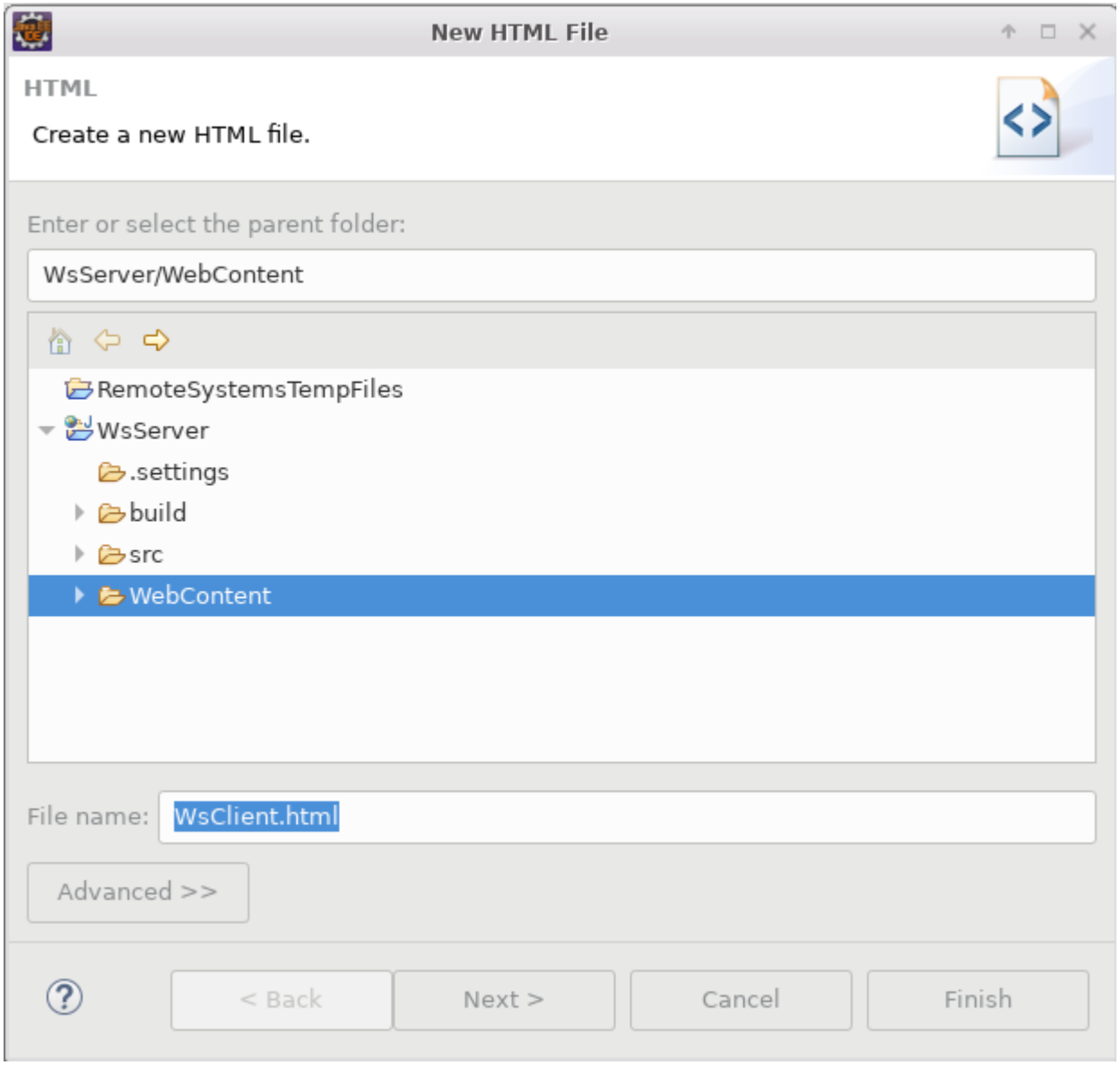

Рисунок 2.9 — Настройки Eclipse EE для страницы WsClient.html

Исходный текст страницы `WsClient.html` представлен на листинге 2.4. Он реализует строку ввода тестовой строки, кнопки передачи сообщения, разрыва соединения с сервером и установки такого соединения. Имеется также ссылка для перехода на станицу `other.html` и текстовое поле для отражения диалога клиента и сервера. Дальнейшие комментарии работы клиента и сервера отражены в тесте самого листинга.

На листинге 2.5 представлен исходный тест страницы `other.html`, назначение которой возврат на страницу `WsClient.html`.

Листинг 2.4 — Исходный текст страницы WsClient.html

```
<!--
Пример статической html-страницы, использующей WebSocket
-->
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Тестовая страница для WebSocket</title>
</head>
<body onunload="wsCloseConnection()">
<!--
Типовая форма страницы, содержащая поле ввода текста и три кнопки
для передачи сообщения, разрыва соединения и установки соединения.
-->
    <form>
        <input id="message" size="40" type="text">

        <input onclick="wsSendMessage();"
            value="Передать сообщение" type="button"><br>

        <input onclick="wsCloseConnection();"
            value="Разъединение"       type="button">

        <input onclick="wsOpenConnection();"
            value="Соединиться"       type="button">
    </form>
<!--
Ссылка для перехода на другую страницу:
предварительно необходимо закрыть соединение по WebSocket.
 -->
      <p><a href="http://localhost:8080/WsServer/other.html"
            onclick="wsCloseConnection();">
            Переход на другую страницу...</a></p>
<!--
Тестовое поле размером 20x60 символов, куда выводятся
все сообщения.
 -->
    <textarea id="echoText" rows="20" cols="60"></textarea>
<!--
Сценарий языка JavaScript, обеспечивающий работу с WebSocket.
 -->
    <script type="text/javascript">
      /**
      Получаем объект, соответствующий вводимому полю сообщения.
      */
          var message = document.getElementById("message");
      /**
      Получаем объект, соответствующий области текста.
      */
```

```
var echoText = document.getElementById("echoText");
echoText.value = "";


var webSocket = 0;

    /**
    Функции, обслуживающие нажатие кнопок формы.
    */
    // Вызывается, когда клиент передает сообщение серверу.
function wsSendMessage(){
    if(!webSocket){
          alert("Соединение закрыто...");
          return;
    }
    webSocket.send(message.value);
    echoText.value += "Посылаю серверу сообщение    : "
          + message.value + "\n";
    message.value = "";
}

    // Вызывается, когда нажата кнопка разрыва соединения.
function wsCloseConnection(){
    webSocket.close();
    webSocket = 0;
}

    // Вызывается, когда нажата кнопка установки соединения.
function wsOpenConnection(){
    if (webSocket){
          alert("Соединение уже открыто...");
          return;
    }

    /**
    Открываем объект типа WebSocket, который имеет
    четыре back-функции, обслуживающие сокет.
    */
    webSocket = new WebSocket(
          "ws://localhost:8080/WsServer/wspoint");

    // Вызывается, когда клиент установил соединение с сервером.
    webSocket.onopen = function(message){
        echoText.value += "Соединился с сервером ... \n\n";
          };

    // Вызавается, когда клиент получил сообщение.
    webSocket.onmessage = function(message){
        echoText.value += "Получаю от сервера сообщение : "
          + message.data + "\n\n";
          };
```

```
            // Вызавается, когда соединение с сервером закрылось.
            webSocket.onclose = function(message){
                echoText.value += "Нажата кнопка 'Разъединение' ... \n";
                  };

            // Вызавается, когда клиент получил сообщение об ошибке.
            webSocket.onerror = function(message){
                  echoText.value += "Error ... \n";
                  };

        }
        // Активизируем функции WebSocket.
        wsOpenConnection();
    </script>
</body>
</html>
```

Конец листинга 2.4

Листинг 2.5 — Исходный текст страницы othe.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Дополнительная страница</title>
</head>
<body>
      <a href="http://localhost:8080/WsServer/WsClient.html">
      Назад к WebSocket!!!...</a>
</body>
</html>
```

Конец листинга 2.5

Шаг 6. Запуск проекта на исполнение.

Для запуска проекта на исполнение следует выделить окно редактора со страницей WsClient.html и нажать кнопку «Run» системы Eclipse EE.

В результате такого действия, сначала появится окно выбора сервера, показанное на рисунке 2.10, которое необходимо подтвердить.

Затем появится вкладка встроенного в Eclipse EE браузера, в котором отобразится html-страница WsClient.html.

Далее, необходимо провести исследование разработанного проекта.

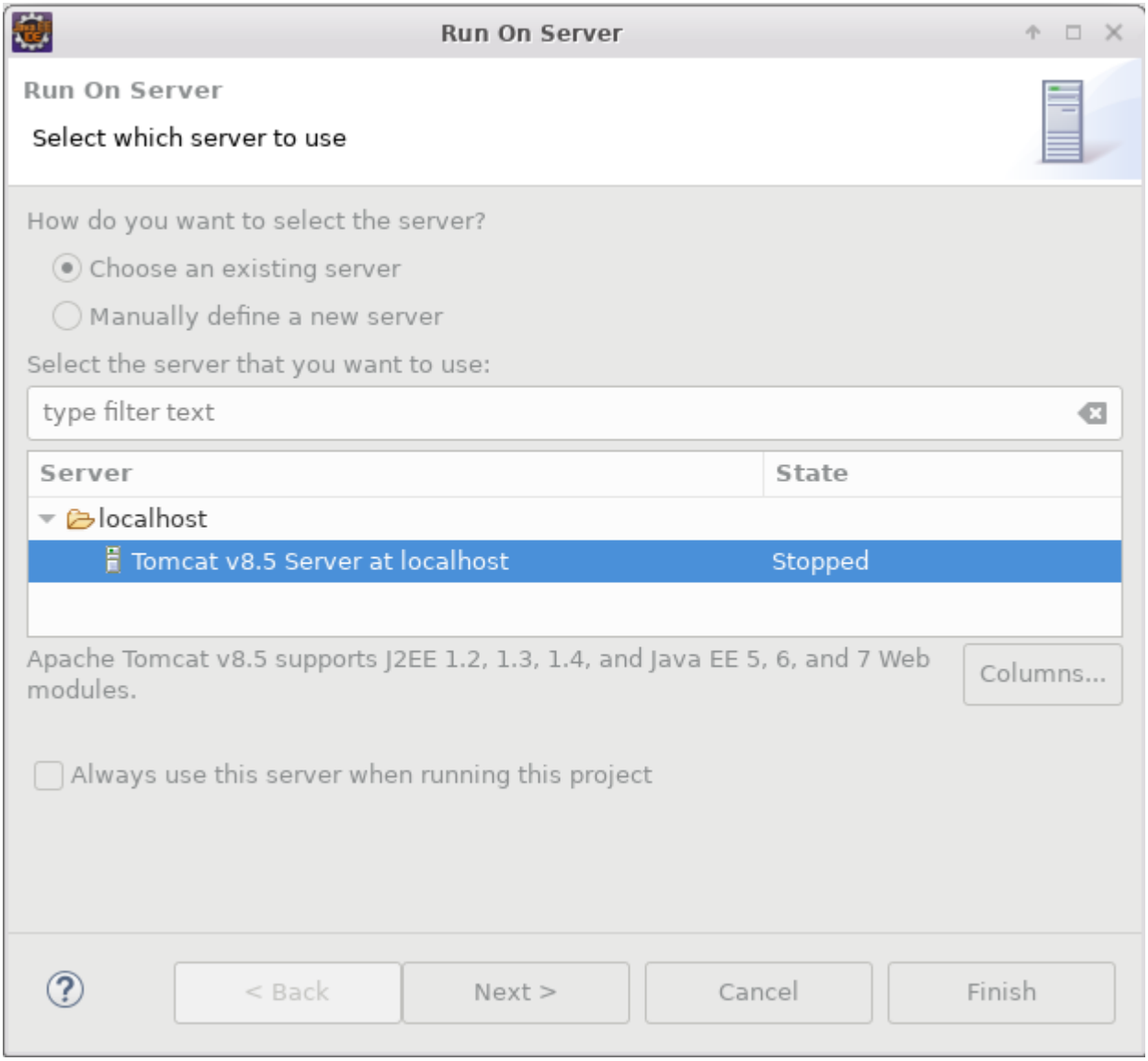

Рисунок 2.10 — Запрос на выбор сервера Apache Tomcat

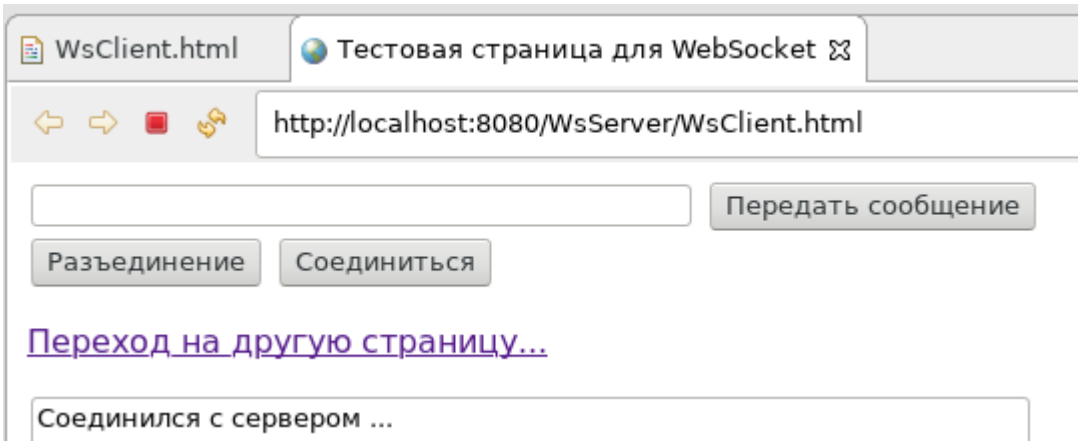

Рисунок 2.11 — Окно браузера для работы клиента с сервером

2.2.2.3 Учебные задания

Аспиранту следует реализовать проект WsServer, в объеме листрингов 2.3 — 2.5, и провести его всестороннее иследование.

Провести модификацию проекта в плане реализации метода `onMessage()` класса `WsSocket`, представленного на листинге 2.3. Эта модификация должна включать анализ сервером принятого от клиента сообщения и формирование ответа, в зависимости от полученного сообщения. Число таких вариантов модификации должно быть не менее пяти.

Провести модификацию проекта в плане реализации нового интерфейса, предоставляемого страницей WsClient.html. Для этого:

- создать в проекте сервлет, который бы принимал первоначальное соединение с сервером;
- разработать JSP-страницу, функционально подобную странице WsClient.html, к которой бы обращался сервлет во время первого соединения клиента с сервером.

Провести исследование модифицированного варианта проекта и результаты отразить в личном отчете.

Замечание

Данную практическую работу рекомендуется проводить с помощью обучающей среды кафедры АСУ, описанной в руководстве [5] и имеющей все необходимые инструменты для реализации заданного проекта.

При необходимости, аспиранту рекомендуется использовать методическое руководство по выполнению практических занятий [3].

2.2.3 Список использованных источников

2.2.1 Всемирная паутина — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/Всемирная_паутина). Проверено: 25.11.2018.

2.2.2 Гипертекст — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/Гипертекст). Проверено: 25.11.2018.

2.2.3 HTML — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/HTML). Проверено: 25.11.2018.

2.2.4 CGI — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/CGI). Проверено: 25.11.2018.

2.2.5 Apache HTTP Server — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/Apache_HTTP_Server). Проверено: 25.11.2018.

2.2.6 PHP — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/PHP). Проверено: 25.11.2018.

2.2.7 JavaScript — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/JavaScript). Проверено: 25.11.2018.

2.2.8 Тонкий клиент — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/Тонкий_клиент). Проверено: 25.11.2018.

2.2.9 Сервлет (Java) — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/Сервлет_(Java)). Проверено: 25.11.2018.

2.2.10 Apache Tomcat — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/Apache_Tomcat). Проверено: 25.11.2018.

2.2.11 Java-апплет — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/Java-апплет). Проверено: 25.11.2018.

2.2.12 AJAX — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/AJAX). Проверено: 25.11.2018.

2.2.13 HTML5 — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/HTML5). Проверено: 25.11.2018.

2.2.14 RFC 6455 — The WebSocket Protocol: [Электронный документ]. - (https://tools.ietf.org/html/rfc6455#page-37). Проверено: 25.11.2018.

2.2.15 WebSocket — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/WebSocket). Проверено: 25.11.2018.

2.2.16 Apache Tomcat WebSocket Tutorial: [Электронный документ]. - (https://examples.javacodegeeks.com/enterprise-java/tomcat/apache-tomcat-websocket-tutorial/). Проверено: 25.11.2018.

2.3 Практическая работа №6 «Технологии мультимедиа»

Цель работы: Освоение технологии мультимедиа на примере использования векторной графики в формате SVG.

2.3.1 Самостоятельная работа

Представление звука и изображения в компьютерных системах. Устройства ввода, обработки и вывода мультимедиа информации. Форматы представления звуковых и видеофайлов. Оцифровка и компрессия. Программные средства записи, обработки и воспроизведения звуковых и видеофайлов. Мультимедиа в вычислительных сетях.

Литература: [2, 10].

2.3.2 Порядок выполнения работы

В учебнике для вузов [10] дается следующее определение: «**Мультимедиа** (multimedia — буквально многосредовость) - *область компьютерной технологии*, связанная с обработкой информации, имеющей различное физическое представление (текст, графика, рисунок звук, анимация, видео и т. п.) и/или существующей на различных носителях (магнитные и оптические диски, аудио- и видео-ленты и т. д.). ...».

Аналогичное по сути определение дает и Википедия [2.3.1]: «**Мультимедиа** (англ. *multimedia*) — контент, или содержание, которое одновременно передаётся в разных формах: звук, анимированная компьютерная графика, видеоряд. Например, в одном объекте-контейнере может содержаться текстовая, аудиальная, графическая и видеоинформация, а также, возможно, способ интерактивного взаимодействия с ней. Это достигается использованием определённого набора аппаратных и программных средств.

Термин *мультимедиа* также зачастую используется для обозначения носителей информации, позволяющих хранить значительные объёмы данных и обеспечивать достаточно быстрый доступ к ним (первыми носителями такого типа были компкт-диски). В таком случае термин *мультимедиа* означает, что компьютер может использовать такие носители и предоставлять информацию пользователю через все возможные виды данных, такие как аудио, видео, анимация, изображение и другие в дополнение к традиционным способам предоставления информации, таким как текст. ...».

Можно рассмотреть еще много разных источников и везде буде встречаться подобное определение, которое указывает на достаточно широкую проблематику этой технологии, развивающейся вместе с отдельными технологическими достиже-

ниями, как это показано на рисунке 2.12.

Рисунок 2.12 — Технологии мультимедия (заимствовано из источника [2.3.2])

Не имея возможности охватить все указанные выше технологические направления мультимедиа, ограничимся мультимедийным представлением графической информации, представляемой форматом векторной графики — SVG.

2.3.2.1 История вопроса

Согласно [2.3.3]: «**SVG** (от англ. ***S**calable **V**ector **G**raphics* — масштабируемая векторная графика) — язык разметки масштабируемой векторной графики, созданный Консорциумом Всемирной паутины (W3C) и входящий в подмножество расширяемого языка разметки XML, предназначен для описания двумерной векторной и смешанной векторно/растровой графики в формате XML. Поддерживает как неподвижную, так и анимированную интерактивную графику — или, в иных терминах, декларативную и скриптовую. Не поддерживает описание трёхмерных объектов (не путать с имитацией трёхмерности путём светотени). Это открытый стандарт, который является рекомендацией консорциума W3C — организации, разработавшей такие стандарты, как HTML и XHTML. В основу SVG легли языки

разметки VML и PGML. Разрабатывается с 1999 года. В 2001 году вышла версия 1.0, в 2011 — версия 1.1, которая остаётся актуальной до сегодняшнего дня. В настоящее время в активной разработке находится версия 2. ...».

SVG позволяет отображать:

- Линии и ломаные линии;
- Многоугольники;
- Окружности и эллипсы;
- Кривые Безье;
- Сложные контуры;
- Текст.

Набор таких элементов позволяет создавать достаточно сложные изображения, которые легко масштабируются и могут подвергаться анимации. Это создает широкие возможности использования этого формата не только в веб-разработке, но и в других приложениях.

Для первоначального знакомства со структурой формата SVG можно воспользоваться статьей [2.3.4]. Полное описание этого формата дано в [2.3.5].

2.3.2.2 Технология мультимедиа на примере формата SVG

Следует заметить, что сам формат SVG хорошо описывает только векторную графику. Мультимедийность ему обеспечивает среда, в которой SVG используется. Такая среда обеспечивается двумя составляющими:

- языковыми средствами HTML (XHTML), каскадными таблицами стилей CSS, средствами языков JavaScript и SMIL;
- средствами отображения, в качестве которых обычно используются браузеры, реализующие интерпретацию указанных выше языковых средств.

В настоящее время все большую популярность приобретает язык SMIL [2.3.6]: «**SMIL** (рекомендованное произношение: [smail] «смил») (*The Synchronized Multimedia Integration Language*) — язык разметки для создания интерактивных мултимедийных презентаций. Является рекомендацией консорциума Всемирной паутины.

SMIL описывает разметку для временной синхронизации, размещения, анимаций, визуальных преобразований и многих других аспектов. Одной из областей применения SMIL является создание слайдшоу для презентаций. Также технология SMIL позволяет демонстрировать многие типы файлов, такие как текст, видео и аудио. SMIL похож на HTML, он выполнен на основе XML и позволяет включать в себя ссылки на другие презентации SMIL, а также кнопки, такие как «Старт», «Стоп» и др.

SMIL был разработан в 1997 году и служит в том числе для описания работы с мультимедийным контентом с многих источников, в том числе серверов в интернете. Другим массовым применением формата является его интеграция в формат цифровых аудиокниг DAISY. ...».

Интерес к этому языку повышается за счет его декларативности, что в целом уменьшает объем кода, необходимого для достижения конкретного результата. Полное описание языка SMIL можно найти в официальной документации [2.3.7], а для практических целей воспользуемся достаточно объемной публикацией [2.3.8], в которой приведено множество конкретных и простых примеров.

Как указано во введении [2.3.8]: «Графика SVG может быть анимирована с использованием элементов анимации. Эти элементы анимации были первоначально определены в спецификации SMIL Animation; они включают:

- <animate> - элемент, который позволяет оживлять скалярные атрибуты и свойства в течение определенного периода времени.
- <set> - удобный и короткий оператор для анимации, который полезн для задания значений анимации нечисловым атрибутам и свойствам, таким как свойство видимости.
- <animateMotion> - элемент, который перемещает элемент вдоль пути движения.
- <animateColor> - элемент, который изменяет значение цвета определенных атрибутов или свойств с течением времени. Обратите внимание, что элемент <animateColor> устарел в пользу простого использования для целевых свойств элемента animate, которые могут принимать значения цвета. Несмотря на то, что он все еще присутствует в спецификации SVG 1.1, четко отмечено, что он устарел; и он полностью исключен из спецификации SVG 2.

В дополнение к элементам анимации, определенным в спецификации SMIL, SVG включает расширения, совместимые со спецификацией анимации SMIL; эти расширения включают атрибуты, расширяющие функциональность элемента <animateMotion> и дополнительных элементов анимации. Расширения SVG включают:

- <animateTransform> - позволяет анимировать один из атрибутов преобразования SVG с течением времени, например атрибут transform.
- path (attribute) - позволяет указать любую особенность синтаксиса данных пути SVG в атрибуте пути к элементу animateMotion (SMIL Animation разрешает только подмножество синтаксиса данных пути SVG в атрибуте path). Более подробно об animateMotion изложено в следующем разделе.
- <mpath> - используется вместе с элементом animateMotion для ссылки на path, который должен использоваться как путь для движения. Элемент mpath включен внутри элемента animateMotion перед закрывающим тегом.
- keypoints (attribute) - используется как атрибут для animateMotion для обеспечения точного управления скоростью анимации на пути движения.
- rotate (attribute) - используется как атрибут для animateMotion для управления направлением движения, чтобы его ось X указывала в том же направлении (или в противоположном направлении) относительно к касательной вектора траектории движения. Этот атрибут является ключом к тому, чтобы движение по пути работало так, как вы ожидали. Подробнее об этом в разделе animateMotion. ...».

Для демонстрации возможностей анамации векторной графики SVG рассмотрим следующую задачу:

1) выделим полосу оранжевого фона, на которой зададим область рисования размером 200x200 пикселей;
2) в центре заданной области нарисуем круг радиуса 50 пикселей, цвет которого циклически (с периодом 3 секунды) будет меняться от синего (#0000cc) до зеленого цвета (#00cc00);
3) вокруг центрального круга будет вращаться (с периодом в 1 секунду) белый круг радиуса 10 пикселей.

Реализацию данной задачи проведем в рамках проекта WsServer, который был использован в предыдущем практическом занятии. Все операции зафиксируем в виде последовательности отдельных шагов.

Шаг 1. Создадим Circ-animation.html, как показано на рисунке 2.13.

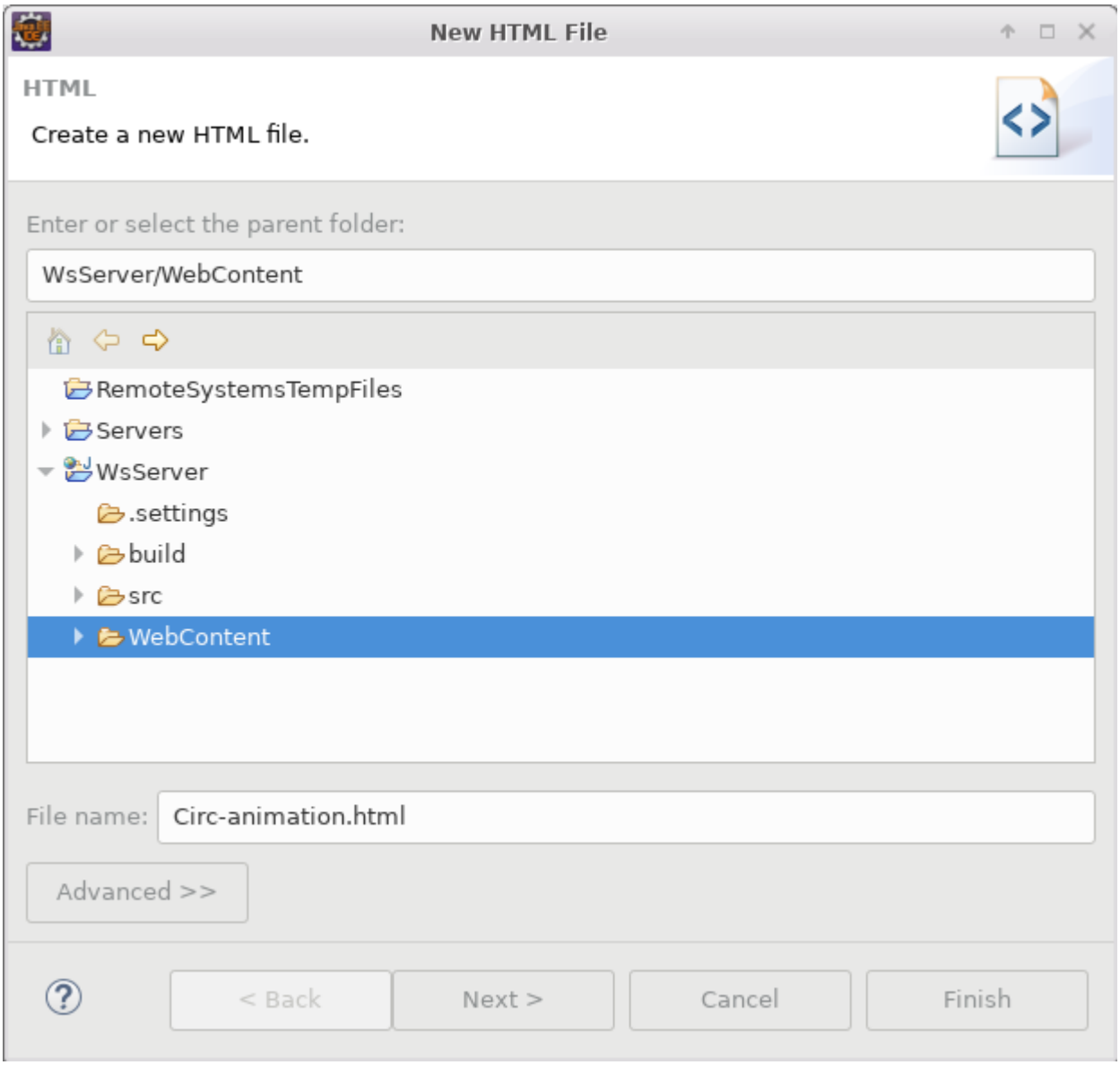

Рисунок 2.13 - Создание html-страницы Circ-animation.html

Шаг 2. Теперь необходимо в страницу *WsClient.html* вставить ссылку на страницу *Circ-animation.html*, как показано на листинге 2.6.

Листинг 2.6 — Ссылка на страницу Circ-animation.html

```
...
<!--
Ссылка для перехода на страницу анимации:
предварительно необходимо закрыть соединение по WebSocket.
 -->
      <p><a href="http://localhost:8080/WsServer/Circ-animation.html"
            onclick="wsCloseConnection();">
            Переход на страницу анимации ...</a></p>
...
```

Конец листинга 2.6

Шаг 3. Проведем создание страницы *Circ-animation.html*, содержащей реализацию поставленной задачи, как показано со всеми комментариями на листинге 2.7.

Листинг 2.7 — Страница Circ-animation.html

```
<!--
Страница Circ-animation.html, реализующая задачу отображения
двух анимационных фигур:
- круг изменяющий цвет за 3 секунды;
- круг белого цвета, вращающийся вокруг первого круга.
-->
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Окружность со спутником</title>
</head>
<body>
      <!--
      Ссылка на страницу сокета.
      -->
      <p><a href="http://localhost:8080/WsServer/WsClient.html">
      Назад к WebSocket!!!...</a></p>

      <!--
      Область рисунка, задающего абзац текста html оранжевого цвета.
      -->
      <p style="background: #ff7700;">

      <!--
      Объявление SVG-рисунка с областью отображения 200x200 пикселей.
       -->
```

```
      <svg viewBox="0 0 200 200" width="200" height="200">

            <!--
            Окружность в центре области отображения SVG-рисунка,
            Задающая круг радиуса 50 пикселей.
             -->
            <circle id="circ1" cx="100" cy="100" r="50" fill="#0000cc"></circle>
            <!--
            Анимация круга, изменением его цвета.
            Цвет меняется от #0000CC до #00CC00 за 3 секунды.
            Количество повторов - не ограничено.
             -->
            <animate
           xlink:href="#circ1"
           attributeName="fill"
           from="#00CC"
           to="#00cc00"
           dur="3s"
           repeatCount="indefinite"
           begin="0s"
           end="move.begin"
           fill="freeze"
           id="changeColor"/>

            <!--
            Окружность белого цвета, радиусом 10 пикселей, расположенная в
            точке с координатами: (10,100).
             -->
            <circle id="circ2" cx="10" cy="100" r="10" fill="#FFF"> </circle>
            <!--
            Анимация круга, посредством движения по заданной траектории.
            Траектория движения задается атрибутом path.
            Количество повторов - не ограничено.
             -->
            <animateMotion
                xlink:href="#circ2"
            dur="1s"
            repeatCount="indefinite"
            begin="0s"
            end="move.begin"
            fill="freeze"
            path="M 0,0 c0,-70,50,-80,90,-90 c0,0,70,10,90,90
                        c0, 70,-50,80,-90,90 c0,0,-70,-10,-90,-90" />

      </svg>
      </p>
</body>
</html>
```

Конец листинга 2.7

Шаг 4. Запуск и исследование приложения динамического SVG-рисунка.

Запуск приложения производится, как и в предыдущем практическом занятии, активизацией страницы WsServer.html и нажатием кнопки «Run».

Когда активированная страница появится в браузере, следует перейти по ссылке «Переход на страницу анимации...». В результате появится страница с анимированным SVG-рисунком, показанная на рисунке 2.14.

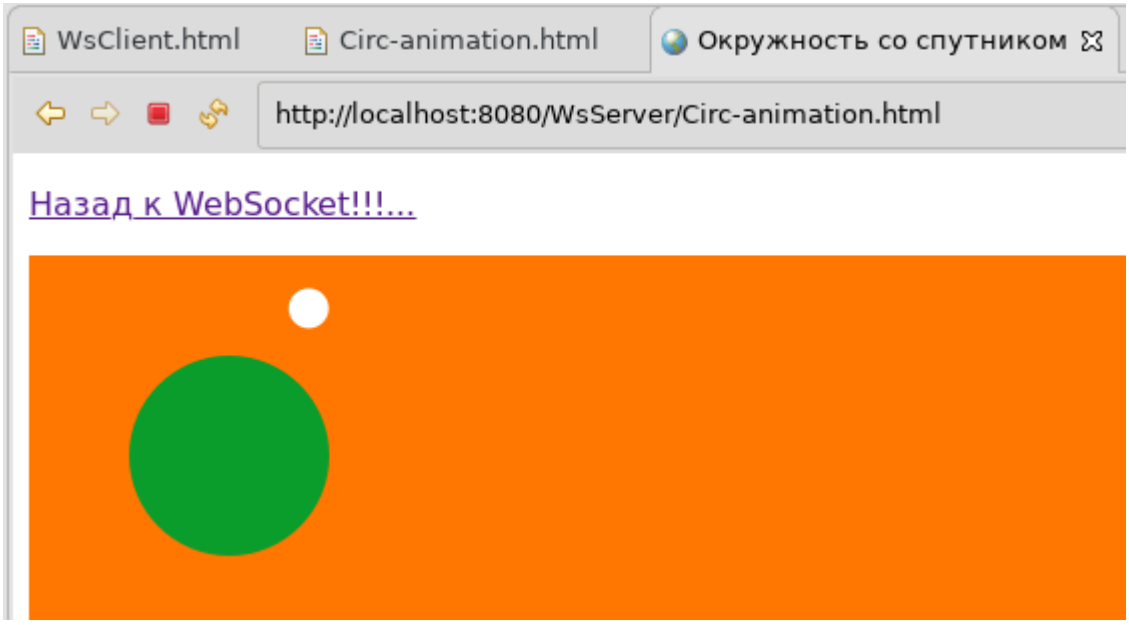

Рисунок 2.14 — Демонстрация анимированного SVG-рисунка

2.3.2.3 Учебные задания

Аспиранту следует внимательно изучить реализацию задачи, представленную на листинге 2.7, и самостотельно повторить ее решение.

Затем следует обратиться к публикации [2.3.8], выбрать, изучить и реализовать один из представленных там примеров.

По завершению реализации примера провести его исследование и описание, а результаты работы отразить в личном отчете.

2.3.3 Список использованных источников

2.3.1 Мультимедиа — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/Мультимедиа). Проверено: 01.12.2018.

2.3.2 Технологии мультимедиа: [Электронный документ]. - (Технологии мультимедиа). Проверено: 01.12.2018.

2.3.3 SVG — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/SVG). Проверено: 01.12.2018.

2.3.4 Структура SVG документа | Описание и примеры стандартных функций SVG: [Электронный документ]. - (https://svg-art.ru/?page_id=1009). Проверено: 01.12.2018.

2.3.5 Scalable Vector Graphics (SVG) 1.1 (Second Edition): [Электронный документ]. - (https://www.w3.org/TR/SVG11/). Проверено: 01.12.2018.

2.3.6 SMIL — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/SMIL). Проверено: 01.12.2018.

2.3.7 SMIL Animation: [Электронный документ]. - (https://www.w3.org/TR/2001/REC-smil-animation-20010904/). Проверено: 01.12.2018.

2.3.8 A Guide to SVG Animations(SMIL): [Электронный документ]. - (https://css-tricks.com/guide-svg-animations-smil/). Проверено: 01.12.2018.

3 Интеллектуальные технологии

Рассматривая интеллектуальные технологии, мы будем опираться на определение системы искусственного интеллекта (СИИ), приведенного в учебном пособии Павлова С.Н. [12, с.12] «СИИ — это компьютерная, креативная система (многофункциональная, интегрированная, интеллектуальная) со сложной структурой, использующая накопление и корректировку знаний (синтаксической, семантической, прагматической информации) для постановки и достижения цели (целенаправленного поведения), адаптации к изменениям среды и внутреннего состояния путем изменения среды или внутреннего состояния.».

Базовой составляющей развиваемых интеллектуальных технологий является понятие искусственного интеллекта, трактуемое в достаточно широких аспектах. Например, Википедия раскрывает это понятие в следующих общих планах [3.1.1]: «**Иску́сственный интелле́кт** (**ИИ**; англ. *artificial intelligence, AI*): наука и технология создания интеллектуальных машин, особенно интеллектуальных компьютерных программ; свойство интеллектуальных систем выполнять творческие функции, которые традиционно считаются прерогативой человека.

ИИ связан со сходной задачей использования компьютеров для понимания человеческого интеллекта, но не обязательно ограничивается биологически правдоподобными методами. ...

Процитированное в преамбуле определение искусственного интеллекта, данное Джоном Маккарти в 1956 году на конференции в Дартмутском университете, не связано напрямую с пониманием интеллекта у человека. Согласно Маккарти, ИИ-исследователи вольны использовать методы, которые не наблюдаются у людей, если это необходимо для решения конкретных проблем. ...

Даются следующие определения искусственного интеллекта:

1. Научное направление, в рамках которого ставятся и решаются задачи аппаратного или программного моделирования тех видов человеческой деятельности, которые традиционно считаются интеллектуальными.
2. Свойство интеллектуальных систем выполнять функции (творческие), которые традиционно считаются прерогативой человека. При этом интеллектуальная система — это техническая или программная система, способная решать задачи, традиционно считающиеся творческими, принадлежащие конкретной предметной области, знания о которой хранятся в памяти такой системы. Структура интеллектуальной системы включает три основных блока — базу знаний, решатель и интеллектуальный интерфейс, позволяющий вести общение с ЭВМ без специальных программ для ввода данных.
3. Направление в информатике и информационных технолгиях, задачей которого является воссоздание с помощью вычислительных систем и иных искусственных устройств разумных рассуждений и действий. ...».

Если говорить непосредственно об интеллектуальных технологиях, то наиболее подходящим термином будет понятие интеллектуальных систем, которое в

[3.1.2] дается следующим образом: «**Интеллектуальная система** (ИС, англ. *intelligent system*) — это техническая или программная система, способная решать задачи, традиционно считающиеся творческими, принадлежащие конкретной предметной области, знания о которой хранятся в памяти такой системы. Структура интеллектуальной системы включает три основных блока — базу знаний, механизм вывода решений и интеллектуальный интерфейс. ...».

Наиболее существенной архитектурной особенностью таких систем является наличие наличие базы знаний [3.1.3]: «**База знаний** (БЗ; англ. *knowledge base, KB*) — база данных, содержащая правила вывода и информацию о человеческом опыте и знаниях в некоторой предметной области (ISO/IEC/IEEE 24765-2010, ISO/IEC 2382-1:1993). В самообучающихся системах база знаний также содержит информацию, являющуюся результатом решения предыдущих задач.

Современные базы знаний работают совместно с системами поиска и извлечения информации. Для этого требуется некоторая модель классификации понятий и определённый формат представления знаний. Иерархический способ представления в базе знаний набора понятий и их отношений называется онтологией.

Онтологию некоторой области знаний вместе со сведениями о свойствах конкретных объектов часто называют «базой знаний». Вместе с тем полноценные базы знаний (в отличие от обычной базы данных) содержат в себе не только фактическую информацию, но и правила вывода, позволяющие делать автоматические умозаключения об уже имеющихся или вновь вводимых фактах и тем самым производить семантическую (осмысленную) обработку информации.

Область наук об искусственном интеллекте, изучающая базы знаний и методы работы со знаниями, называется инженением знаний. ...».

Именно инженерии знаний, включающей в себя технологии построения онтологий различных предметных областей, и посвящены последние три практических занятия данной дисциплины.

Чтобы сосредоточить и конкретизировать направление практических работ, ограничимся предметной областью семантической путины [3.1.4]: «**Семантическая паутина** (англ. *semantic web*) — это общедоступная глобальная семантическая сеть, формируемая на базе Всемирной паутины путём стандартизации представления информации в виде, пригодном для машинной обработки.

В обычной Всемирной паутине, основанной на HTML-страницах, информация заложена в тексте страниц и предназначена для чтения и понимания человеком. Семантическая паутина состоит из машинно-читаемых элементов — узлов семантической сети, с опорой на онтологии. Благодаря этому программы-клиенты получают возможность непосредственно получать из интернета утверждения вида «предмет — вид взаимосвязи — другой предмет» и вычислять по ним логические заключения. Семантическая паутина работает параллельно с обычной Всемирной паутиной и на её основе, используя протокол HTTP и идентификаторы ресурсов URI.

Название «Семантическая паутина» было впервые введено Тимом Бернерсом-Ли (изобретателем Всемирной паутины) в сентябре 1998 года, и называется им «следующим шагом в развитии Всемирной паутины». Позже в своём блоге он предложил в качестве синонима термин «гигантский глобальный граф» (англ. *giant*

global graph, *GGG*, по аналогии с WWW). Концепция семантической паутины была принята и продвигается консорциумом Всемирной паутины. ...».

3.1 Практическая работа №7 «Инженерия знаний»

Цель работы: общее изучение представлений о семантической паутине, модели ***R**esource **D**escription **F**ramework* (RDF), а также о редакторе онтологий и RDF - Protege.

3.1.1 Самостоятельная работа

Основные разделы теории и приложений искусственного интеллекта. Описание и постановка задачи. Задачи в пространстве состояний, в пространстве целей. Классификация задач по степени сложности. Линейные алгоритмы. Полиномиальные алгоритмы. Экспоненциальные алгоритмы. Виды и уровни знаний. Знания и данные. Факты и правила. Принципы организации знаний.

Литература: [12 - 16].

3.1.2 Порядок выполнения работы

В соответствии с заявленной тематикой практического занятия, изучаемые интеллектуальные технологии относятся к области науки — инженерия знаний [3.1.5]: «**Инженерия знаний** (англ. *knowledge engineering*) — область наук об искусственном интеллекте, связанная с разработкой экспертных систем и баз знаний. Изучает методы и средства извлечения, представления, структурирования и использования знаний. …

Инженерия знаний (ИЗ) была определена Фейгенбаумом и МакКордак в 1983 году как:

«ИЗ — раздел (дисциплина) инженерии, направленный на внедрение знаний в компьютерные системы для решения сложных задач, обычно требующих богатого человеческого опыта.»

В настоящее время это также предполагает создание и обслуживание подобных систем (Кендэл, 2007). Это также тесно соприкасается с разработкой программного обеспечения и используется во многих информационных исследованиях, например таких, как исследования искусственного интеллекта, включая базы данных, сбор данных, экспертные системы, систем поддержки принятия решений и географические информационные системы. ИЗ связана с математической логикой, также используемой в разных научных дисциплинах, например в социологии где «подопытными» являются люди, а цели исследований — понимание, как работает человеческая логика на примере взаимоотношений в обществе. ...».

Поскольку одна из главных задач инженерии знаний — построение баз знаний о предметной области исследования, то отличительной особенностью представления знаний, по сравнению с представлением данных, является семантика понятий и отношений описывающих предметную область. С другой стороны, существенное значение имеет объем хранимых знаний и широта их использования в различных программных приложениях.

Очевидно, широкое использование информационных источников в сети Internet, порождает актуальную задачу современного общества к систематизации знаний в системе, которая названа Всемирной паутиной. Посколку такая система ориентирована на максимально широкий круг взаимодействующих сторон поиска и обмена информацией, то на задачу представления знаний были брошены значительные научные силы и материальные ресурсы. Как результат, были разработаны и продолжают разрабатываться средства поиска, извлечения и обмена информацией, отражающие семантику объектов и субъектов Всемирной паутины.

В данной практической работе мы проведем краткое общее изучение представлений о семантической паутине, построенной на модели ***R**esource **D**escription **F**ramework* (RDF), а затем проведем установку специального редактора онтологий и RDF — Protege, представляющего пример свободно распространяемого инструментария работы семантикой Всемирной паутины.

В последующем, результаты этой работы используются на следующих двух практических занятиях.

3.1.2.1 История вопроса

Основным теоретическим концептом, отражающим представление знаний, является представление о семантической сети [3.1.6]: «**Семанти́ческая сеть** — информационная модель предметной области, имеющая вид ориентированного графа, вершины которого соответствуют объектам предметной области, а дуги (рёбра) задают отношения между ними. Объектами могут быть понятия, события, свойства, процессы. Таким образом, семантическая сеть является одним из способов представления знаний. В названии соединены термины из двух наук: семантика в языкознании изучает смысл единиц языка, а сеть в математике представляет собой разновидность графа — набора вершин, соединённых дугами (рёбрами), которым присвоено некоторое число. В семантической сети роль вершин выполняют понятия базы знаний, а дуги (причем направленные) задают отношения между ними. Таким образом, семантическая сеть отражает семантику предметной области в виде понятий и отношений. ...».

На рисунке 3.1 представлен пример семантической сети, заимствованный из источника [3.1.6]. Там же отмечено, что «… Неправильно приравнивать друг другу понятия «Семантическая сеть» (англ. *Semantic Network*) и «Семантическая паутина» (англ. *Semantic Web*). Хотя эти понятия не эквивалентны, тем не менее, они связаны …

Концепция организации гипертекста напоминает *однородную бинарную* семантическую сеть, однако здесь есть существенное отличие:

1. Связь, осуществляемая гиперссылкой, не имеет семантики, то есть не описы-

вает смысла этой связи. Назначение семантической сети состоит в том, чтобы описать *взаимосвязи* объектов, а не дополнительную информацию по предметной области. Человек может разобраться, зачем нужна та или иная гиперссылка, но компьютеру эта связь не понятна.

2. Страницы, связываемые гиперссылками, являются *документами*, описывающими, как правило, проблемную ситуацию в целом. В семантической сети вершины (то, что связывают отношения) представляют собой *понятия* или *объекты реального мира*.

Попытка создания семантической сети на основе Всемирной паутины получила название **семантической паутины**. Эта концепция подразумевает использование языка RDF (языка разметки на основе XML) и призвана придать ссылкам некий смысл, понятный компьютерным системам. Это позволит превратить Интернет в распределённую базу знаний глобального масштаба. ...».

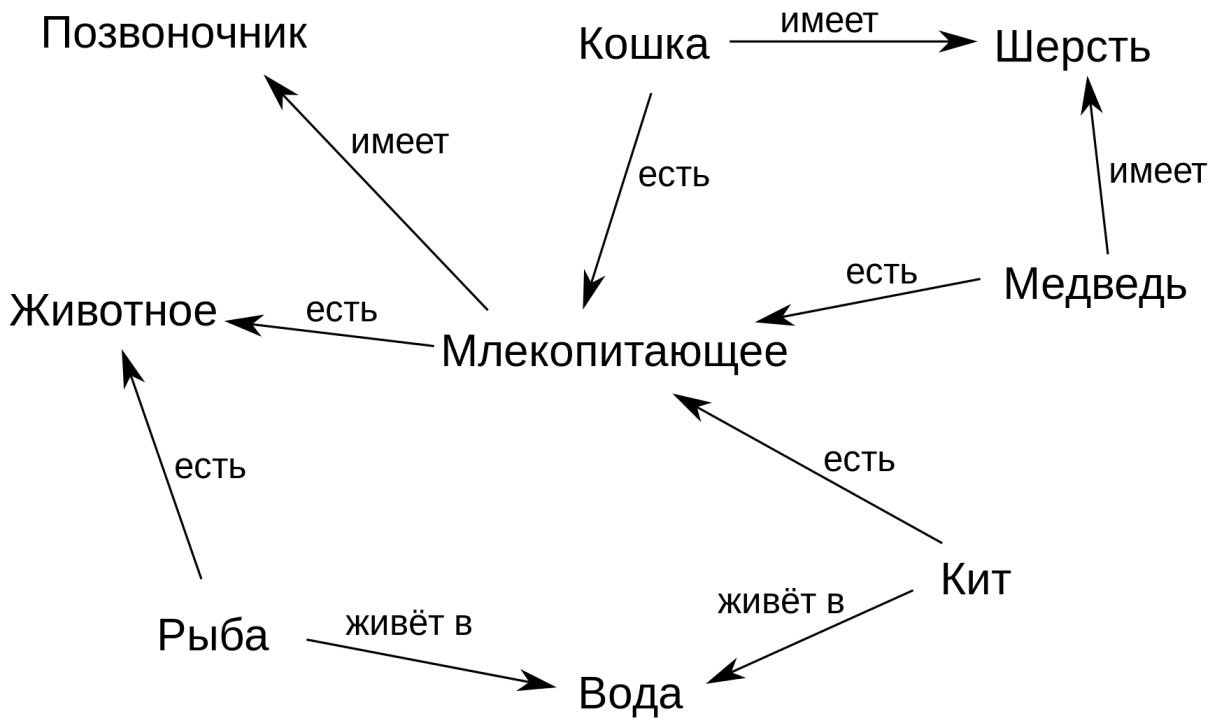

Рисунок 3.1 — Пример семантической сети (заимствовано из источника [3.1.6])

Современная архитектура семантической паутины представляется в следующем виде [3.1.4]: «Техническую часть Семантической паутины составляет семейство стандартов на языки описания, включающее XML, XML Schema, RDF, RDF Schema, OWL, а также некоторые другие. Располагая их в порядке повышения уровня абстракции, реализуемого тем или иным языком, получаем:

- **XML** предоставляет синтаксис для определения структуры документа, подлежащего машинной обработке. Синтаксис XML не несёт семантической наг-

рузки.

- **XML Schema** определяет ограничения на структуру XML-документа. Стандартный синтаксический анализатор языка XML в состоянии проверить произвольный XML-документ на соответствие его структуры так называемой *схеме документа*, описанной в XML Schema.
- **RDF** представляет собой простой способ описания экземплярных данных в формате *субъект-отношение-объект*, в котором в качестве любого элемента этой тройки используются только идентификаторы ресурсов (за исключением объекта, которому разрешено быть литералом). Существует стандартизованное отображение этих троек на XML-документы предопределённой структуры (то есть консорциумом W3 определена *схема* XML-документов, содержащих RDF-описания), а также на другие форматы представления (например, в нотацию N3).
- **RDF Schema** описывает набор атрибутов (здесь их точнее назвать *отношениями*), таких, как `rdfs:Class`, для определения новых типов RDF-данных. Языком поддерживается также отношение наследования типов `rdfs:subClassOf`.
- **OWL** расширяет возможности по описанию новых типов (в частности, добавлением перечислений), а также позволяет описывать новые типы данных RDF Schema в терминах уже существующих (например, определять тип, являющийся пересечением или объединением двух существующих).
- **Микроданные (HTML microdata)** — это международный стандарт семантической разметки HTML-страниц, с помощью атрибутов, описывающих смысл информации, содержащейся в тех или иных HTML-элементах. Такие атрибуты делают контент страниц машиночитаемым, то есть позволяют в автоматическом режиме находить и извлекать нужные данные. ...».

Таким образом мы видим, что семантическая паутина имеет достаточно емкий набор понятий, основанный на языке XML. Общий стек этих понятий представлен на рисунке 3.2 и должен быть реализован программными средствами, обеспечивающими доступ к знаниям Всемирной паутины.

Практическая реализация самого доступа к семантической сети реализуется с помощью описания ее ресурсов [3.1.7]: «**Resource Description Framework** (RDF, «среда описания ресурса») — это разработанная консорциум Всемирной путины модель для представления данных, в особенности — метаданных. RDF представляет *утверждения* о *ресурсах* в виде, пригодном для машинной обработки. RDF является частью концепции семантической паутины.

Ресурсом в RDF может быть любая сущность — как информационная (например, веб-сайт или изображение), так и неинформационная (например, человек, город или некое абстрактное понятие). Утверждение, высказываемое о ресурсе, имеет вид «субъект — предикат — объект» и называется ***триплетом***. Утверждение «небо голубого цвета» в RDF-терминологии можно представить следующим образом: субъект — «небо», предикат — «имеет цвет», объект — «голубой». Для обозначения субъектов, отношений и объектов в RDF используются URI. ...».

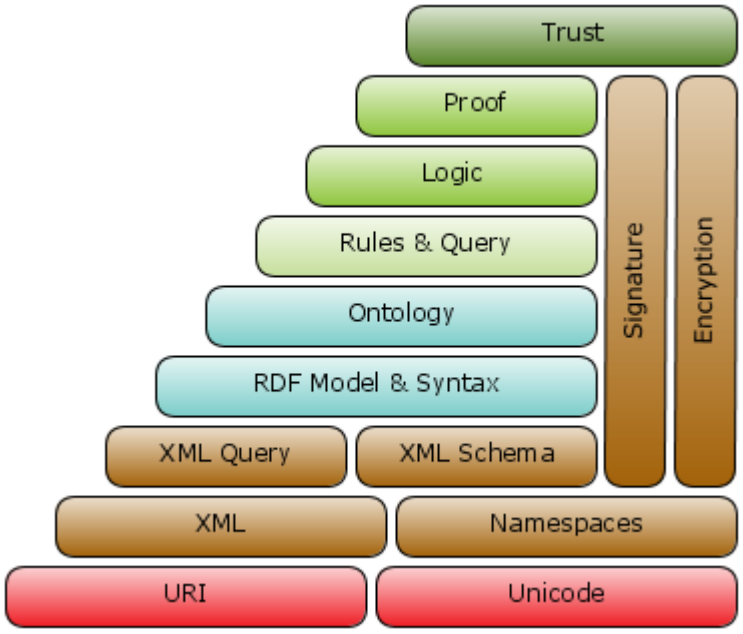

Рисунок 3.2 — Стек понятий семантической паутины (заимствовано из источника [3.1.4])

В плане преемственности моделей, в [3.1.7] отмечается, что: «История появления RDF имеет своё начало в 1990 году, когда Тим Бернерс-Ли предложил отмечать типы ссылок между документами для облегчения автоматической обработки. Типы ссылок однако не были включены в первую спецификацию HTML, но идея была подхвачена в системе описания метаданных MCF. Обобщённое представление метаданных нашло отражение в рекомендации W3C по RDF в 1999 году. С тех пор стандарты RDF развиваются, появляются новые средства для работы с RDF. …

Множество RDF-утверждений образует ориентированный граф, в котором вершинами являются субъекты и объекты, а рёбра отображают отношения.

RDF сам по себе является не форматом файла, а только лишь абстрактной моделью данных, то есть описывает предлагаемую структуру, способы обработки и интерпретации данных. Для хранения и передачи информации, уложенной в модель RDF, существует целый ряд форматов записи.

Для обработки RDF-данных предлагается реализовать языки запросов: SPARQL (стандарт W3C), RQL, RDQL. ...».

И так, в модели семантической паутины среда описания ресурсов (RDF) создается наборами элементарных отношений (триплетов), как это показано на рисунке 3.3. Поскольку сам триплет легко представляется направленным графом, то модель RDF обеспечивает описание семантической паутины.

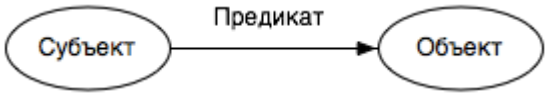

Рисунок 3.3 — Триплет RDF (заимствовано из источника [3.1.7])

В феврале 2004 года, организация W3C опубликовала набор рекомендаций по модели RDF, которые постоянно обновляются и служат основой развития идей и программных средств Всемирной паутины. С русскоязычным переводом одного из таких документов («Среда Описания Ресурса (RDF): Понятия и Абстрактный Синтаксис») можно познакомится по сылке [3.1.8].

Поскольку реальное манипулирование данными, отражающими семантические отношения на языке XML, довольно объемны и затруднительны, на практике используются различные инструментальные средства, поддерживающие правильный синтаксис и семантику обрабатываемых документов. Одним из таких свободно распространяемых инструментальных средств является система Protege, разработанная «Стендфортским центром исследований биометрической информатики» [3.1.9]: «**Protege** — это свободный, открытый редактор онтологий и фреймворк для построения баз знаний.

Платформа Protege поддерживает два основных способа моделирования онтологий посредством редакторов Protégé-Frames и Protégé-OWL. Онтологии, построенные в Protégé, могут быть экспортированы во множество форматов, включая RDF (RDF Schema), OWL и XML Schema.

Protégé имеет открытую, легко расширяемую архитектуру за счёт поддержки модулей расширения функциональности.

Protégé поддерживается значительным сообществом, состоящим из разработчиков и учёных, правительственных и корпоративных пользователей, использующих его для решения задач, связанных со знаниями, в таких разнообразных областях, как биомедицина, сбор знаний и корпоративное моделирование.

Protégé доступен для свободного скачивания с официального сайта вместе с плагинами и онтологиями. ...».

3.1.2.2 Установка редактора онтологий - Protege

Как представлено на сайте «Protege Wiki» [3.1.10], система Protege разрабатывается достаточно давно и имеется множество ее версий. Например, версия 1.1 датируется ноябрем 1999 года. Соответственно, в сети Internet можно найти множество ее описаний и рекомендаций работы с ней. Но анализ этих описаний показывает, что интерфесы взаимодействия самой системы и пользователя существенно изменяются от версии к версии. В такой ситуации, чтобы обеспечить учебный процесс, необходима краткая инструкция по установке конкретной версии системы.

На момент на написания данного учебного пособия, для скачивания дистрибутива Protege была доступна версия 5.2.0, которую можно найти по адресу [3.1.11]. Сам дистрибутив имеет сжатый размер порядка 107 Мегабайт и разворачивается в объеме 217 Мегабайт дискового пространства. В нем также содержится runtime-система виртуальной машины Java, обеспечивающая автономную установку всего программного обеспечения в целом.

Сохраняя технологическую традицию самого процесса обучения, ориентированного на использование обучающей операционной среды УПК АСУ [5], для

целей установки выбран дистрибутив *protege-5.2.0-5-any.pkg.tar.xz*, входящий в состав ПО ОС ArchLinux. Далее, описан пошаговый процесс настройки, тестирования и запуска системы Protege применительно к этому дистрибутиву и среды обучения.

Шаг 1. Запуск среды обучения ОС УПК АСУ.

Аспирант выполняет компьютере типовые действия по запуску обучающей операционной среды УПК АСУ с использованием личного flashUSB и согласно инструциям, описанным в методическом руководстве [5]:

1) проводится ***login*** пользователем ***asu***;
2) в среде пользователя ***asu***, подключается личная рабочая область аспиранта;
3) осуществляется выход из сессии пользователя ***asu*** и вход в сессию пользователем ***upk***.

Шаг 2. Проверка программной инфраструктуры системы Protege.

Для нормального запуска и работы системы Protege, в обучающей среде УПК АСУ должна присутствовать следующая инфраструктура:

1) в директории ***/run/basefs/asu64upk/opt/*** должен находиться файл ***protege.sfs***;
2) в рабочей директории пользователя ***asu*** (*/home/upk/*) должны иметься директории ***/Mastro-Protege-plugin*** и ***ontologies***;
3) в директории ***/homeupk/bin/*** должен присутствовать файл сценария ***protege***, содержимое которого приведено на листнге 3.1.

Листинг 3.1 — Содержимое сценария /home/upk/bin/protege

```
#!/bin/sh

if [ ! -f /opt/protege/run.sh ]; then
    sudo mount /run/basefs/asu64upk/opt/protege.sfs /opt/protege
fi

if [ ! -f /opt/protege/run.sh ]; then
    echo "Не могу подмонтировать систему protege..!"
    echo "Для завершения, - нажми клавишу <Enter>"
    read aa;
    exit 1;
fi

cd /opt/protege


CMD_OPTIONS="-Dapple.laf.useScreenMenuBar=true \
    -Dswing.defaultlaf=javax.swing.plaf.metal.MetalLookAndFeel \
    -Dawt.useSystemAAFontSettings=on \
    -Dswing.aatext=true \
```

```
    -Dsun.java2d.xrender=true" ./run.sh "$@"...
```

Конец листинга 3.1

Замечание

Если аспирант заметил нарушения в приведенной выше инфраструктуре системы Protege, ему следует, самостоятельно или с помощью преподавателя, устранить имеющиеся недостатки.

Шаг 3. Запуск и первоначальная настройка системы Protege.

Для запуска системы Protege аспиранту необходимо открыть окно терминала и набрать команду ***protege***.

В результате появится окно системы, показанное на рисунке 3.4.

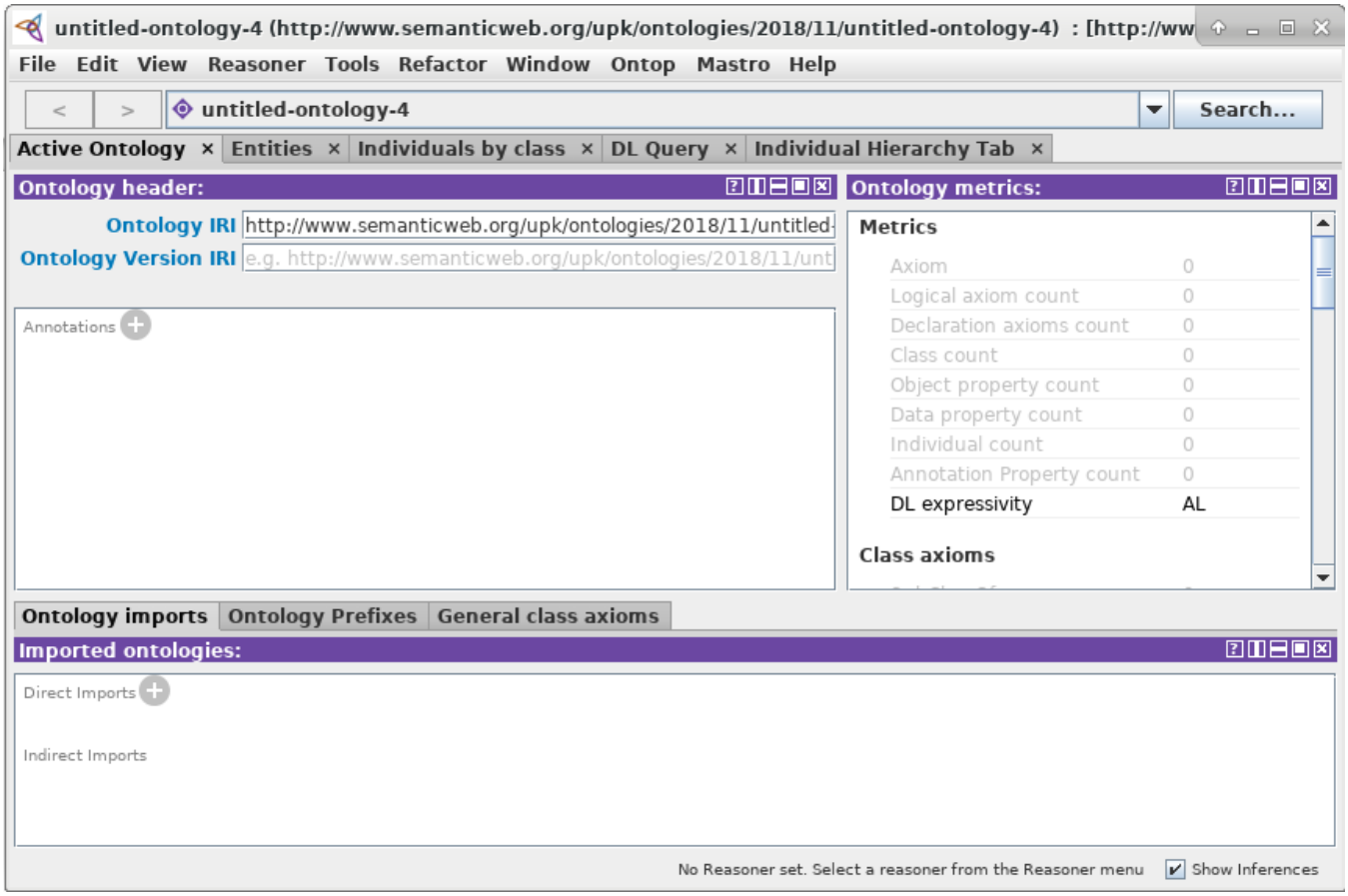

Рисунок 3.4 — Окно системы Protege после запуска

Замечание

При первом запуске потребуется ввести пароль пользователя ***upk***.

В целом, а рабочей среде пользователя ***upk*** уже проведены ряд настроек системы, поэтому следует открыть уже существующий проект онтологии ***test1.owl***.

Выполняется это следующим образом: следует выбрать в меню системы «***File*→*Open...***». В результате, появится окно диалога, в котором следует выбрать файл ***test1.owl***, как показано на рисунке 3.5. Также изменится IRI системы, как показано на рисунке 3.6.

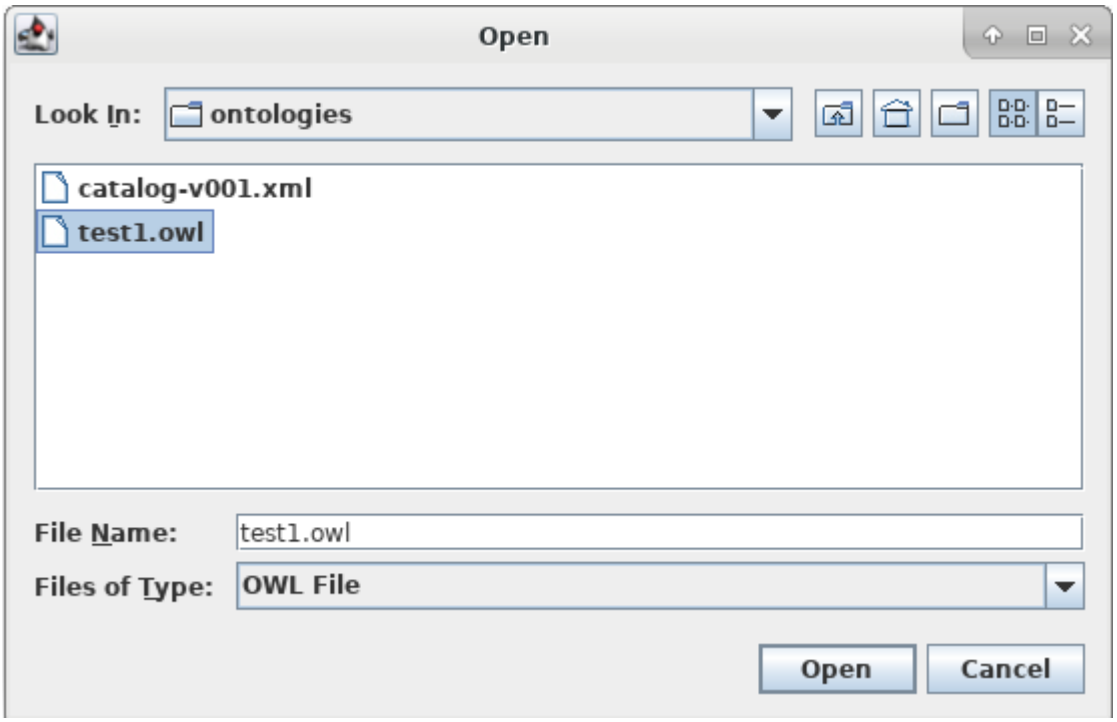

Рисунок 3.5 — Открытие онтологии test1.owl

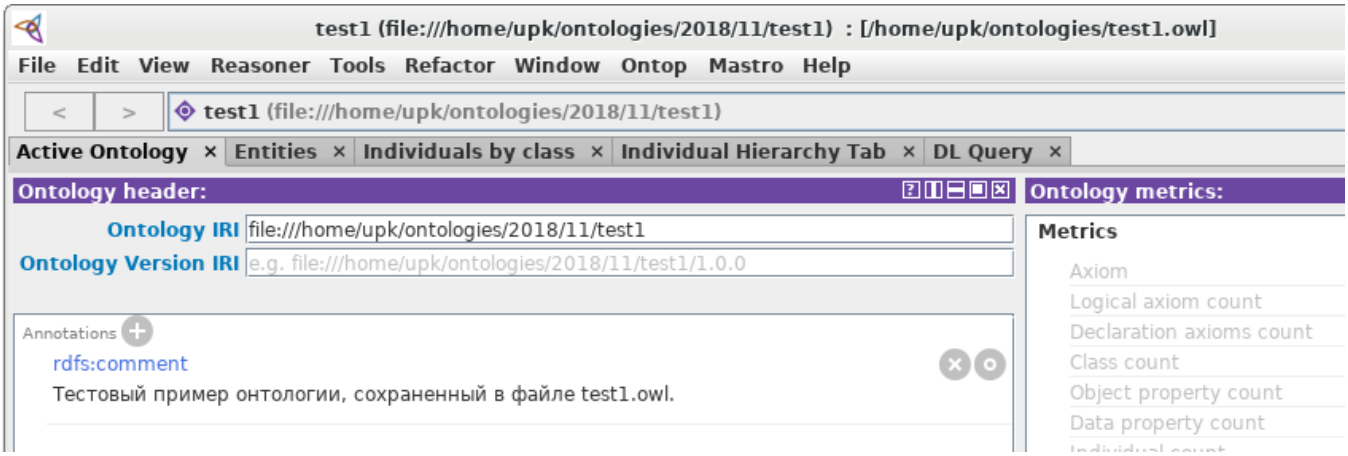

Рисунок 3.6 — Окно системы Protege после открытия проекта test1.owl

Замечание

После открытия конкретной онтологии можно проводить ее модификацию и сохранять изменения комбинацией клавишь «Ctrl-S» или из меню: «File->Save».

По результатам указанных выше действий, все изменения сохранены в двух файлах: ***/home/upk/ontologies/catalog-v001.xml*** и ***/home/upk/ontologies/test1.owl***.

Содержимое файла ***catalog-v001.xml*** приведено на листинге 3.2. Его назначение — обеспечивать ссылки на проекты онтологий.

Листинг 3.2 — Содержимое файла catalog-v001.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<catalog prefer="public" xmlns="urn:oasis:names:tc:entity:xmlns:xml:catalog">
    <group id="Folder Repository, directory=, recursive=true, Auto-Update=true,
version=2" prefer="public" xml:base="">
        <uri id="Automatically generated entry, Timestamp=1545486943303"
name="file:///home/upk/ontologies/2018/11/test1" uri="test1.owl"/>
    </group>
</catalog>
```

Конец листинга 3.2

Содержимое файла ***test1.owl*** приведено на листинге 3.2. Его назначение — описание конкретной онтологии.

Листинг 3.3 — Содержимое файла test1.owl

```
<?xml version="1.0"?>
<rdf:RDF xmlns="file:///home/upk/ontologies/2018/11/test1#"
     xml:base="file:///home/upk/ontologies/2018/11/test1"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xml="http://www.w3.org/XML/1998/namespace"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
    <owl:Ontology rdf:about="file:///home/upk/ontologies/2018/11/test1">
        <rdfs:comment>Тестовый пример онтологии, сохраненный в файле
test1.owl.</rdfs:comment>
    </owl:Ontology>
</rdf:RDF>

<!-- Generated by the OWL API (version 4.2.8.20170104-2310)
https://github.com/owlcs/owlapi -->
```

Конец листинга 3.3

Из листинга 3.3 хорошо видно, что содержимое файла ***test1.owl*** представляет собой содержимое документа на языке XML в формате RDF, причем описание самой онтологии ограничено только общим коментарием.

По мере разработки самой онтологии в файл ***test1.owl*** будут добавляться новые записи, отражающие новые свойства самой онтологии, но их содержимое и назначение является темой следующего практического занятия.

3.1.2.3 Учебные задания

Аспиранту следует внимательно изучить формат описания ресурсов RDF, минимум по источникам [3.1.7] и [3.1.8].

Провести установку редактора онтологий Protege, как это было описано в предыдущем пункте.

Разобраться с содержимым файлов ***catalog-v001.xml*** и ***test1.owl***, представленных на листингах 3.2 и 3.3, а также отобразить проведенную работу в личном отчете.

В заключение, следует создать новую онтологию с именем ***test2.owl*** и сохранить ее в рабочей области. Для этого, при запущенной системе Protege, необходимо выполнить следующие шаги.

Шаг 1. Сохранить текущую антологию из меню системы «***File*** → ***Save***».

Шаг 2. Из меню системы «***File*** → ***New...***» инициировать новую онтологию и отредактировать поле «***Ontology IRI***», как показано на рисунке 3.7.

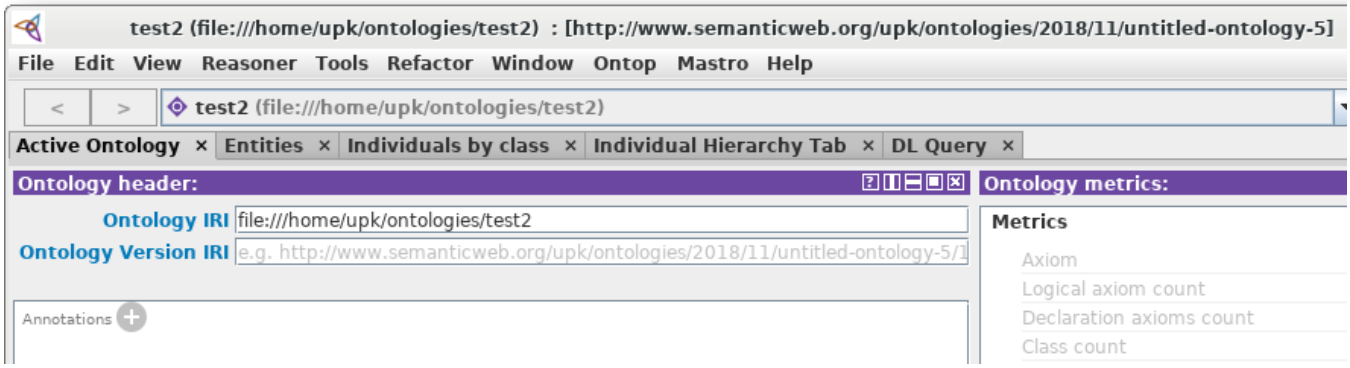

Рисунок 3.7 — Редактирование поля «*Ontology IRI*»

Шаг 3. Из меню системы «***File*** → ***Save as...***» сохранить результат, выполненный на шаге 2, в файле ***test2.owl***.

После сохранения новой онтологии, следует провести анализ файлов директории ***~/ontologies/*** и описать в личном отчете обнаруженные изменения.

3.1.3 Список использованных источников

3.1.1 Искусственный интеллект — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/Искуственный_интеллект). Проверено: 04.12.2018.

3.1.2 Интеллектуальная система — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/Интеллектуальная_система). Проверено: 04.12.2018.

3.1.3 База знаний — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/База_знаний). Проверено: 19.12.2018.

3.1.4 Семантическая паутина — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/Семантическая_паутина). Проверено: 19.12.2018.

3.1.5 Инженерия знаний — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/Инженерия_знаний). Проверено: 19.12.2018.

3.1.6 Семантическая сеть — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/Семантическая_сеть). Проверено: 19.12.2018.

3.1.7 Resource Description Framework — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/Resource_Description_Framework). Проверено: 19.12.2018.

3.1.8 Среда Описания Ресурса (RDF): Понятия и Абстрактный Синтаксис: [Электронный документ]. - (https://www.w3.org/2007/03/rdf_concepts_ru/). Проверено: 19.12.2018.

3.1.9 Protege — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/Protege). Проверено: 19.12.2018.

3.1.10 Protege Desktop Older Versions — Protege Wiki: [Электронный документ]. - (https://protegewiki.stanford.edu/wiki/Protege_Desktop_Old_Versions). Проверено: 19.12.2018.

3.1.11 Release Protege 5.2.0: [Электронный документ]. - (https://github.com/protegeproject/protege-distribution/releases/tag/v5.2.0). Проверено: 19.12.2018.

3.2 Практическая работа №8 «Язык описания онтологий (OWL)»

Цель работы: получение навыков работы с языком описания онтологий для Всемирной семантической паутины (OWL) в среде системы Protege.

3.2.1 Самостоятельная работа

Требования, предъявляемые к системам представления и обработки знаний. Формализмы, основанные на классической и математической логиках. Современные логики. Фреймы. Семантические сети и графы. Модели, основанные на прецедентах. Приобретение и формализация знаний. Пополнение знаний. Обобщение и классификация знаний. Логический вывод и умозаключение на знаниях.

Литература: [12 - 16].

3.2.2 Порядок выполнения работы

Одна из главных особенностей баз знаний — отражение онтологии о предметной области исследования. Очевидно, что определение самого термина онтология дается в различных аспектах. Например, информационные науки оперируют термином высшая онтология [3.2.1]: «В информационной науке **высшей онтологией** (англ. *upper ontology*) называется онтология, включающая очень общие термины — например, «объект», «свойство», «реляция» — актуальные для всех предметных областей. Важная функция высшей онтологии заключается в поддержке семантической интероперабельности большого числа предметно-ориентированных онтологий. Поддержка предполагает создание общей стартовой точки для формулирования определений. Термины предметно-ориентированных онтологий подчинены терминам высшей.

Предложено несколько проектов высших онтологий. Всякая высшая онтология может считаться компьютерной реализацией натурфилософии — эмпирического метода исследований в рамках онтологии физической.

Системы библиотечной классификации предвосхитили появление высших онтологий. Несмотря на то, что библиотечные классификации организуют и категоризуют знание на основе общих концепций, применимых ко всем предметным областям, системы не заменяют друг друга. ...».

Что касается информатики, то она отражает следующий аспект онтологии [3.2.2]: «**Онтоло́гия** в информатике (новолат. *ontologia* от др.-греч. ὤν род. п. ὄντος — сущее, то, что существует и λόγος — учение, наука) — это попытка всеобъемлющей и подробной формализации некоторой области знаний с помощью концептуальной схемы. Обычно такая схема состоит из структуры данных, содержащей все релевантные классы объектов, их связи и правила (теоремы, ограничения),

принятые в этой области. Этот термин в информатике является производным от древнего философского понятия «онтология».

Онтологии используются в процессе программирования как форма представления знаний о реальном мире или его части. Основные сферы применения — моделирование бизнес-процессов, семантическая паутина (англ. *Semantic Web*), искусственный интеллект. ...».

В данной практической работе мы проведем краткое общее изучение специального языка OWL (*Web Ontology Language*), а также создадим ряд конкретных онтологий в системе Protege.

3.2.2.1 История вопроса

Википедия дает следующее определение языка OWL [3.2.3]: «**OWL** (англ. *Web Ontology Language*) — язык описания онтологий для семантической паутины. Язык OWL позволяет описывать классы и отношения между ними, присущие веб-документам и приложениям. OWL основан на более ранних языках OIL и DAML+ OIL и, в настоящее время, является рекомендованным консорциумом Всемирной паутины.

В основе языка — представление действительности в модели данных «объект — свойство». OWL пригоден для описания не только веб-страниц, но и любых объектов действительности. Каждому элементу описания в этом языке (в том числе свойствам, связывающим объекты) ставится в соответствие URI. ...».

В настоящий момент актуальной версией является OWL 2, опубликованный в рекомендациях W3C от 27 октября 2009 года. В частности, документ [3.2.4] указывает на стандартные блоки языка OWL 2, показанные на рисунке 3.8. В нем же можно найти ссылки на другие официальные документы W3C.

Следует отметить, что основа языка OWL была разработатна, документально оформлена в версии 1.1 и опубликована, как рекомендации W3C, 10 февраля 2004 года. Например, в документе [3.2.5] отмечается, что «… Semantic Web - это взгляд в будущее Сети, в которой информация наделена явным значением, облегчая автоматическую машинную обработку и интеграцию информации, доступной в Сети. Semantic Web будет основываться на способности XML определять настраиваемые схемы разметки и гибкий подход RDF к представлению данных. Первый уровень над RDF, требуемый для Semantic Web - это язык онтологий, который может формально описать значение терминов, используемых в веб-документах. Если, как предполагается, на машины возложить полезные рассудочные задачи в отношении этих документов, то данный язык должен выйти за пределы основной семантики RDF Схем. ...

OWL был разработан, чтобы удовлетворить эту потребность в Языке Веб-Онтологий. OWL - часть растущего комплекта рекомендаций W3C, связанных с Semantic Web.

- **XML** обеспечивает синтаксис для структурированных документов, но не налагает никаких семантических ограничений на значение этих документов.

• **XML Schema** определяет структуру документов XML, а также дополняет XML конкретными типами данных.

• **RDF** позволяет описать модель данных (datamodel) для объектов ("ресурсов") и отношения между ними, обеспечивает простую семантику для этой моделей данных, представляя их в XML синтаксисе.

• **RDF Schema** предоставляет средства для описания свойств и классов RDF-ресурсов, а также семантику для иерархий-обобщений таких свойств и классов.

• **OWL** добавляет еще больше возможностей для того, чтобы описать свойства и классы: в частности, отношения между классами (например, непересекаемость), кардинальность (например, "точно один"), равенство, больше типов свойств, характеристик свойств (например, симметрия), и перечисляемые классы. ...».

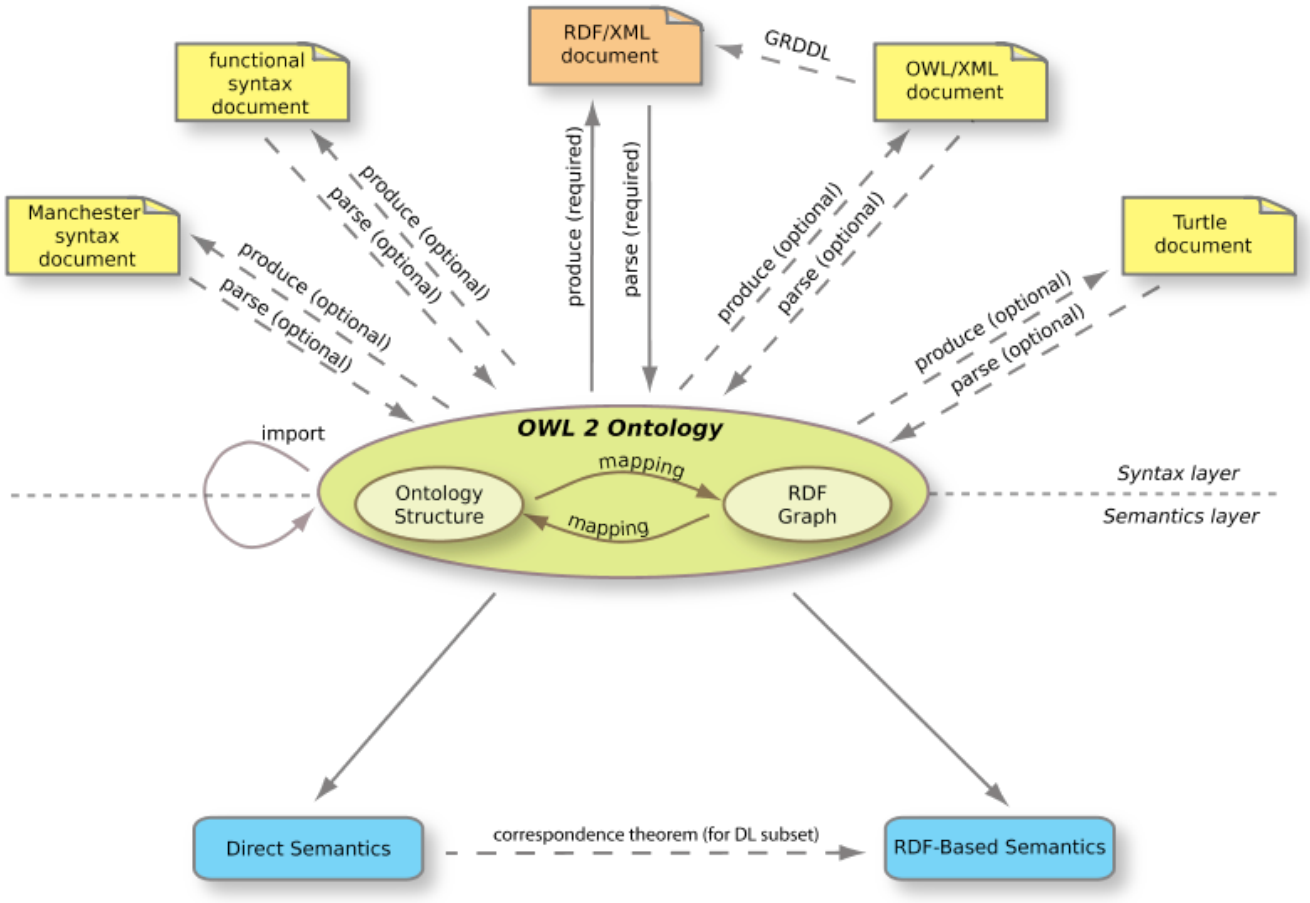

Рисунок 3.8 — Структура OWL 2 (заимствовано из источника [3.2.4])

В целом, язык OWL поддерживает несколько диалектов, отличающихся различными возможностями, но все они опираются на семантику RDF Schema. Для простейших предметно-ориентированных технологий вполне достаточно средств диалекта OWL Lite [3.2.5]: «… В OWL Lite включены следующие понятия RDF Schema.

• ***Class***: Класс определяет группу индивидов, которых объединяет наличие

некоторых общих свойств. Например, Дебора и Франк - оба члены класса Человек. Классы могут быть организованы в иерархии, используя. Есть встроенный самый общий класс по имени Thing (Вещь), который является классом всех индивидов и суперкласс для всех OWL классов. Есть также встроенный наиболее специфичный класс по имени Nothing (Ничто), который не имеет никаких представителей и является подклассом всех OWL классов.

• ***rdf:subClassOf***: Иерархии классов могут быть созданы, путем одного или нескольких утверждений, что данный класс - подкласс другого класса. Например, класс Человек может быть представлен как подкласс класса Млекопитающие. Из этого можно вывести что, если индивид - Человек, то он также Млекопитающее.

• ***rdf:Property***: Свойства могут использоваться, чтобы установить отношения между индивидами или от индивидов к значениям данных. Возьмем, например, четыре свойства: имеетРебенка, имеетРодственника, имеетБрата и имеетВозраст. Первые три могут использоваться, чтобы связать представителя класса Человек с другими представителями класса Человек (и, таким образом, являются Свойствами-объектами ObjectProperty), а последнее (имеетВозраст) может использоваться, чтобы связать представителя класса Человек с представителем типа данных Целое число (и, таким образом, является Свойством-значением DatatypeProperty). И owl:ObjectProperty, и owl:Datatype Property - подклассы RDF класса rdf:Property.

• ***rdfs:subPropertyOf***: Иерархии свойств могут быть созданы путем одного или нескольких утверждений, что данное свойство - подсвойство одного или нескольких других свойств. Например, свойство имеетБрата может быть представлено как подсвойство имеетРодственника. Из этого можно вывести что, если индивид связан с другим свойством имеетБрата, то это также связывает его с другим свойством имеетРодственника.

• ***rdfs:domain***: Область определения (домен) свойства ограничивает индивидов, к которым это свойство может быть применено. Если свойство связывает индивида с другим индивидом, и это свойство имеет какой-то класс в качестве одного из своих доменов, то индивид должен принадлежать этому классу. Например, может быть заявлено, что свойство имеетРебенка имеет область определения Млекопитающие. Из этого можно вывести что, если Франк имеетРебенка Анна, то Франк должен быть Млекопитающим. Заметьте, что *rdfs:domain* называют глобальным ограничением, так как ограничение накладывается на все свойство, а не только на свойство, связанное с конкретным классом. См. также обсуждение ограничений свойств ниже.

• ***rdfs:range***: Диапазон свойства, ограничивающий индивидов, которые могут выступать в качестве значений этого свойства. Если свойство связывает одного индивида с другим, и это свойство имеет класс в качестве своего диапазона, то другой индивид должен принадлежать этому классу. Например, для свойства имеетРебенка может быть заявлен диапазон Млекопитающие. Из этого можно вывести что, если Луиза связана с Деборой свойством имеетРебенка, (т.е., Дебора - ребенок Луизы), то Дебора относится к Млеко-

питающим. Диапазон - также глобальное ограничение, как и вышеизложенный домен. См. также обсуждение локальных ограничений ниже (напр. AllValuesFrom).

• ***Individual***: Индивиды - это представители классов, и чтобы связать одного индивида с другим могут использоваться свойства. Например, индивид по имени Дебора может быть описан как представитель класса Человек, и свойство имеетРаботодателя может быть использовано, чтобы связать индивида Дебору с индивидом Стенфордский Университет. ...».

Для изучения других особенностей языка OWL можно порекомендовать перевод руководства по этому языку, опубликованному в источнике [3.2.6].

Очень хорошим руководством по применению OWL и инструментальных средств его поддерживающих является методическое руководство [3.2.7].

Мы, на данном практическом занятии, освоим только описание простейших учебных предметно-ориентированных онтологий, реализуемых средствами системы Protege.

3.2.2.2 Пример постронения онтологии в системе Protege

Продемонстрируем технологию построения предметно-ориентированной онтологии на следующем примере:

- имеется объект высшего онтологического уровня с именем «Кафедра АСУ», который включает в себя два класса объектов: «Сотрудники кафедры» и «Учебные дисциплины»;
- класс «Сотрудники кафедры» содержит набор индивидуальных объектов, каждый из которых характеризуется строкой вида «Фамилия Имя Отчество», а также отношением к некоторым объектам класса «Учебные дисциплины» типа «может преподавать»;
- класс «Учебные дисциплины» содержит набор индивидуальных объектов, каждый из которых характеризуется строкой вида «Название дисциплины», а также может иметь отношение к конкретному объекту класса «Сотрудники кафедры» типа «преподается»;
- отношение «преподается» детализируется на следующие составляющие: «читаются лекции», «проводятся лабораторные работы», «проводятся практические занятия»;
- на основании указанных ограничений построить онтологию объекта «Кафедра АСУ» в инструментальной среде Protege.

Прежде чем приступить к реализации данного примера в системе Protege, нам следует учесть, что как ранее было отмечено в [3.2.3] относительно языка OWL: «… Каждому элементу описания в этом языке (в том числе свойствам, связывающим объекты) ставится в соответствие URI. ...». Поэтому учтем следующую информацию относительно URI [3.2.8]: «**URI** (англ. *Uniform Resource Iden-*

tifier) — унифицированный (единообразный) идентификатор ресурса. По-русски иногда говорят [*ýpu*]. URI — последовательность символов, идентифицирующая абстрактный или физический ресурс. Ранее назывался **Universal Resource Identifier** — универсальный идентификатор ресурса. …

URL стал фундаментальным нововведением в Интернете, поэтому принципы URI документально закреплялись так, чтобы обеспечить полную совместимость с URL. Отсюда появился и большой недостаток URI, пришедший как наследство от URL. В URI, как и в URL, можно использовать только ограниченный набор латинских символов и знаков препинания (даже меньший, нежели в ASCII). Иными словами, если мы захотим использовать в URI символы кирилицы, или иероглифы, или, скажем, специфические символы французкого языка, то нам придётся кодировать URI таким же образом, каким в Википедии кодируются URL с символами Юникода. ...».

Таким образом, для построения онтологии, которая могла бы обрабатываться средствами Всемироной паутины и была бы легко пониматься человеком, необходимо провести кодировку ресурсов предметной области в виде отдельных слов, написанных допустимыми для URI символами кодировки ASCII. И только после такой перекодировки предметной области можно будет воспользоваться средствами инструментальной системы Protege для реализации заданной в примере онтологии.

Дальнейшие рассуждения проведем в виде последовательности шагов, разделяющих отдельные этапы создания онтологии.

Шаг 1. Кодирование ресурсов онтологии предметной области.

Кодирование ресурсов онтологии предметной области проведем в виде сокращенных составных слов английского языка, обозначающего ресурсы нашего примера, заданные в терминах русского языка. Результат такой кодировки представлен в таблице 3.1.

Таблица 3.1

Ресурс предметной области	***Кодировка классаов и отношений***
Кафедра АСУ	`DepASU`
Сотрудники кафедры	`DepStaff`
«Название N-го объекта класса Сотрудники кафедры»	`EmpN`
может преподавать	`canTeach`
Учебные дисциплины	`AcadDis`
«Название N-го объекта класса Учебные дисциплины»	`DiscN`
преподается	`taught`
читаются лекции	`lecturesWork`
проводятся лабораторные работы	`laboratoryWork`
проводятся практические занятия	`practicalWork`

Шаг 2. Создание начального ресурса онтологии ***DepASU***.

Следует запустить систему Protege и на вкладке «Active Ontology» отредактировать поле «Ontology IRI», как показано на рисунке 3.9.

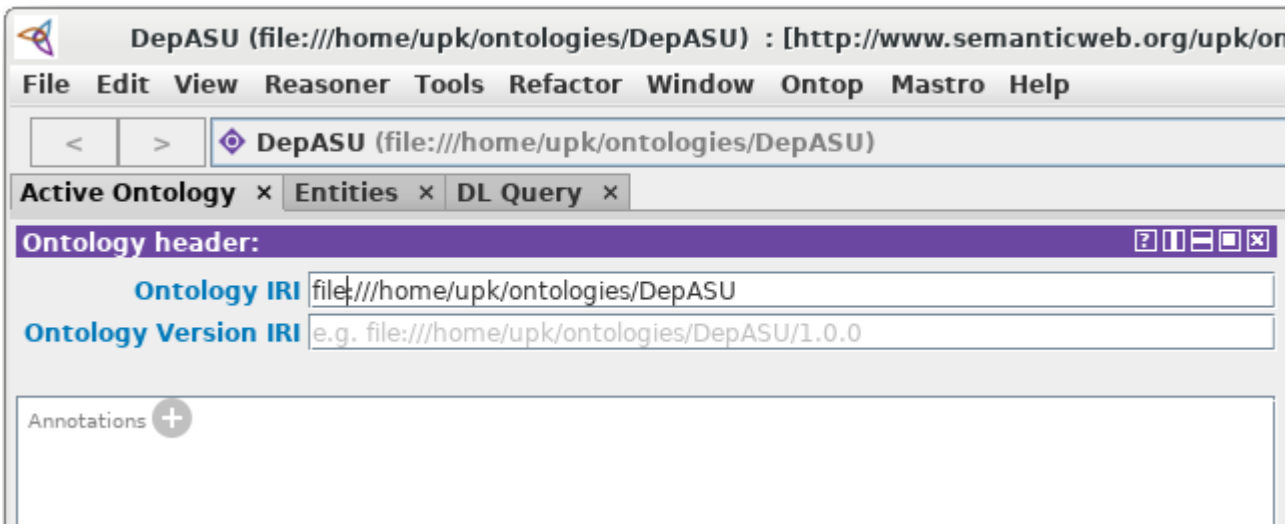

Рисунок 3.9 — Установка поля «Ontology IRI» системы Protege для онтологии DepASU

На панели меню системы выбрать «***File*** → ***Save as...***», после чего появится окно выбора формата сохранения онтологии, как показано на рисунке 3.10.

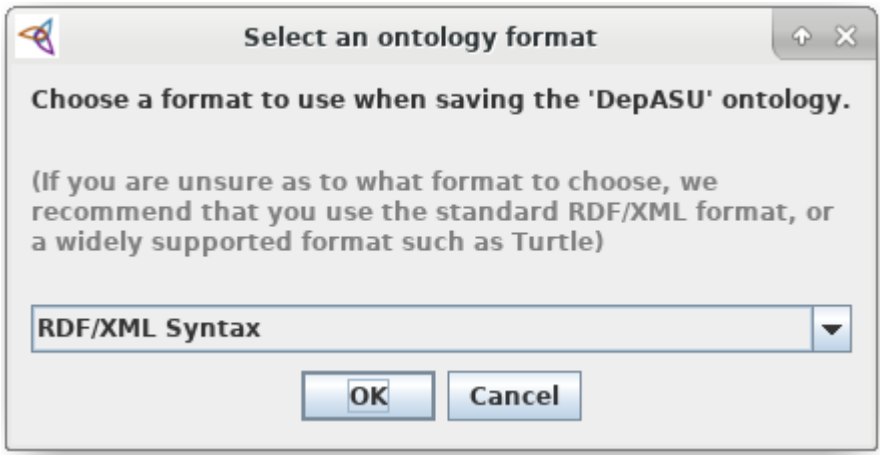

Рисунок 3.10 — Окно выбора формата сохранения онтологии

После активации кнопки «***OK***», появтся диалоговое окно, показанное на рисунке 3.11, в котором следует указать имя файла для сохранаяемой онтологии (в нашем случае - DepASU).

После активации кнопки «***Save***», начальное значение описания нашей онтологии сохранится в файле ***/home/upk/ontologies/DepASU.owl*** (в формате синтаксиса RDF/XML).

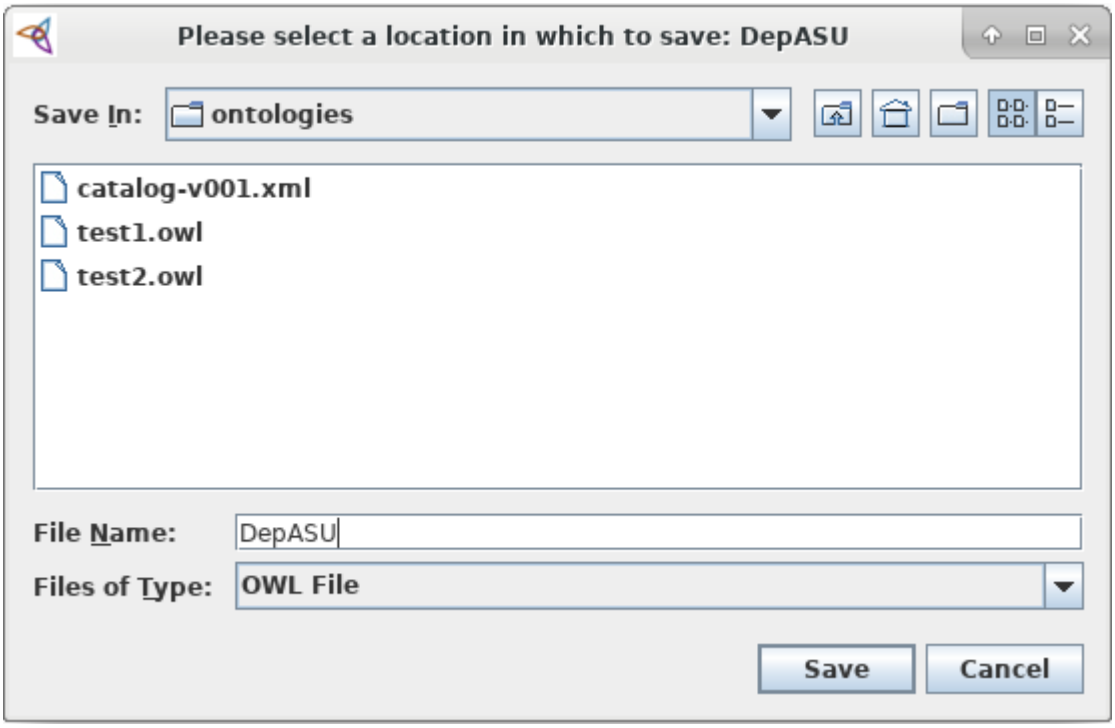

Рисунок 3.11 — Окно выбора имени файла для сохранения онтологии

После всех указанных действий, онтологию DepASU можно будет:

- подключать из главного меню: «***File*** → ***Open***»;
- сохранять изменения комбинацией клавишь «***Ctrl-s***».

Шаг 3. Создание класса DepASU и иерархии его классов.

В системе Protege следует перейти на вкладку «***Entities*** → ***Classes***», как показано на рисунке 3.12.

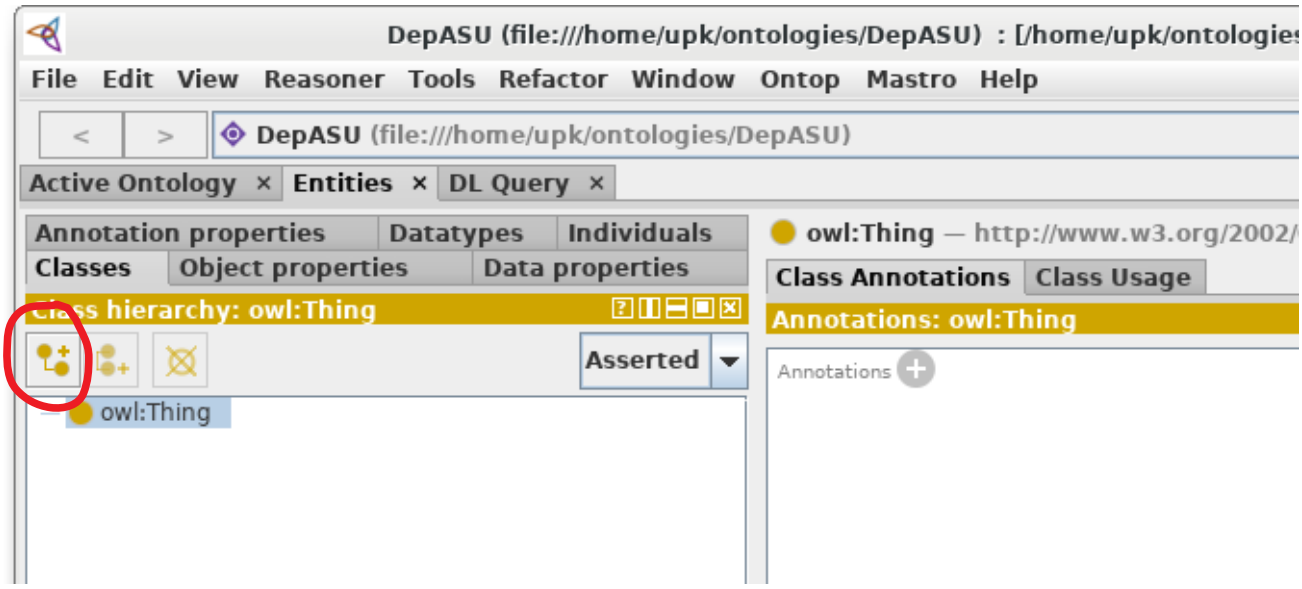

Рисунок 3.12 — Выбор вкладки для создания классов

Используя кнопку создания вложенных классов, показанную красной обводкой, построить иерархию из классов DepASU, DepStaff и AcadDis, как показано на рисунке 3.13.

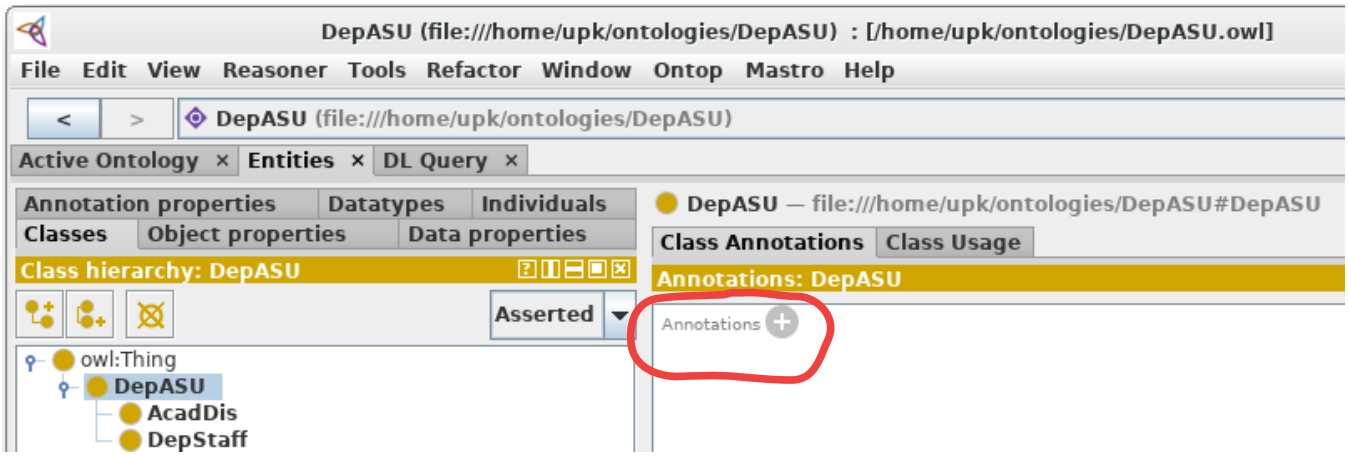

Рисунок 3.13 — Результат создания классов DepASU, DepStaff и AcadDis

Шаг 4. Создание комментариев и литеральный представлений классов.

Поскольку работать с кодировками классов и отношений — неудобно, то воспользуемся инструментальным средством «***Annotations*** +», обведенным красной линией на рисунке 3.13.

Соответственно, на рисунке 3.14 показано добавление литерала «Кафедра АСУ» типа rdfs:label для класса DepASU.

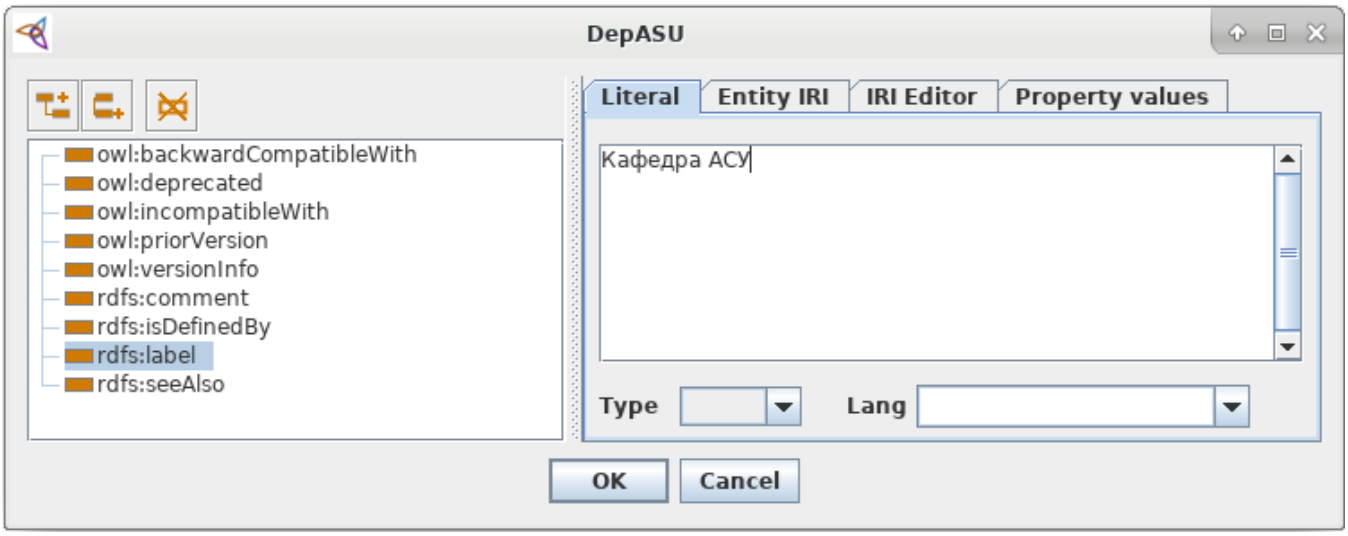

Рисунок 3.14 — Добавление литерала типа rdfs:label для класса DepASU

Дополнительно, рекомендуется добавлять литералы типа rdfs:comment, раскрывающие содержательную часть используемых кодировок.

На рисунке 3.15 показан результат добавления литералов типа rdfs:label и rdfs:comment для иерархии классов DepASU, DepStaff и AcadDis.

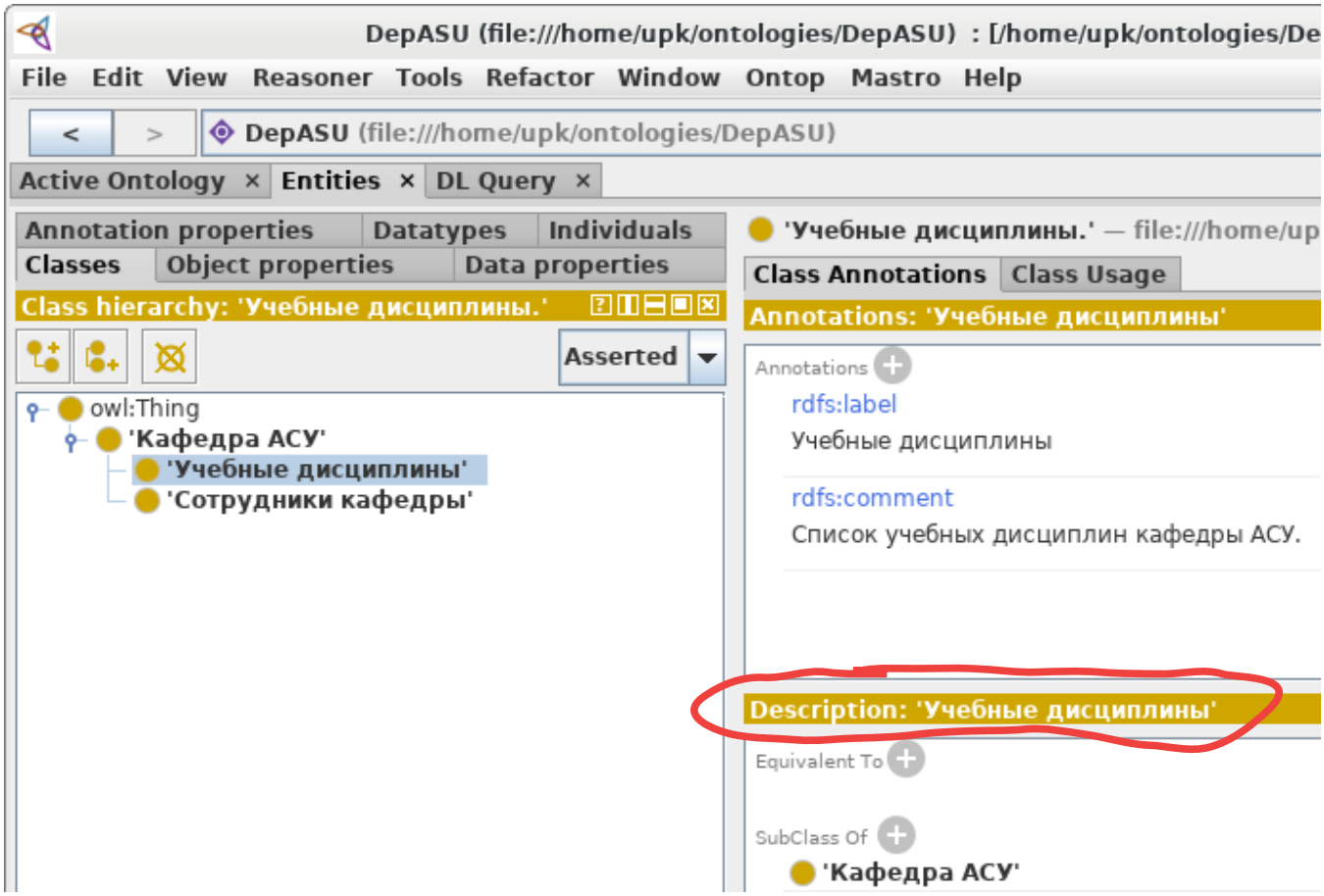

Рисунок 3.15 — Результат создания иерархии классов DepASU, DepStaff и AcadDis, после добавления литералов типа rdfs:label и rdfs:comment

Шаг 5. Создание объектов класса «Учебные дисциплины».

Добавим в указанный класс пять объектов согласно данным, представленным в таблице 3.2.

Таблица 3.2

Кодировка объекта	***Наименование дисциплины***
Disc1	Операционные системы
Disc2	Современные операционные системы
Disc3	Базы данных
Disc4	Программирование
Disc5	Архитектура вычислительных комплексов

Для этого, выделим, как показано на рисунке 3.15, «Учебные дисциплины», а в окне «Description: Учебные процессы» найдем кнопу «**Instances +**» и активируем ее.

В появившемся окне, показанном на рисунке 3.16, активируем выделенную красной линией кнопку «***Add individual***» и в новом появившемся окне, показанном на рисунке 3.17, вписываем значение кодировки дисциплины ***Disc1***.

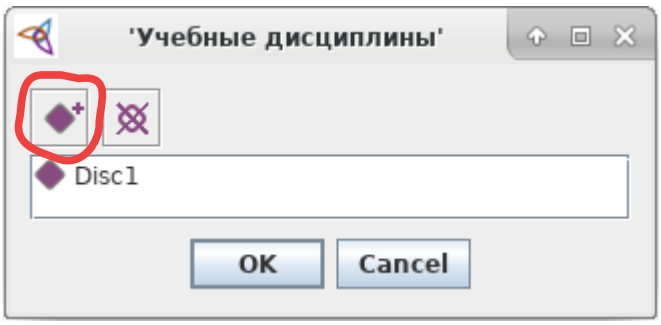

Рисунок 3.16 — Добавление кодировок объектов класса «Учебные дисциплины»

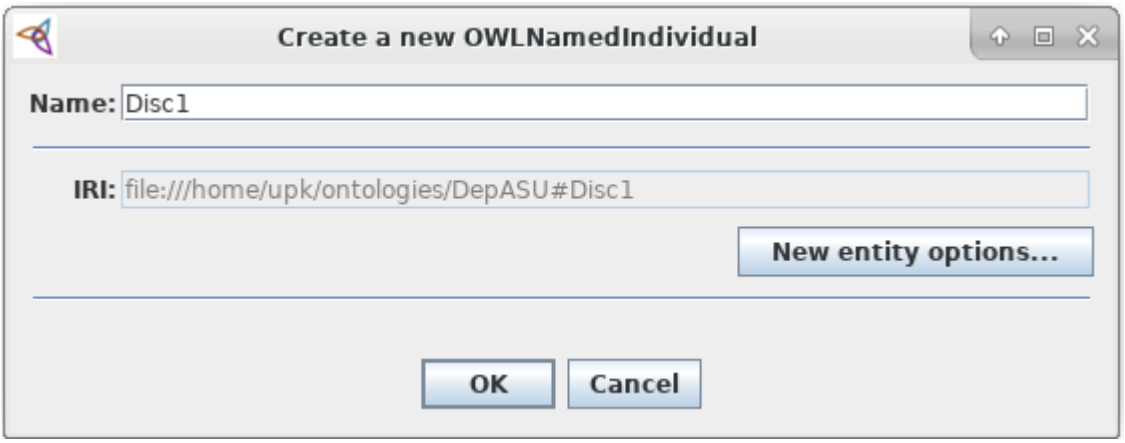

Рисунок 3.17 — Ввод имени кодировки объекта

Повторяя приведенную выше процедуру еще четыре раза, мы получаем результат, показанный на рисунке 3.18.

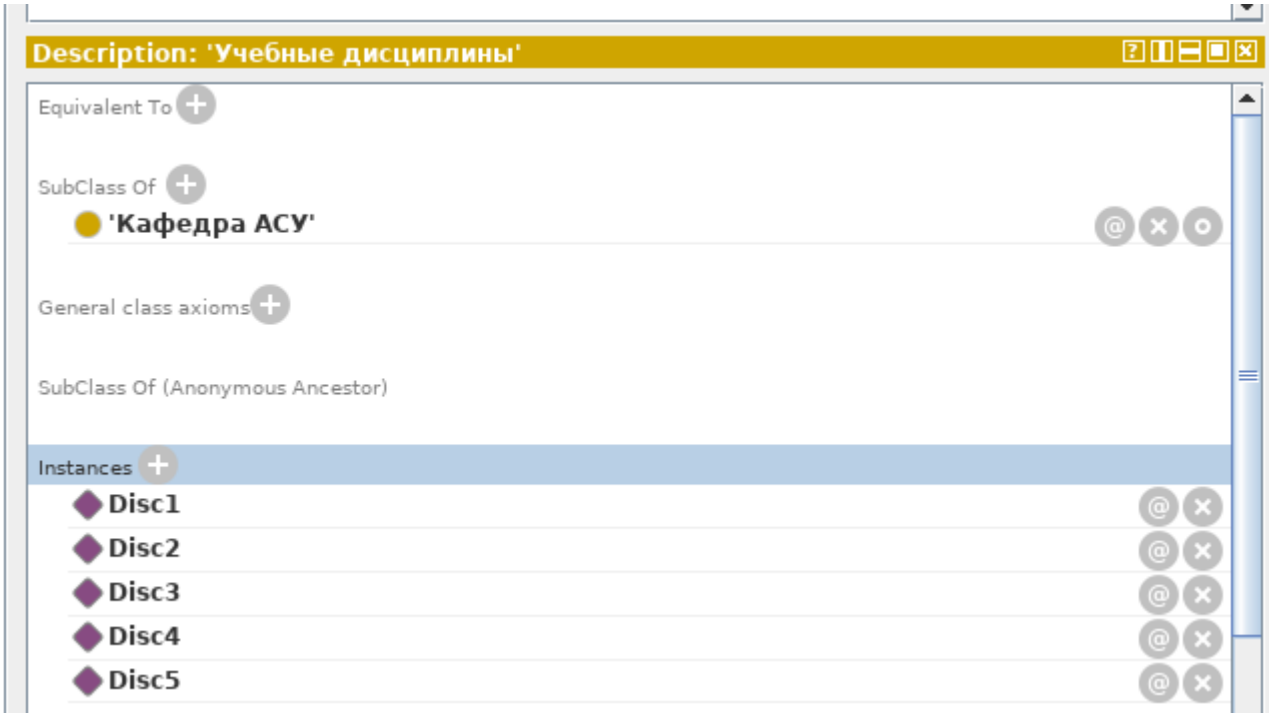

Рисунок 3.18 — Список кодировок объектов класса «Учебные дисциплины»

Теперь перейдем на вкладку «***Entities->Individuals***», как показано на рисунке 3.19.

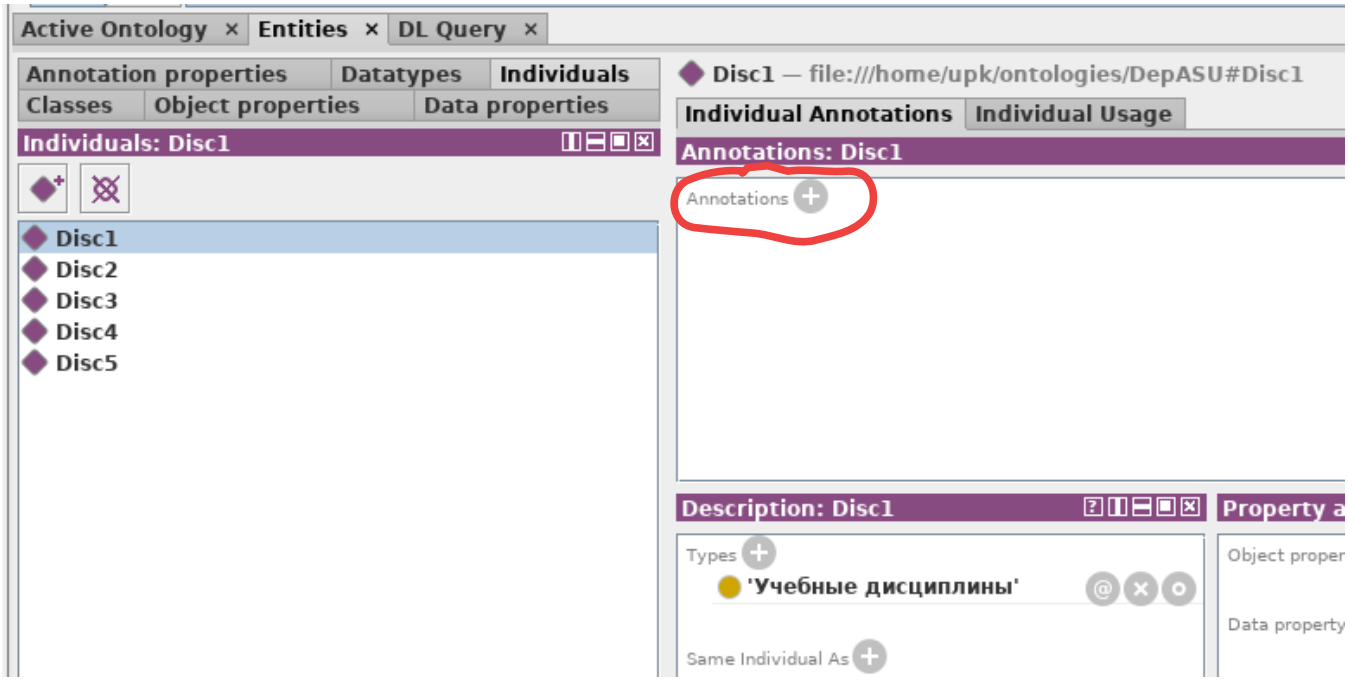

Рисунок 3.19 — Окна настройки свойств объектов Individuals

Применяя инструмент «***Annotations +***», проводим именование дисциплин с помощью ***rdfs:label***, согласно значениям таблицы 3.2. В результате получим представление показанное на рисунке 3.20.

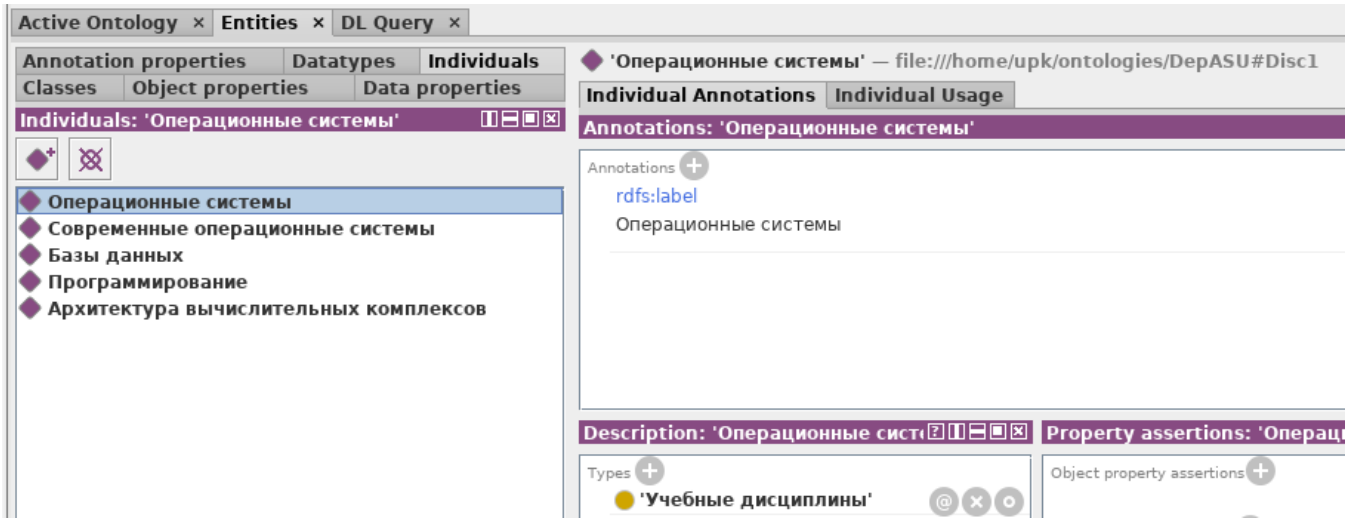

Рисунок 3.20 — Результат полной настройки объектов класса «Учебные дисциплины»

Шаг 5. Создание объектов класса «Сотрудники кафедры».

Добавим в указанный класс пять объектов согласно данным, представленным в таблице 3.3.

Таблица 3.3

Кодировка объекта	***ФИО сотрудника кафедры АСУ***
Emp1	Кориков Анатолий Михайлович
Emp2	Горитов Александр Николаевич
Emp3	Катаев Юрий Михайлович
Emp4	Резник Виталий Григорьевич
Emp5	Сибилев Валерий Дмитриевич

После создания объектов, общий вид системы Protege для класса «Сотрудники кафедры» будет иметь вид показанный на рисунке 3.21.

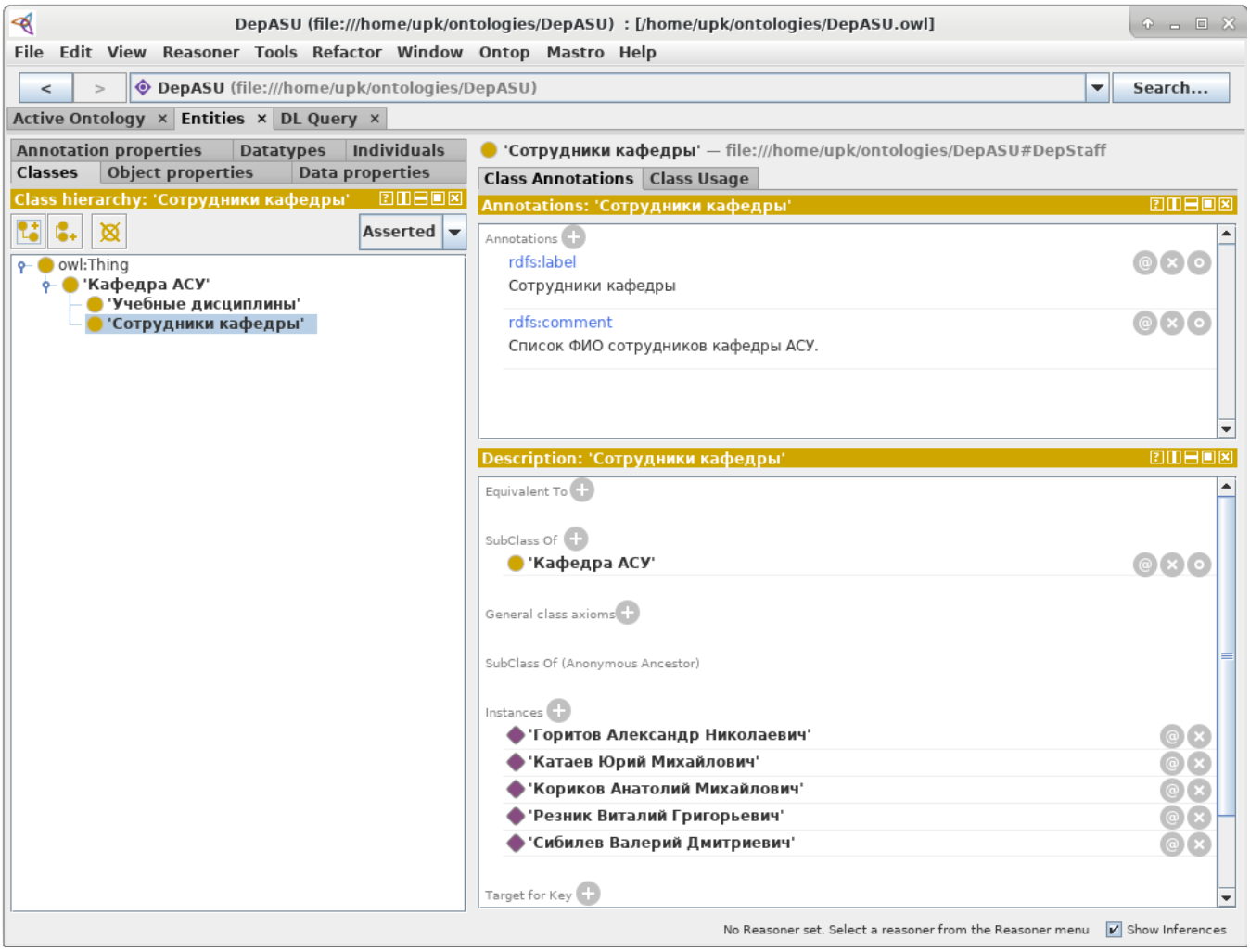

Рисунок 3.21 — Результат полной настройки объектов класса «Сотрудники кафедры»

Шаг 7. Создание объектных отношений.

Структура объектных отношений создается на вкладке «***Entities*** → ***Object properties***», как показано на рисунке 3.22.

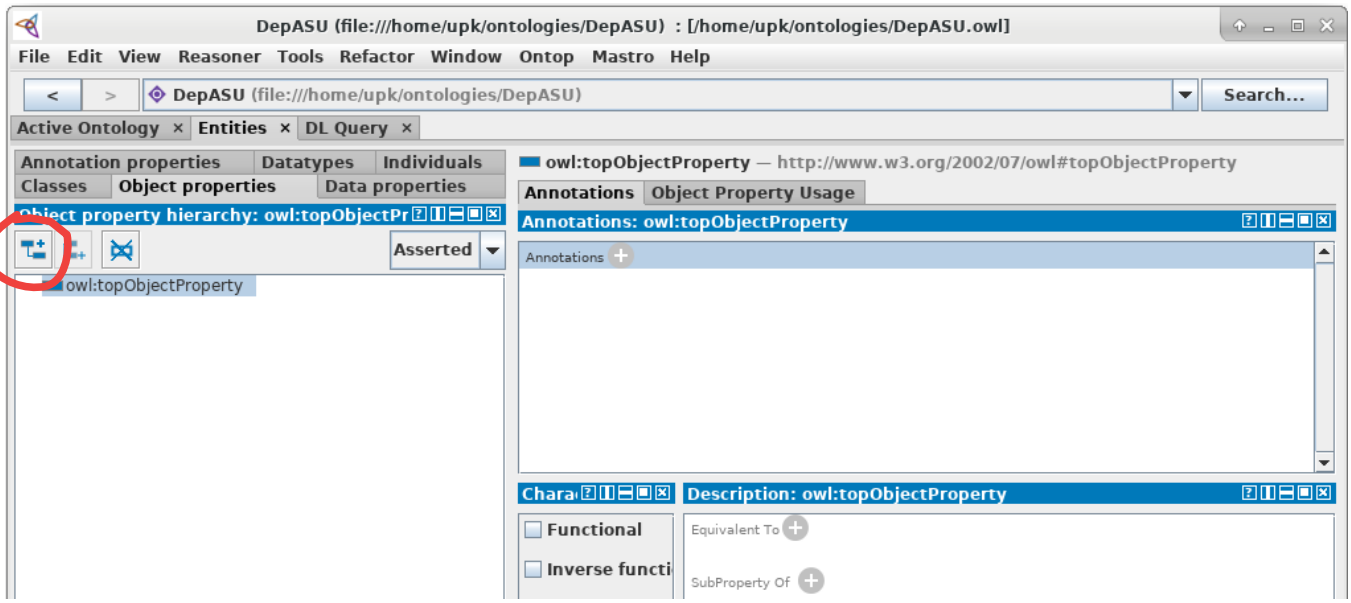

Рисунок 3.22 — Вкладка настройки свойств объектных отношений

Используя кнопку, выделенную на рисунке 3.22, построим иерархию «Object properties» соглано кодировке отношений, приведенной в таблице 3.1.

Результат показан на рисунке 3.23.

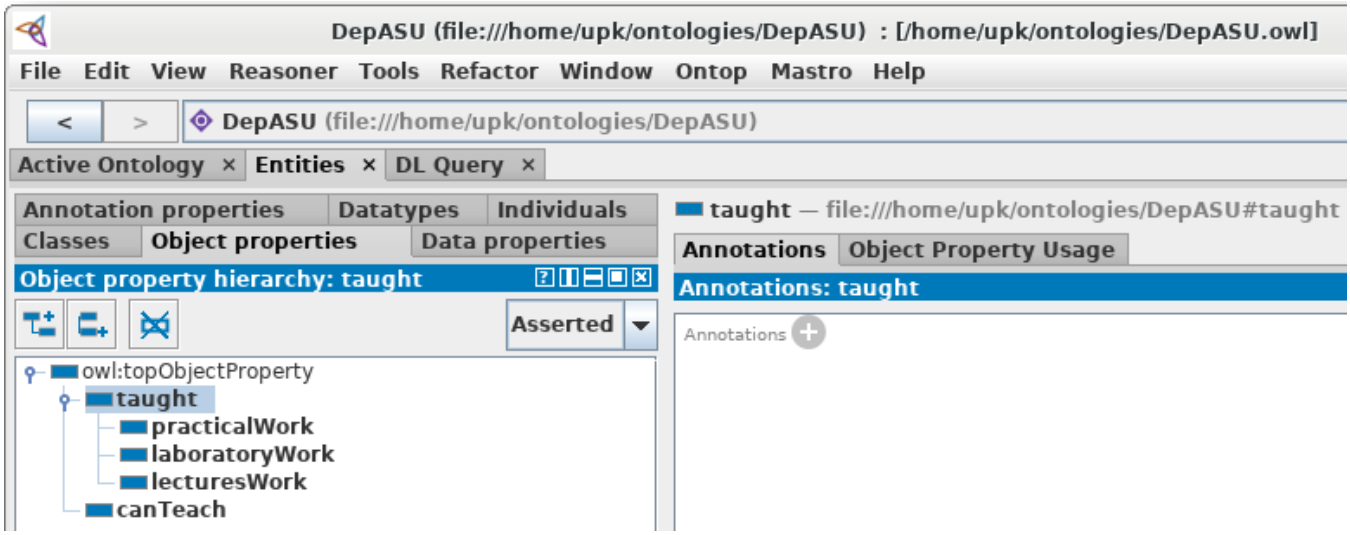

Рисунок 3.23 — Иерархия свойств объектных отношений на уровне их кодировок

Теперь, используя инструмент «***Annotations +***», создадим ***rdfs:label*** для каждого свойства, согласно данным таблицы 3.1.

Результат показан на рисунке 3.24.

Шаг 8. Установка отношений «может преподавать».

Отношение «может преподавать» устанавливается между отдельным объектами классов «Сотрудники кафедры» и «Учебные дисциплины».

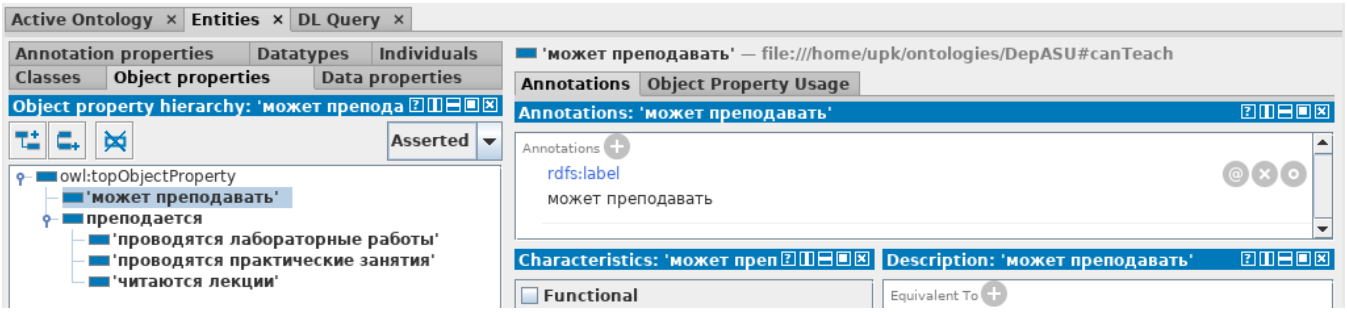

Рисунок 3.24 — Результат создания иерархии свойств объектных отношений

Само отношение индивида класса «Сотрудники кафедры» отражает некоторую «готовность» по отношению к отдельным объектам класса «Учебные дисциплины», которая обусловлена многими нераскрываемыми причинами и может меняться во времени под воздействием дополнительных причин. Поэтому, такие отношения задаются конкретно. Например:

- «Резник Виталий Григорьевич» может преподавать все дисциплины, кроме «Программирование»;
- «Сибилев Валерий Дмитриевич» может преподавать дисциплины «Базы данных» и «Программирование»;
- остальные «сотрудники кафедры» не могут преподавать дисциплины, входящие в класс «Учебные дисциплины».

Прдемонстрируем задание отношения объекту «Сибилев Валерий Дмитриевич» к объекту «Базы данных». Для этого, выделим исходный объект, как показано на рисунке 3.25.

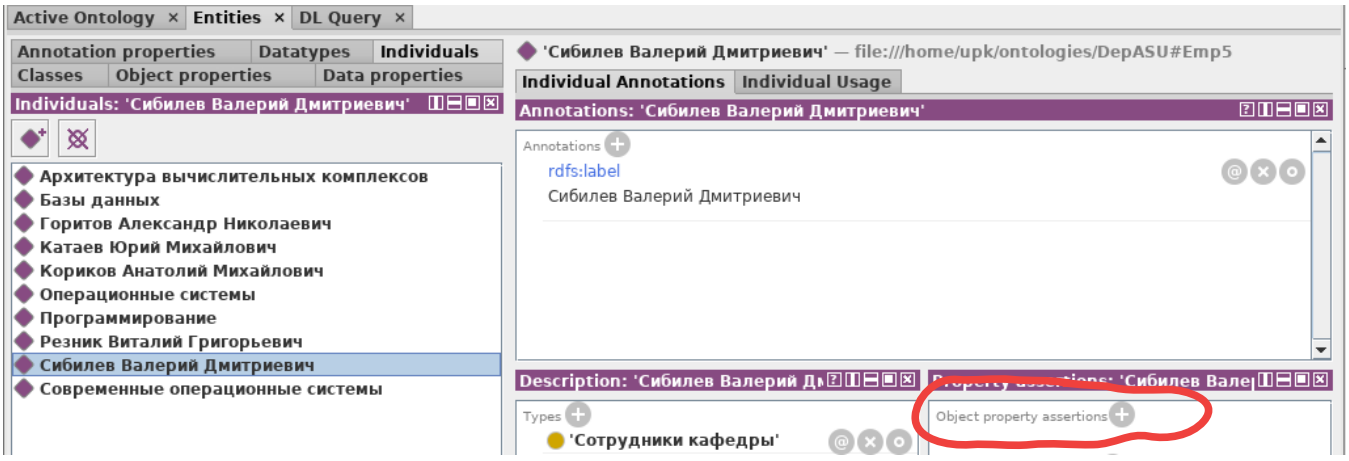

Рисунок 3.25 — Выделение исходного объекта отношения

Затем, воспользуемся кнопкой «***Object property assertions*** +» и заполним значениями диалоговое окно, показанное на рисунке 2.26.

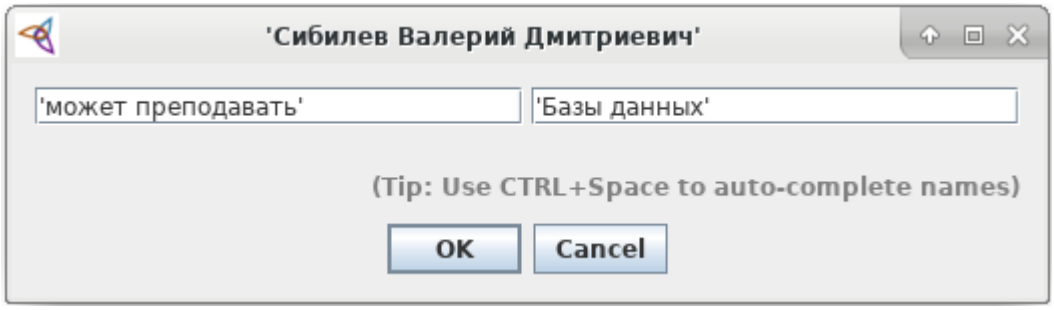

Рисунок 3.26 — Заполнение диалогового окна для «Сибилев Валерий Дмитриевич»

Аналогично связываем объект с дисциплиной «Программирование» и получаем результат, показанный на рисунке 3.27.

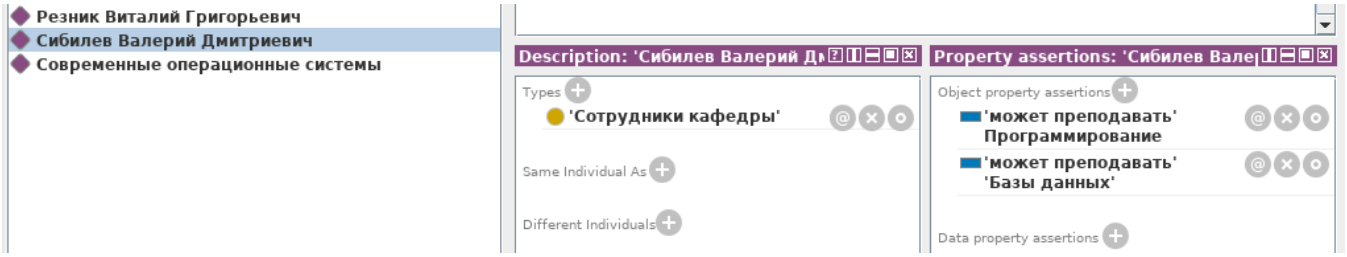

Рисунок 3.27 — Результат отношения «может преподавать» для объекта «Сибилев Валерий Дмитриевич»

Таким же образом создаются отношения «может преподавать» для объекта «Резник Виталий Григорьевич». Результат показан на рисунке 3.28.

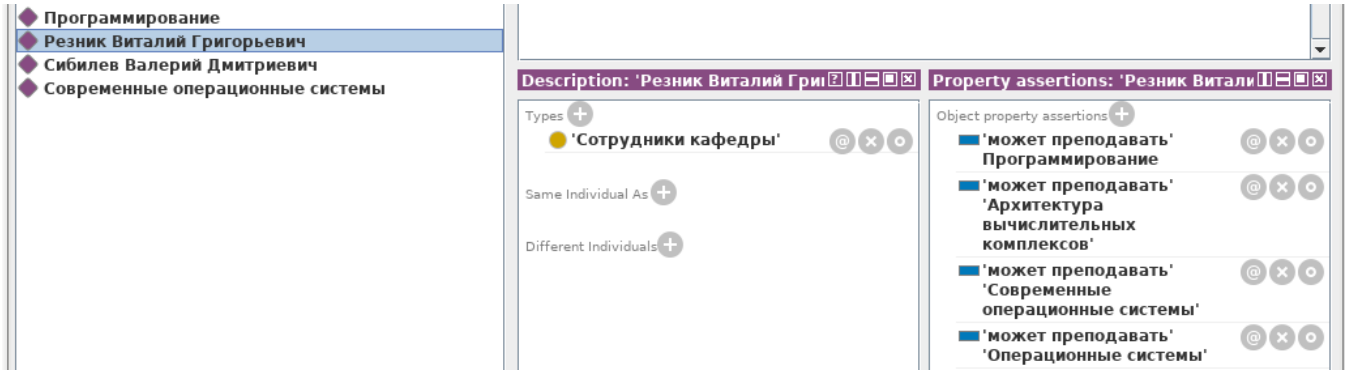

Рисунок 3.28 — Результат отношения «может преподавать» для объекта «Резник Виталий Григорьевич»

Шаг 9. Создание объектных отношений «преподается», «читаются лекции», «проводятся лабораторные работы» и «проводятся практические занятия».

В целом, указанные отношения создаются как и на предыдущем шаге, но имеют следующие особенности:

- отношение «преподается» подразумевает наличие и трех последующих отно-

шений;

- любое из этих отношений имеет с крытый временный характер, не отраженный в условиях постановки задачи;
- любое из этих отношений предполагает наличие отношения «может преподавать».

Учитывая сказанное выше, будем считать, что на данный момент времени только «Резник Виталий Григорьевич» преподает дисциплину «Операционные системы» и читает лекции по дисциплине «Архитектура вычислительных комплексов», а остальные дисциплины еще не распределены между преподавателями.

Реализация указанных отношений демонстрируется на рисунках 3.29 и 3.30.

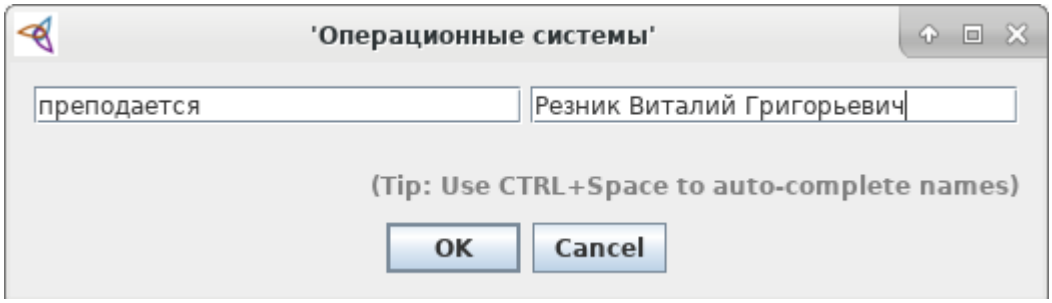

Рисунок 3.29 — Заполнение диалогового окна «Операционные системы»

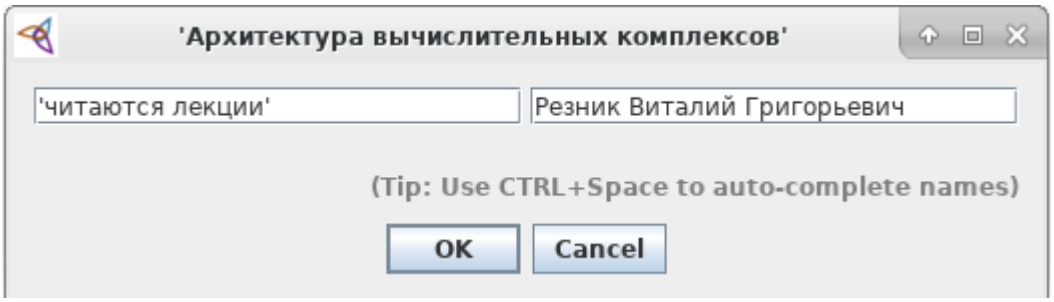

Рисунок 3.30 — Заполнение диалогового окна «Архитектура вычислительных комплексов»

Шаг 10. Отображение онтологии в виде графа.

Система Protege имеет графические средства для отображения созданных онтологий. Чтобы воспользоваться этим средством, достаточно перейти на вкладку «***Active Ontology***».

Если на вкладке «***Active Ontology***» отсутствует окно «***OntoGraf:***», то его следует туда поместить, выбрав из меню системы пункт «***Window*** → ***Views*** → ***Class views*** → ***OntoGraf***».

На рисунке 3.31 показано графическое представление созданной нами онтологии «Кафедра АСУ».

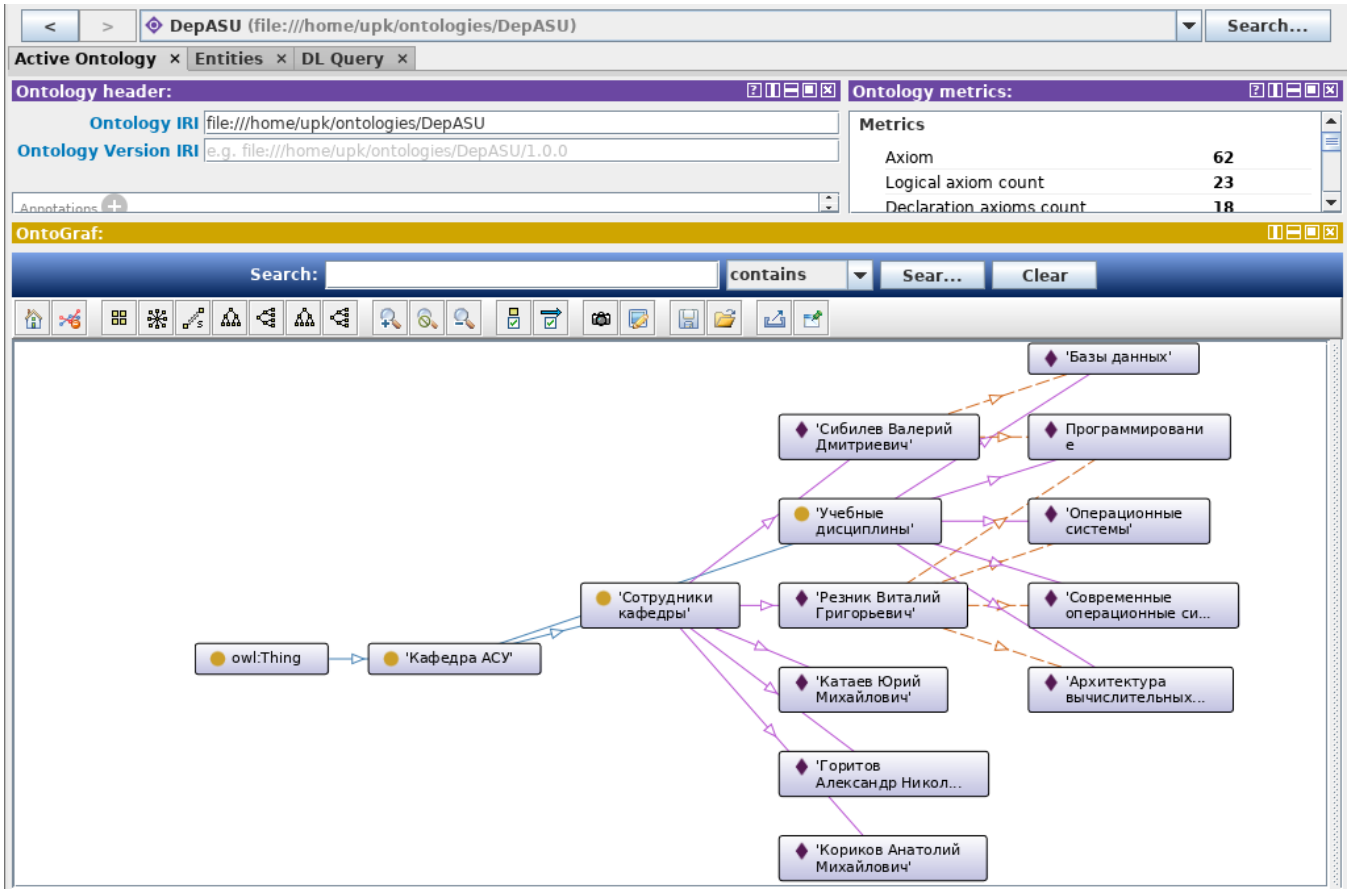

Рисунок 3.31 — Граф построенной онтологии «Кафедра АСУ»

3.2.2.3 Учебные задания

Аспиранту следует выполнить в системе Protege следующий перечень работ:

1. Провести персональное создание онтологии учебного примера и отобразить эту работу в личном отчете.
2. Затем, применительно к объекту высшего онтологического уровня с именем «Кафедра АСУ», создать таблицу с кратким описанием классов: «Сотрудники кафедры», «Учебные дисциплины», «Студенты кафедры», «Аудитории кафедры», «Учебные планы», «Направления обучения», «Должности сотрудников», «Ученые степени сотрудников», «Учебные группы».
3. Выбрать из перечисленного списка один из классов, не совпадающих с уже имеющимися классами учебного проекта, и добавить его в онтологию учебного примера.
4. Результаты самостоятельной разработки описать в личном отчете.
5. Отобразить в личном отчете граф полученной онтологии.

3.2.3 Список использованных источников

3.2.1 Высшая онтология — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/Высшая_онтология). Проверено: 19.12.2018.

3.2.2 Онтология (информатика) — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/Онтология_(иформатика)). Проверено: 19.12.2018.

3.2.3 Web Ontology Language — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/Web_Ontology_Language). Проверено: 19.12.2018.

3.2.4 OWL 2, Язык Веб-онтологий. Общий обзор документа: [Электронный документ]. - (http://zajtcev.org/docs/w3c/ru/REC-owl2-overview-20091027.html). Проверено: 19.12.2018.

3.2.5 OWL, язык веб-онтологий. Краткий обзор: [Электронный документ]. - (http://sherdim.ru/pts/semantic_web/REC-owl-features-20040210_ru.html). Проверено: 19.12.2018.

3.2.6 OWL, язык веб-онтологий. Руководство: [Электронный документ]. - (http://sherdim.ru/pts/semantic_web/REC-owl-guide-20040210_ru.html). Проверено: 19.12.2018.

3.2.7 Горшков С. Введение в онтологическое моделирование. - ООО «ТриниДата», 2014-2016: [Электронный документ]. - (https://trinidata.ru/files/SemanticIntro.pdf). Проверено: 19.12.2018.

3.2.8 URI — Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/URI). Проверено: 19.12.2018.

3.3 Практическая работа №9 «Технология применения семантики языка OWL 2»

Цель работы: освоение технологии применения языка OWL 2.

3.3.1 Самостоятельная работа

Проблемы и перспективы представления знаний. Назначение и принципы построения экспертных систем. Классификация экспертных систем. Методология разработки экспертных систем. Этапы разработки экспертных систем. Проблемы и перспективы построения экспертных систем.

Литература: [12 - 16].

3.3.2 Порядок выполнения работы

На предыдущих двух занятиях были изучены инструментальные средства использования языка OWL. К таким средствам относятся:

- система Protege, обеспечивающая автоматизацию построения онтологий на основе визуализации специализированных редакторов, ориентированных на синтаксис и семантику языка OWL 2;
- синтаксические средства самого языка OWL, опирающиеся на синтаксис моделей RDF и языка XML.

Как результат, были показаны и закреплены на практике навыки построения ряда онтологий на примере предметной области - «Кафедра АСУ».

Данное практическое занятие посвящено двум технологическим аспектам построения предметно-ориентированных онтологий:

- конкретизации базовых средств языка OWL 2, которые поддерживаются системой Protege и другими аналогичным инструментами;
- концептуальным аспектам построения предметно-ориентированных онтологий, демонстрируемых на трех примерах, входящих в уже рассмотренную ранее предметную область - «Кафедра АСУ».

3.3.2.1 Конкретизация базовых средств языка OWL 2

Формально, синтаксис языка OWL 2 описан в рекомендации W3C [3.3.1], важной особенностью которой является тот факт, что каждый элемент триплета «Субъект-Предикат-Объект» идентифицируется не URI, как в спецификации OWL, а абсолютным указателем IRI.

Согласно [3.3.2]: «**IRI** (англ. ***Internationalized Resource Identifier***) — интер-

национализированный идентификатор ресурса. ... IRI — это короткая последовательность символов, идентифицирующая абстрактный или физический ресурс на любом языке мира. Идентификаторы IRI призваны в будущем заменитьURI. …

IRI — это символьная строка, позволяющая идентифицировать какой-либо ресурс: документ, изображение, файл, службу, ящик электронной почны и т. д. Прежде всего, речь идёт, конечно, о ресурсах сети Интернет и Всемирной паутины. Идентификаторы IRI создавались как замена единообразным идентификаторам **URI** (англ. ***Uniform Resource Identifier***) с целью избежать их ограничения на символы: URI могут содержать только латинские символы и знаки препинания из набора символов US-ASCII (в общей сложности около 60 символов). В результате, если мы захотим использовать в URI символы кирилицы, иероглифы или, скажем, специфические символы французкого языка и эсперанто, то нам придётся кодировать URI ... символами Юникода. Например, строка вида:

```
http://ru.wikipedia.org/wiki/Кириллица
```

кодируется в URL как:

```
http://ru.wikipedia.org/wiki/%D0%9A%D0%B8%D1%80%D0%B8%D0%BB%D0%BB%D0%B8%D1%86%D0%B0
```

...».

Базовая часть семантики языка OWL 2 отражается синтаксическими конструкциями, показанными на рисунке 3.32 и содержащими встроенную семантику языка.

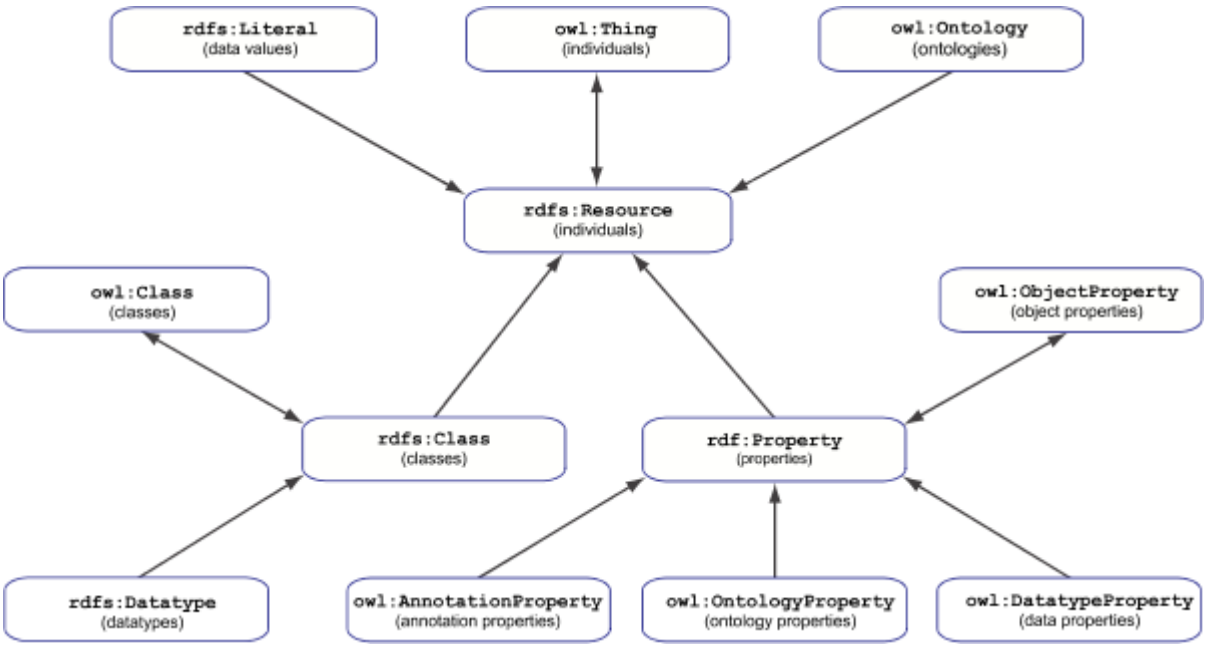

Рисунок 3.32 — Часть иерархии семантики языка OWL 2 (заимствовано из источника [3.3.3])

Вся указанная на рисунке 3.32 базовая встроенная семантика языка OWL 2 полностью поддерживается системой Protege и может быть использована для построения предметно-ориентированных онтологий. С другой стороны, только наличие этих средств не обеспечивает адекватность и однозначность отображения предметной области. Покажем это конкретными примерами.

3.3.2.2 Онтология «Список дисциплин»

Чтобы показать неоднозначность отображения с помощью онтологии ее предметной области, проведем анализ класса «Учебные дисциплины», рассмотренный нами на предыдущем занятии. Этот пример оформим в виде отдельной онтологии «Список дисциплин» и будем использовать в других примерах данного практического занятия.

Прежде всего заметим, что этот список дисциплин разработан в рамках онтологии DepASU («Кафедра АСУ») и представлен таблицей 3.2, содержащей кодировки объектов DiscN и рускоязычные названия дисциплин, где N — задает модальность их различия. Если снять ограничения URI, накладываемые языком OWL, и заменить их средствами русскоязычной кодировки IRI, предоставляемые языком OWL 2, то мы получим эквивалентное описание одной и той же онтологии.

С другой стороны, возникают вопросы: «Насколько адекватно и однозначно имена, используемые в таблице 3.2, отображают объекты подразумеваемых дисциплин?»:

1) является ли отдельная дисциплина объектом (owl:NamedIndividual) или классом (owl:Class), включающим совокупность знаний различных научных направлений?
2) являются ли эквивалентными дисциплины, имеющие одинаковое имя, но преподаваемые студентам, различных кафедр и факультетов вуза?
3) являются ли эквивалентными дисциплины, имеющие одинаковое имя, но преподаваемые студентам различных направлений обучения?

Анализируя представленные вопросы, следует отметить наличие возможных альтернативных толкований понятия дисциплины по ее заявленному имени. Например, можно рассматривать онтологию дисциплины «Операционные системы» и, тем более, сравнивать ее с онтологией дисциплины «Современные операционные системы», но в рамках онтологии отдельной кафедры («Кафедры АСУ») каждая отдельная дисциплина интерпретируется как отдельный объект, имя которого фигурирует:

- в отдельных документах, регулирующих учебный процесс кафедры;
- в отношениях (предикатах, owl:ObjectProperty) объектов классов «Сотрудники кафедры», «Учебные группы», «Учебные планы» и других.

Таким образом, создаваемая отнология предметной области «Список дисциплин» является хоть и примитивной, но адекватной и востребованной в более общей онтологии «Кафедра АСУ».

С целью реализации онтологии «Список дисциплин» воспользуемся инструментом системы Protege:

- создадим класс ***СУД*** (Список учебных дисциплин);
- создадим объекты класса, согласно данным таблицы 3.4;

- сохраним онтологию в файле: ***/home/upk/ontologies/disciplines.owl***.

Таблица 3.4

Кодировка объекта	***Наименование дисциплины***
ОС	Операционные системы
СОС	Современные операционные системы
БД	Базы данных
ПРОГ	Программирование
АВК	Архитектура вычислительных комплексов

Результат разработки онтологии СУД представлен на рисунке 3.33.

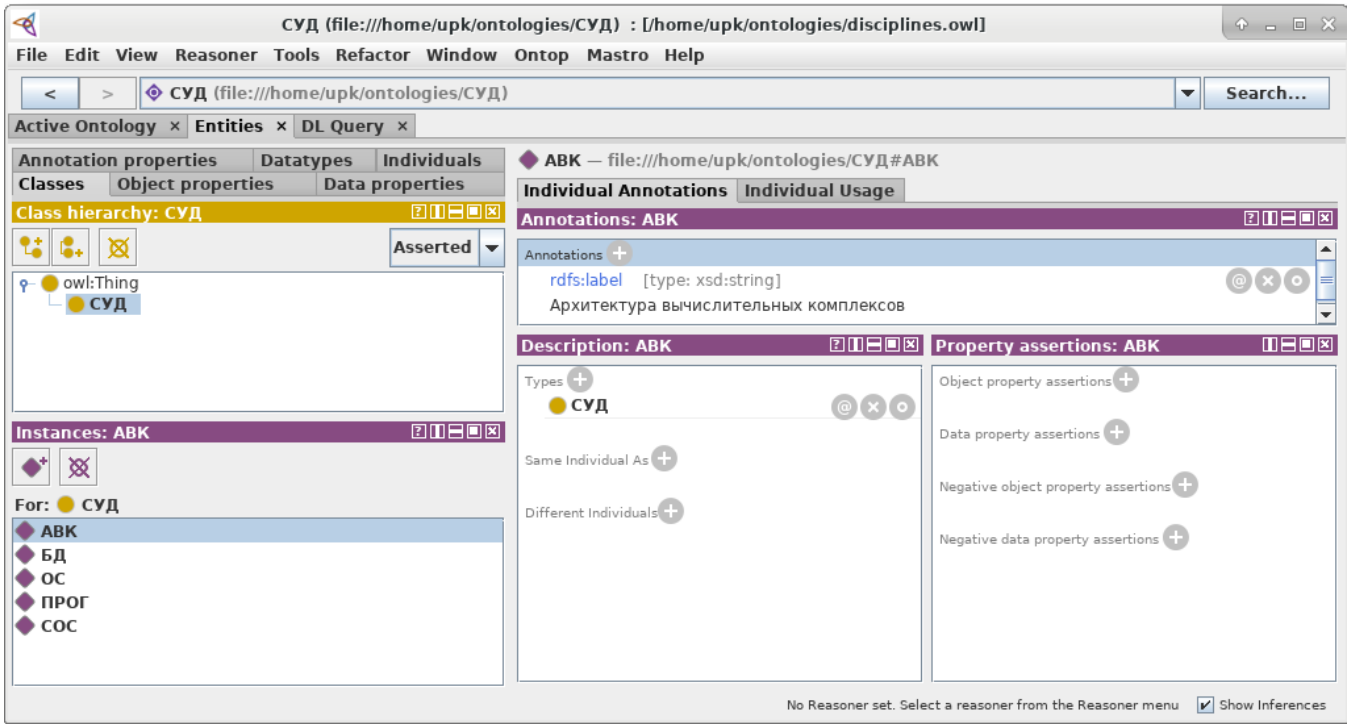

Рисунок 3.33 — Онтология СУД в системе Protege

На листинге 3.3 представлено содержимое файла ***disciplines.owl***.

Листинг 3.3 — Содержимое файла disciplines.owl

```
<?xml version="1.0"?>
<rdf:RDF xmlns="file:///home/upk/ontologies/СУД#"
     xml:base="file:///home/upk/ontologies/СУД"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xml="http://www.w3.org/XML/1998/namespace"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
    <owl:Ontology rdf:about="file:///home/upk/ontologies/СУД"/>
```

```
    <!--
    ///////////////////////////////////////////////////////////////////////////////
    //
    // Classes
    //
    ///////////////////////////////////////////////////////////////////////////////
     -->

    <!-- file:///home/upk/ontologies/СУД#СУД -->

    <owl:Class rdf:about="file:///home/upk/ontologies/СУД#СУД">
        <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Учебные
дисциплины</rdfs:label>
    </owl:Class>


    <!--
    ///////////////////////////////////////////////////////////////////////////////
    //
    // Individuals
    //
    ///////////////////////////////////////////////////////////////////////////////
     -->

    <!-- file:///home/upk/ontologies/СУД#АВК -->

    <owl:NamedIndividual rdf:about="file:///home/upk/ontologies/СУД#АВК">
        <rdf:type rdf:resource="file:///home/upk/ontologies/СУД#СУД"/>
        <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Архитектура вычислительных
комплексов</rdfs:label>
    </owl:NamedIndividual>


    <!-- file:///home/upk/ontologies/СУД#БД -->

    <owl:NamedIndividual rdf:about="file:///home/upk/ontologies/СУД#БД">
        <rdf:type rdf:resource="file:///home/upk/ontologies/СУД#СУД"/>
        <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Базы
данных</rdfs:label>
    </owl:NamedIndividual>


    <!-- file:///home/upk/ontologies/СУД#ОС -->

    <owl:NamedIndividual rdf:about="file:///home/upk/ontologies/СУД#ОС">
        <rdf:type rdf:resource="file:///home/upk/ontologies/СУД#СУД"/>
        <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Операционные
системы</rdfs:label>
    </owl:NamedIndividual>
```

```
    <!-- file:///home/upk/ontologies/СУД#ПРОГ -->

    <owl:NamedIndividual rdf:about="file:///home/upk/ontologies/СУД#ПРОГ">
        <rdf:type rdf:resource="file:///home/upk/ontologies/СУД#СУД"/>
        <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Программирование</
rdfs:label>
    </owl:NamedIndividual>


    <!-- file:///home/upk/ontologies/СУД#СОС -->

    <owl:NamedIndividual rdf:about="file:///home/upk/ontologies/СУД#СОС">
        <rdf:type rdf:resource="file:///home/upk/ontologies/СУД#СУД"/>
        <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Современные операционные
системы</rdfs:label>
    </owl:NamedIndividual>
</rdf:RDF>

<!-- Generated by the OWL API (version 4.2.8.20170104-2310)
https://github.com/owlcs/owlapi -->
```

Конец листинга 3.3

3.3.2.3 Онтология «Сотрудники кафедры»

Онтология «Сотрудники кафедры» также была рассмотрена на предыдущем занятии и кажется, что с ней — все нормально:

- сами сотрудники являются четко выделенными объектами (owl:NamedIndividual) и принадлежат конкретному классу (owl:Class);
- разумным является также объектное отношение (owl:OblectProperty) с именем «может преподавать», связывающее субъект «Сотрудника кафедры» с объектом «Сотрудники кафедры».

Более глубокий анализ показывает следующие недостатки такого онтологического описания:

- предикат «может преподавать» - действительно имеет место, но он больше подходит для учебного примера, поскольку перекрывается предикатами, входящими в руководящий документ, например, «Ведомость учебных поручений кафедре»;
- каждый сотрудник кафедры имеет некоторую должность, которую следует рассматривать как объект, например, класса «Штатное расписание кафедры» или «Список должностей кафедры»; причем, окончательное регулирование должностями осуществляется не на уровне отдельной кафедры или факультета, а на уровне вуза;
- каждый сотрудник кафедры имеет (или не имеет) ученую степень, которая

является объектом класса «Список ученых степеней»; причем этот список формируется и утверждается за пределами вуза;

- аналогично, каждый сотрудник кафедры может иметь ученое звание и т. д.

Тамим образом, проведенный анализ показывает, что онтология «Список сотрудников» имеет свои особенности, наиболее важные из которых следует учесть при ее построении и последующем использовании. Другими словами, предметно-ориентированные онтологии, как и структуры баз данных, всегда создаются в условиях ряда ограничений, определяемых ее прикладным назначением. Дополнительно, на указанные ограничения существенное влияние оказывает размер исследуемой предметной области: чем больше размер области, - тем больше в ней можно выделить классов и отношений.

Учитывая слеланные замечания, создадим онтологию с именем «Список сотрудников», со следующей структурой:

- иерархия классов представлена названиями, приведенными в таблице 3.5:
- индивидуальные объекты сотрудников взяты из таблицы 3.3 и представлены в формате таблицы 3.6;
- онтология сохраняется в файле ***employees.owl***.

Таблица 3.5

Имя класса	***Структура класса: объединение подклассов***
СписокСотрудников	`Должности, УченыеСтепени`
Должности	`Профессора, Доценты, Ассистенты, Ст.Преподаватели`
УченыеСтепени	`д.т.н, д.ф.м.н, к.т.н, к.ф.м.н`

Таблица 3.6

Кодировка объекта	***ФИО сотрудника кафедры АСУ***
А.М.Кориков	`Кориков Анатолий Михайлович`
А.Н.Горитов	`Горитов Александр Николаевич`
Ю.М.Катаев	`Катаев Юрий Михайлович`
В.Г.Резник	`Резник Виталий Григорьевич`
В.Д.Сибилев	`Сибилев Валерий Дмитриевич`

Воспользовавшись инструментальным средством Protege, создаем нужную нам онтологию, как показано на рисунке 3.34.

Соответственно, на листинге 3.4 онтология «СписокСотрудников» представлен как содержимое файла ***employees.owl***.

Обратите внимание, что индивидуалные объекты сотрудников принадлежат нескольки различным классам.

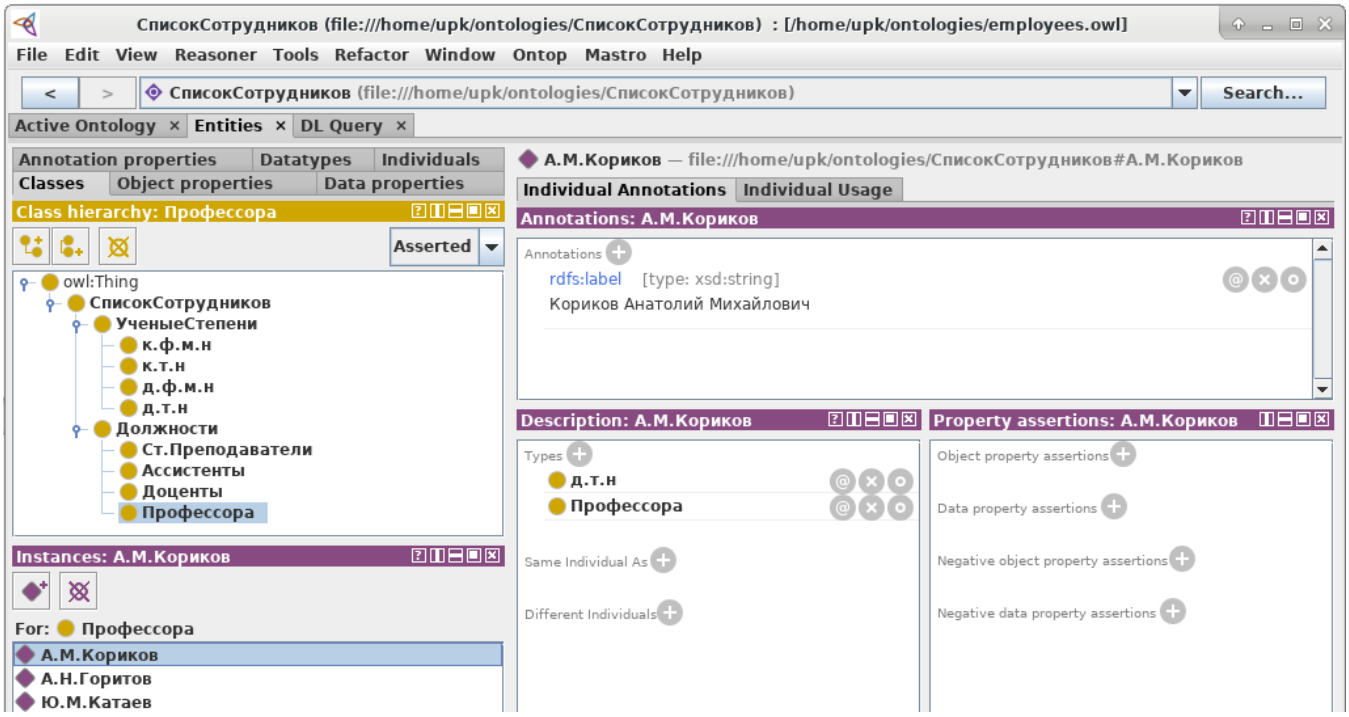

Рисунок 3.34 — Онтология СписокСотрудников в системе Protege

Листинг 3.4 — Содержимое файла employees.owl

```
<?xml version="1.0"?>
<rdf:RDF xmlns="file:///home/upk/ontologies/СписокСотрудников#"
     xml:base="file:///home/upk/ontologies/СписокСотрудников"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xml="http://www.w3.org/XML/1998/namespace"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
    <owl:Ontology rdf:about="file:///home/upk/ontologies/СписокСотрудников"/>


    <!--
    ///////////////////////////////////////////////////////////////////////////////////
    //
    // Classes
    //
    ///////////////////////////////////////////////////////////////////////////////////
     -->

    <!-- file:///home/upk/ontologies/СписокСотрудников#Ассистенты -->

    <owl:Class
rdf:about="file:///home/upk/ontologies/СписокСотрудников#Ассистенты">
        <rdfs:subClassOf
rdf:resource="file:///home/upk/ontologies/СписокСотрудников#Должности"/>
    </owl:Class>
```

```
    <!-- file:///home/upk/ontologies/СписокСотрудников#Должности -->

    <owl:Class rdf:about="file:///home/upk/ontologies/СписокСотрудников#Должности">
        <rdfs:subClassOf
rdf:resource="file:///home/upk/ontologies/СписокСотрудников#СписокСотрудников"/>
    </owl:Class>


    <!-- file:///home/upk/ontologies/СписокСотрудников#Доценты -->

    <owl:Class rdf:about="file:///home/upk/ontologies/СписокСотрудников#Доценты">
        <rdfs:subClassOf
rdf:resource="file:///home/upk/ontologies/СписокСотрудников#Должности"/>
    </owl:Class>


    <!-- file:///home/upk/ontologies/СписокСотрудников#Профессора -->

    <owl:Class
rdf:about="file:///home/upk/ontologies/СписокСотрудников#Профессора">
        <rdfs:subClassOf
rdf:resource="file:///home/upk/ontologies/СписокСотрудников#Должности"/>
    </owl:Class>


    <!-- file:///home/upk/ontologies/СписокСотрудников#СписокСотрудников -->

    <owl:Class
rdf:about="file:///home/upk/ontologies/СписокСотрудников#СписокСотрудников"/>


    <!-- file:///home/upk/ontologies/СписокСотрудников#Ст.Преподаватели -->

    <owl:Class
rdf:about="file:///home/upk/ontologies/СписокСотрудников#Ст.Преподаватели">
        <rdfs:subClassOf
rdf:resource="file:///home/upk/ontologies/СписокСотрудников#Должности"/>
    </owl:Class>


    <!-- file:///home/upk/ontologies/СписокСотрудников#УченыеСтепени -->

    <owl:Class
rdf:about="file:///home/upk/ontologies/СписокСотрудников#УченыеСтепени">
        <rdfs:subClassOf
rdf:resource="file:///home/upk/ontologies/СписокСотрудников#СписокСотрудников"/>
    </owl:Class>


    <!-- file:///home/upk/ontologies/СписокСотрудников#д.т.н -->

    <owl:Class rdf:about="file:///home/upk/ontologies/СписокСотрудников#д.т.н">
        <rdfs:subClassOf
rdf:resource="file:///home/upk/ontologies/СписокСотрудников#УченыеСтепени"/>
    </owl:Class>
```

```
    <!-- file:///home/upk/ontologies/СписокСотрудников#д.ф.м.н -->

    <owl:Class rdf:about="file:///home/upk/ontologies/СписокСотрудников#д.ф.м.н">
        <rdfs:subClassOf
rdf:resource="file:///home/upk/ontologies/СписокСотрудников#УченыеСтепени"/>
    </owl:Class>


    <!-- file:///home/upk/ontologies/СписокСотрудников#к.т.н -->

    <owl:Class rdf:about="file:///home/upk/ontologies/СписокСотрудников#к.т.н">
        <rdfs:subClassOf
rdf:resource="file:///home/upk/ontologies/СписокСотрудников#УченыеСтепени"/>
    </owl:Class>


    <!-- file:///home/upk/ontologies/СписокСотрудников#к.ф.м.н -->

    <owl:Class rdf:about="file:///home/upk/ontologies/СписокСотрудников#к.ф.м.н">
        <rdfs:subClassOf
rdf:resource="file:///home/upk/ontologies/СписокСотрудников#УченыеСтепени"/>
    </owl:Class>


    <!--
    //////////////////////////////////////////////////////////////////////////////////
    //
    // Individuals
    //
    //////////////////////////////////////////////////////////////////////////////////
     -->


    <!-- file:///home/upk/ontologies/СписокСотрудников#А.М.Кориков -->

    <owl:NamedIndividual
rdf:about="file:///home/upk/ontologies/СписокСотрудников#А.М.Кориков">
        <rdf:type
rdf:resource="file:///home/upk/ontologies/СписокСотрудников#Профессора"/>
        <rdf:type
rdf:resource="file:///home/upk/ontologies/СписокСотрудников#д.т.н"/>
        <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Кориков
Анатолий Михайлович</rdfs:label>
    </owl:NamedIndividual>


    <!-- file:///home/upk/ontologies/СписокСотрудников#А.Н.Горитов -->

    <owl:NamedIndividual
rdf:about="file:///home/upk/ontologies/СписокСотрудников#А.Н.Горитов">
        <rdf:type
rdf:resource="file:///home/upk/ontologies/СписокСотрудников#Профессора"/>
        <rdf:type
rdf:resource="file:///home/upk/ontologies/СписокСотрудников#д.т.н"/>
        <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Горитов
```

```
Александр Николаевич</rdfs:label>
    </owl:NamedIndividual>


    <!-- file:///home/upk/ontologies/СписокСотрудников#В.Г.Резник -->

    <owl:NamedIndividual
rdf:about="file:///home/upk/ontologies/СписокСотрудников#В.Г.Резник">
        <rdf:type
rdf:resource="file:///home/upk/ontologies/СписокСотрудников#Доценты"/>
        <rdf:type
rdf:resource="file:///home/upk/ontologies/СписокСотрудников#к.т.н"/>
        <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Резник
Виталий Григорьевич</rdfs:label>
    </owl:NamedIndividual>


    <!-- file:///home/upk/ontologies/СписокСотрудников#В.Д.Сибилев -->

    <owl:NamedIndividual
rdf:about="file:///home/upk/ontologies/СписокСотрудников#В.Д.Сибилев">
        <rdf:type
rdf:resource="file:///home/upk/ontologies/СписокСотрудников#Доценты"/>
        <rdf:type
rdf:resource="file:///home/upk/ontologies/СписокСотрудников#к.т.н"/>
        <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Сибилев
Валерий Дмитриевич</rdfs:label>
    </owl:NamedIndividual>


    <!-- file:///home/upk/ontologies/СписокСотрудников#Ю.М.Катаев -->

    <owl:NamedIndividual
rdf:about="file:///home/upk/ontologies/СписокСотрудников#Ю.М.Катаев">
        <rdf:type
rdf:resource="file:///home/upk/ontologies/СписокСотрудников#Профессора"/>
        <rdf:type
rdf:resource="file:///home/upk/ontologies/СписокСотрудников#д.т.н"/>
        <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Катаев
Юрий Михайлович</rdfs:label>
    </owl:NamedIndividual>
</rdf:RDF>


<!-- Generated by the OWL API (version 4.2.8.20170104-2310)
https://github.com/owlcs/owlapi -->
```

Конец листинга 3.4

Замечание

Обратите внимание, что онтологии «СУД» и «СписокСотрудников» не зависят друг от друга и не включают в себя других онтологий предметной области «Кафедра АСУ», поэтому они могут развиваться в плане расширения достаточно автономно.

3.3.2.4 Онтология «Учебные планы»

В отличие от предыдущих двух примеров, онтология «Учебные планы» не была рассмотрена ранее, а входила в список заданий предыдущего практического занятия. Чтобы составить общее представление об этой онтологии, приведем ряд ее обобщенных характеристик:

- «Учебные планы» - это набор (коллекция) отдельных учебных планов, каждый из которых создается по определенному направлению обучения, охватывает определенный период времени (2 или 4 года), утверждается на конкретный календаный год, обычно распространяется на последующие годы и, со временем, заменяется новым учебным планом;
- каждый учебный план состоит из последовательности временных периодов, называемых семестрами (4 или 8 семестров);
- для каждого семестра учебного плана указывается перечень изучаемых дисциплин;
- каждая дисциплина назделена на типы учебных занятий из списка: лекции, лабораторные работы, практические занятия, самостоятельная работа, консультации, зачеты, экзамены и другие;
- для каждого типа учебных занятий указывается их объем: период обучения, заданный в единицах академического часа (45 минут астрономического времени).

Очевидно, перечисленные характеристики онтологии указывают на сложную систему, имеющую множество различных отношений, для которых еще недостаточно конкретной информации. Тем не менее, уже их этих характеристик можно сделать вполне конструктивные выводы:

- онтология «Учебные планы» включает в себя онтологию «Список дисциплин», рассмотренную в предыдущем примере;
- онтология «Учебные планы» состоит из объектов (owl:NamedIndividual), которые можно назвать «Единицы обучения», имеющие отношения с объектами онтологии «Список дисциплин» и объектами онтологии «Тип обучения», а также — характеристику «Объем обучения», заданную в единицах академического часа;
- элементы, объединенные названием «Тип обучения», необходимо выделить в отдельную онтологию с последующим импортом в рассматриваемую, поскольку они явно будут присутствовать в других, еще не рассмотренных нами, онтологиях.

Таким образом, план нашего занятия будет состоять из двух этапов:

- построение онтологии «Тип обучения»;
- построение онтологии «Учебные планы» с реализацией части отдельного учебного плана, например, «Учебный план для направления 09.04.01 на 2018 год».

Этап 1. В системе Protege реализуем простейшую онтологию, включающую следующие состаляющие:

- класс ***TunОбучения***;
- объекты класса — согласно данным, представленным в таблице 3.7;
- онтологию сохраним в файле ***type-of-training.owl***.

Таблица 3.7

Кодировка объекта	***Полное название типа обучения***
лекция	`лекция`
лаб.раб	`лабораторная работа`
прак.зан	`практическое занятие`
сам.раб	`самостоятельная работа`
консультация	`консультация`
зачет	`зачет`
экзамен	`экзамен`
курс.пр	`курсовой проект`
семинар	`семинар`

Результат создания онтологии ***TunОбучения*** представлен на рисунке 3.35.

Этап 2. В системе Protege создадим онтологию ***УчебныеПланы***, которую сохраним в файле ***educational-plans.owl***.

Дальнейшую реализацию онтологии проведем в виде отдельных шагов.

Шаг 1. Создадим вложенные классы, согласно отношениям представленным в таблице 3.8. Результат создания на этом шаге онтологии ***УчебныеПланы*** представлен на рисунке 3.36.

Таблица 3.8

Имя класса	***Структура класса: объединение подклассов***
УчебныеПланы	`УП-09.04.01-2018`
`УП-09.04.01-2018`	`СУП1-09.04.01-2018, СУП2-09.04.01-2018, СУП3-09.04.01-2018, СУП4-09.04.01-2018`

Замечание

УП-09.04.01-2018 — подразумевает учебный план по направлению 09.04.01 на 2018 и последующие годы обучения. Указанную расшифровку следует указать для этого класса в метке типа owl:label.

СУПN-09.04.01-2018 — N-й семестр учебного плана `УП-09.04.01-2018`.

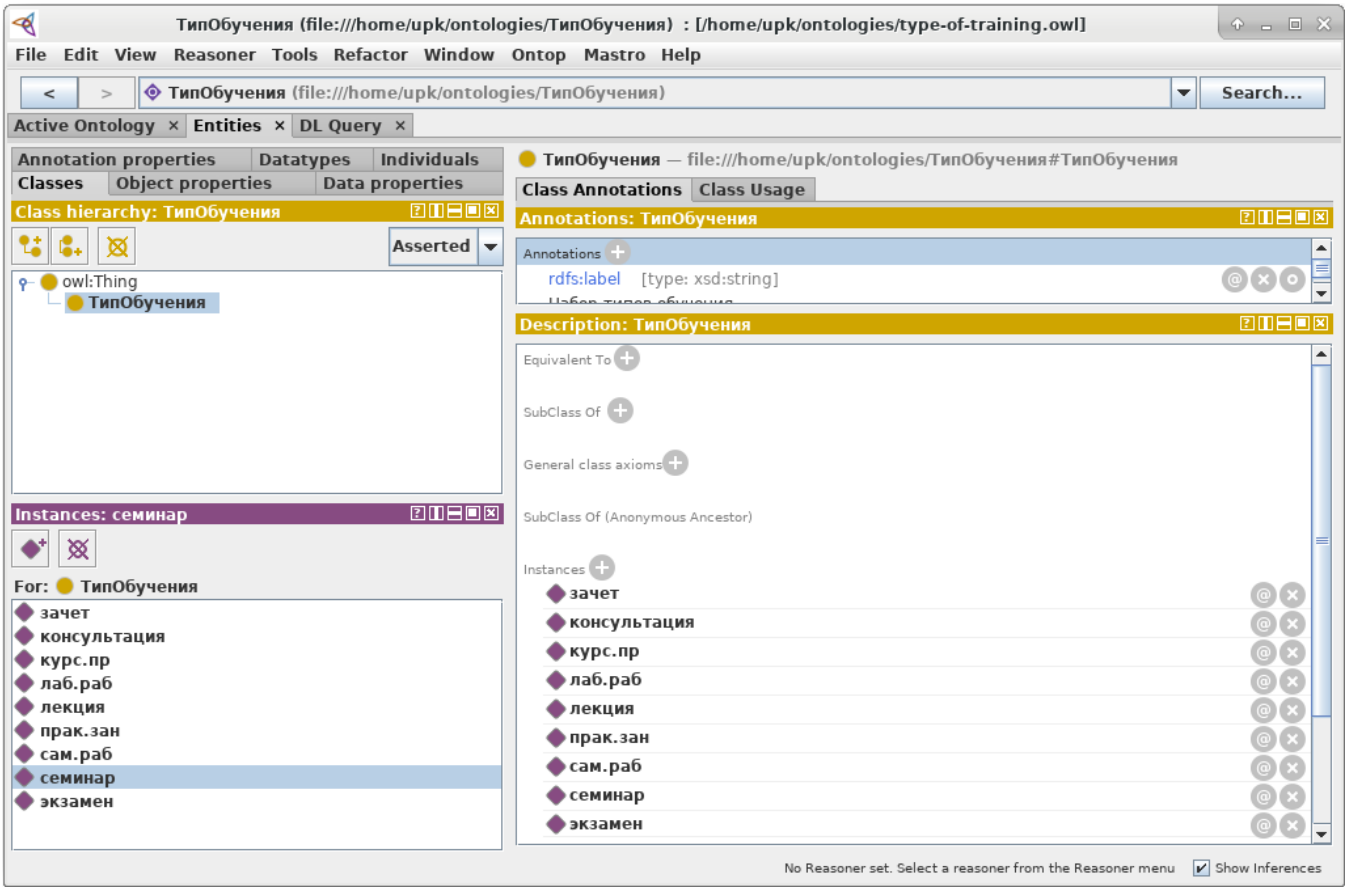

Рисунок 3.35 — Онтология ТипОбучения в системе Protege

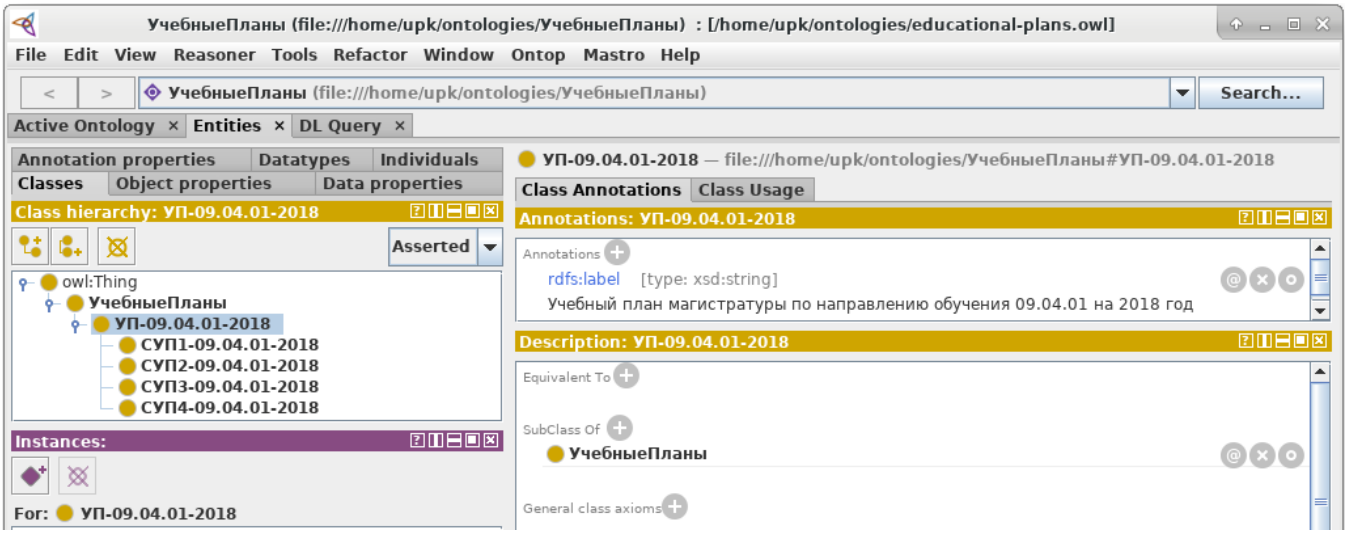

Рисунок 3.36 — Онтология УчебныеПланы в системе Protege на шаге 1

Шаг 2. Импортируем в создаваемую онтологию две другие: ***СУД*** и ***СписокСотрудников***. Для этого необходимо в системе Protege:

- перейти на вкладку «***Ontology header***»;
- активировать диалог «***Direct Imports +***», как показано на рисунке 3.37;
- выбрать файл , как показано на рисунке 3.38, нажать кнопку «***Continue***» и, на следующем окне, нажать кнопку «***Finish***».

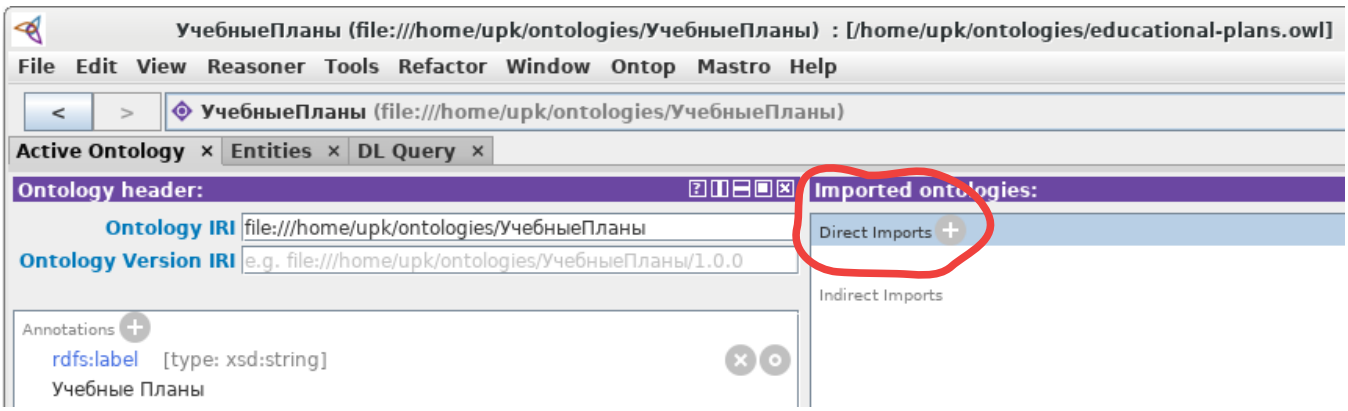

Рисунок 3.37 — Активация диалога «Direct Imports +»

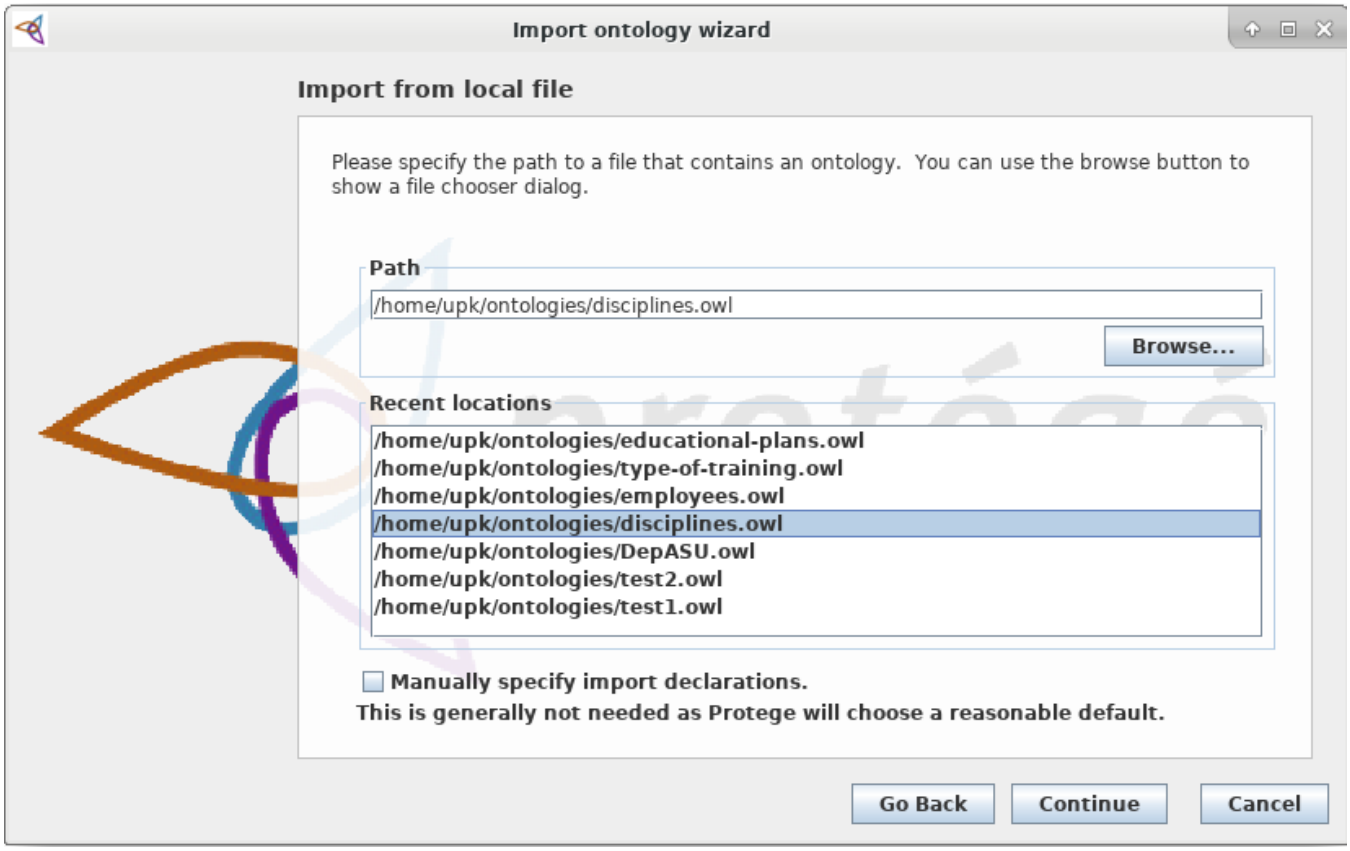

Рисунок 3.38 — Выбор импортируемого файла

Результат импорта нужных файлов показан на рисунке 3.39.

Если перейти на вкладку «***Entities*** → ***Classes***», то мы увидим, как показано на рисунке 3.40, что импортированные онтологии ***СУД*** и ***ТипОбучения*** — полностью доступны в онтологии ***УчебныеПланы***.

Рассотрев заголовочную часть файла ***educational-plans.owl***, как это показано на листинге 3.5, мы увидим каким образом внешние классы подключаются к пространству активной онтологии (см. теги ***owl:imports***).

Таким образом, мы подключили все нужные внешние объекты для последующей реализации онтологии ***УчебныеПланы***.

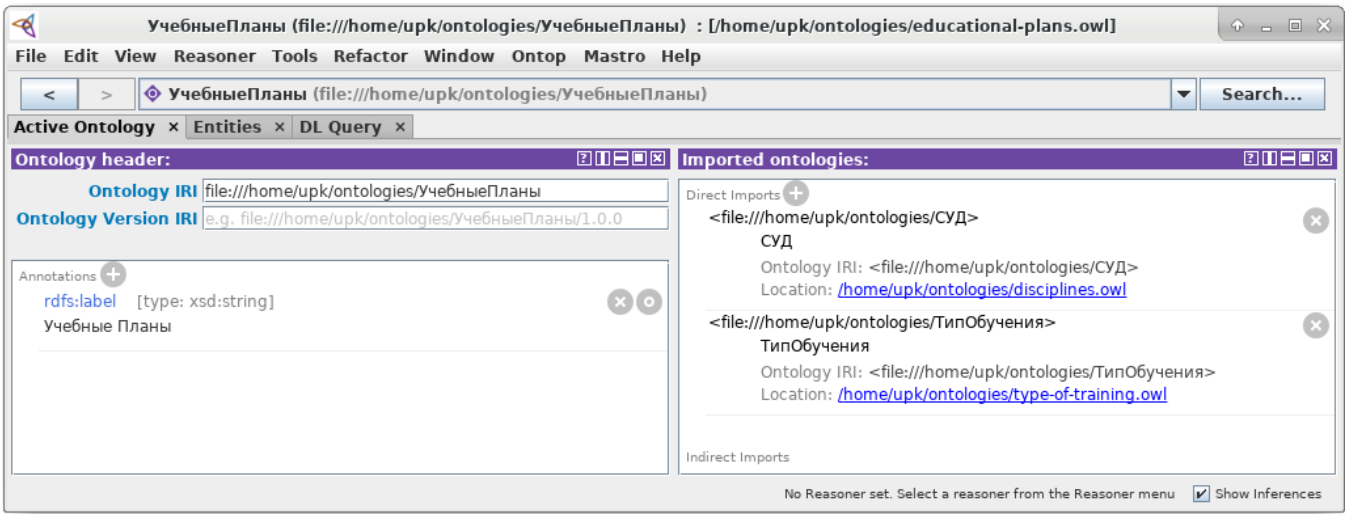

Рисунок 3.39 — Результат импорта онтологий СУД и ТипОбучения

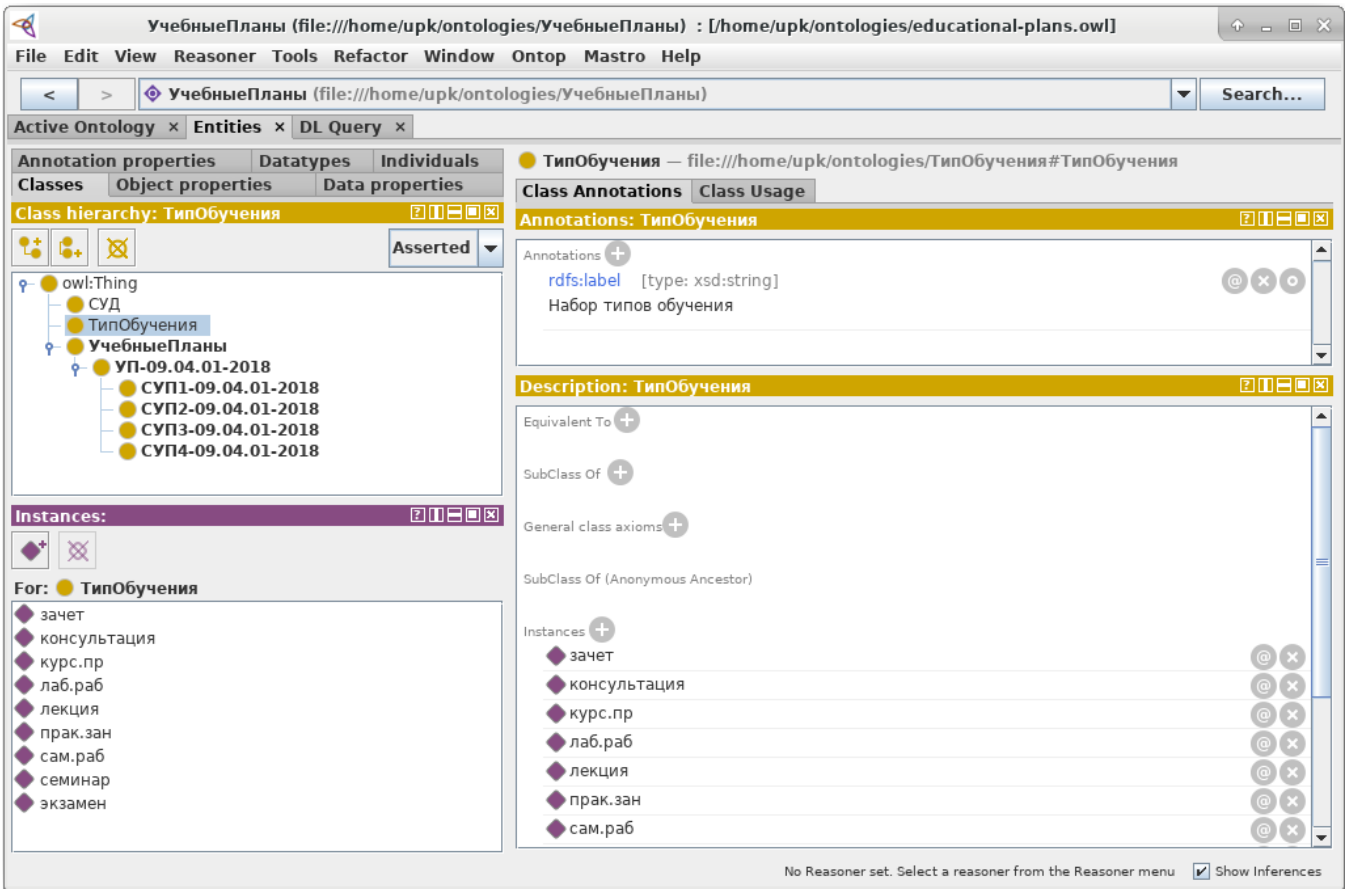

Рисунок 3.40 — Демонстрация доступности онтологий СУД и ТипОбучения в среде онтологии УчебныеПланы

Листинг 3.5 — Заголовочная часть файла educational-plans.owl

```
<?xml version="1.0"?>
<rdf:RDF xmlns="file:///home/upk/ontologies/УчебныеПланы#"
     xml:base="file:///home/upk/ontologies/УчебныеПланы"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
```

```
      xmlns:owl="http://www.w3.org/2002/07/owl#"
      xmlns:xml="http://www.w3.org/XML/1998/namespace"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
    <owl:Ontology rdf:about="file:///home/upk/ontologies/УчебныеПланы">
        <owl:imports rdf:resource="file:///home/upk/ontologies/СУД"/>
        <owl:imports rdf:resource="file:///home/upk/ontologies/ТипОбучения"/>
        <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Учебные
Планы</rdfs:label>
    </owl:Ontology>
```

Конец листинга 3.5

Шаг 3 и другие. Завершающая часть создания онтологии ***УчебныеПланы*** проводится в рамках пункта 3.3.2.5 «Учебные задания».

3.3.2.5 Учебные задания

Аспиранту следует выполнить в системе Protege следующий перечень работ:

1. Освоить и описать в личном отчете технологии, рассмотренные в рамках примеров пунктов 3.3.2.1 — 3.3.2.4.
2. Завершить создание онтологии ***УчебныеПланы*** посредством создания объектов типа (owl:NamedIndividual), обозначающие отдельные планируемые единицы обучения, и включить эти объекты в нужные классы согласно данным почасовой нагрузки, представленной в таблице 3.9;
3. Именовать отдельные идиницы обучения, согласно формату: ***Пример 3.1***.

Таблица 3.9

Тип обучения	***СОС***	***АВК***
Семестр	1	3
Лекции	20	18
Лабораторные работы	36	54
Самостоятельная работа	88	108
Подготовка к кзамену	36	36

Пример 3.1. Формат именования планируемых единиц обучения. Например, для лекций дисциплины СОС объемом 20 академических часов, имеем:

- имя обекта — ***лекцииСОС-20***;
- тип данных (owl:DatatypeProperty) с именем ***объемЧасов***, значением (xsd: unsignedInt) равным 20;
- два предиката (owl:ObjectProperty) с именами «***планируетсяТип***» и «***планируетсяСУД***».

3.3.3 Список использованных источников

3.3.1 OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition): [Электронный документ]. - (https://www.w3.org/TR/owl2-syntax/). Проверено: 19.12.2018.

3.3.2 Internationalized Resource Identifier - Википедия: [Электронный документ]. - (https://ru.wikipedia.org/wiki/Internationalized_Resource_Identifier). Проверено: 19.12.2018.

4 Организация самостоятельной работы аспирантов

Самостоятельная работа при подготовке к выполнению практических работ регламентирована выше и состоит, в основном, в изучении теоретического материала, необходимого не только для для проведения практической работы, но и для формирование у аспирантов необходимой компетенции ПК-3.

4.1 Самостоятельная работа аспирантов при выполнении практических работ

На практических занятиях проводится также собеседование по следующим вопросам:

1. Определение и общая классификация видов информационных технологий. Модели, методы и средства сбора, хранения, коммуникации и обработки информации с использованием компьютеров.
2. Программно-технические средства реализации современных офисных технологий. Стандарты пользовательских интерфейсов.
3. Создание и обработка текстовых файлов и документов с использованием текстовых редакторов и процессоров. Программные средства создания и обработки электронных таблиц.
4. Программные средства создания графических объектов, графические процессоры (векторная и растровая графика).
5. Понятие информационной системы, банки и базы данных. Логическая и физическая организация баз данных. Модели представления данных, архитектура и основные функции СУБД.
6. Распределенные БД.
7. Принципиальные особенности и сравнительные характеристики файл-серверной, клиент-серверной и интранет технологий распределенной обработки данных.
8. Реляционный подход к организации БД. Базисные средства манипулирования реляционными данными. Методы проектирования реляционных баз данных (нормализация, семантическое моделирование данных, ER-диаграммы).
9. Языки программирования в СУБД, их классификация и особенности. Стандартный язык баз данных SQL.
10. Перспективные концепции построения СУБД (ненормализованные реляционные БД, объектно-ориентированные базы данных и др.).
11. Основные сетевые концепции. Глобальные, территориальные и локальные сети. Проблемы стандартизации.
12. Сетевая модель OSI. Модели взаимодействия компьютеров в сети.
13. Среда передачи данных. Преобразование сообщений в электрические

сигналы, их виды и параметры. Проводные и беспроводные каналы передачи данных.

14. Локальные сети. Протоколы, базовые схемы пакетов сообщений и топологии локальных сетей. Сетевое оборудование ЛВС.
15. Глобальные сети. Основные понятия и определения. Сети с коммутацией пакетов и ячеек, схемотехника и протоколы.
16. Принципы межсетевого взаимодействия и организации пользовательского доступа.
17. Методы и средства защиты информации в сетях. Базовые технологии безопасности.
18. Сетевые операционные системы. Архитектура сетевой операционной системы: сетевые оболочки и встроенные средства. Обзор и сравнительный анализ популярных семейств сетевых ОС. Принципы функционирования Internet, типовые информационные объекты и ресурсы.
19. Ключевые аспекты WWW-технологии. Адресация в сети Internet.
20. Методы и средства поиска информации в Internet, информационно-поисковые системы.
21. Языки и средства программирования Internet приложений. Язык гипертекстовой разметки HTML, основные конструкции, средства подготовки гипертекста (редакторы и конверторы). Базовые понятия VRML. Организация сценариев отображения и просмотра HTML документов с использованием объектно-ориентированных языков программирования.
22. Представление звука и изображения в компьютерных системах. Устройства ввода, обработки и вывода мультимедиа информации.
23. Форматы представления звуковых и видеофайлов. Оцифровка и компрессия.
24. Программные средства записи, обработки и воспроизведения звуковых и видеофайлов. Мультимедиа в вычислительных сетях.
25. Основные разделы теории и приложений искусственного интеллекта. Описание и постановка задачи. Задачи в пространстве состояний, в пространстве целей.
26. Классификация задач по степени сложности. Линейные алгоритмы. Полиномиальные алгоритмы. Экспоненциальные алгоритмы.
27. Виды и уровни знаний. Знания и данные. Факты и правила. Принципы организации знаний.
28. Требования, предъявляемые к системам представления и обработки знаний.
29. Формализмы, основанные на классической и математической логиках. Современные логики. Фреймы. Семантические сети и графы. Модели, основанные на прецедентах.
30. Приобретение и формализация знаний. Пополнение знаний.
31. Обобщение и классификация знаний. Логический вывод и умозаключение на знаниях. Проблемы и перспективы представления знаний.
32. Назначение и принципы построения экспертных систем. Классификация экспертных систем.

33. Методология разработки экспертных систем. Этапы разработки экспертных систем. Проблемы и перспективы построения экспертных систем.

4.2 Тестовые задания

Для оценки степени сформированности и уровня освоения закрепленной за дисциплиной компетенции используются следующие оценочные тестовые задания:

Тестовый вопрос № 1 с вариантами ответов:

В январе 1979 года был опубликован формат архиватора для ОС UNIX, который получил сокращенное название ...

a) ar
b) zip
c) gzip
d) tar

Тестовый вопрос № 2 с вариантами ответов:

В ... году Фил Кац, основатель компании PKWARE, опубликовал формат архиватора ZIP.

a) 1998
b) 1979
c) 1988
d) 1989

Тестовый вопрос № 3 с вариантами ответов:

В настоящее время, спецификации на формат zip-архивов являются ...

a) проприетарным продуктом
b) свободными от налогов
c) принадлежат компании PKWARE
d) общественным достоянием

Тестовый вопрос № 4 с вариантами ответов:

Укажите, какие из перечисленных имен файлов, являются форматом zip:

a) Книга.doc
b) xml_type.jar
c) Инструкция.docx
d) Методичка.odt

Тестовый вопрос № 5 с вариантами ответов:

Какой из перечисленных файлов является документом ODF?

a) Книга.doc
b) xml_type.jar
c) Инструкция.docx
d) Методичка.odt

Тестовый вопрос № 6 с вариантами ответов:

sqlite3 является ...

a) компактной встраиваемой СУБД
b) новым языком манипулирования данными
c) ядром СУБД SQLite
d) утилитой доступа к СУБД

Тестовый вопрос № 7 с вариантами ответов:

В файловых системах ОС UNIX и Linux, сокет (socket) является ...

a) каналом связи между процессами
b) точкой доступа к ядру ОС
c) средством передачи сигналов
d) файлом

Тестовый вопрос № 8 с вариантами ответов:

Протокол SCTP обслуживает … уровень модели OSI.

a) канальный
b) прикладной
c) сетевой
d) транспортный

Тестовый вопрос № 9 с вариантами ответов:

JavaScript является … HTML-страниц.

a) протоколом обмена
b) языком разметки
c) технологией программирования
d) встроенным языком

Тестовый вопрос № 10 с вариантами ответов:

WebSocket является … HTML-страниц.

a) протоколом обмена
b) языком разметки
c) технологией программирования
d) встроенным языком

Тестовый вопрос № 11 с вариантами ответов:

AJAX является … HTML-страниц.

a) протоколом обмена
b) языком разметки
c) технологией программирования
d) встроенным языком

Тестовый вопрос № 12 с вариантами ответов:

Apache Tomcat является …

a) языком программирования
b) средством поддержки апплетов
c) языком разметки
d) контейнером сервлетов

Тестовый вопрос № 13 с вариантами ответов:

После первого обращения к JSP-странице, она преобразуется в ...

a) HTML-страницу
b) страницу языка PHP
c) аплет
d) сервлет

Тестовый вопрос № 14 с вариантами ответов:

SMIL является …

a) частью языка JavaScript
b) средством поддержки апплетов
c) контейнером сервлетов
d) языком разметки

Тестовый вопрос № 15 с вариантами ответов:

OWL является языком описания ...

a) HTML-страниц
b) XML-страниц
c) сервлетов
d) онтологий

Тестовый вопрос № 16 с вариантами ответов:

Объекты онтологии описываются конструкций … языка OWL 2.

a) owl:ObjectProperty
b) owl:DatatypeProperty

c) owl:NamedIndividual
d) owl:Thing
e) owl:Class

Тестовый вопрос № 17 с вариантами ответов:

Предикаты онтологии описываются конструкций … языка OWL 2.

a) owl:ObjectProperty
b) owl:DatatypeProperty
c) owl:NamedIndividual
d) owl:Thing
e) owl:Class

Тестовый вопрос № 18 с вариантами ответов:

Классы онтологии описываются конструкций … языка OWL 2.

a) owl:ObjectProperty
b) owl:DatatypeProperty
c) owl:NamedIndividual
d) owl:Thing
e) owl:Class

Тестовый вопрос № 19 с вариантами ответов:

Высшая сущность онтологии описываются конструкций … языка OWL 2.

a) owl:ObjectProperty
b) owl:DatatypeProperty
c) owl:NamedIndividual
d) owl:Thing
e) owl:Class

Тестовый вопрос № 20 с вариантами ответов:

Типы данных онтологии описываются конструкций … языка OWL 2.

a) owl:ObjectProperty
b) owl:DatatypeProperty
c) owl:NamedIndividual
d) owl:Thing
e) owl:Class

Список использованных источников

1 Кориков, А. М. Системный анализ, управление и обработка информации. Часть I: Методические указания по выполнению практических работ и организации самостоятельной работы для аспирантов [Электронный ресурс] / А. М. Кориков, М. П. Силич. — Томск: ТУСУР, 2018. — 40 с. — Режим доступа: https://edu.tusur.ru/publications/7587

2 Резник В.Г. Современные компьютерные технологии. Учебное пособие по направлению подготовки «Информатика и вычислительная техника» по специализации 05.13.01 - «Системный анализ, управление и обработка информации (информация и информационные системы, экономика, энергетика, промышленность, образование)». – Томск, ТУСУР, 2018. – 123 с. [Электронный ресурс]. – Режим доступа: - http://asu.tusur.ru/graduate/spec051301/spec051301-lect1.pdf

3 Резник В.Г. Современные компьютерные технологии. Методические указания по практическим занятиям аспирантов по направлению «09.06.01 - Информатика и вычислительная техника». – Томск, ТУСУР, 2018. – 25 с. [Электронный ресурс]. – Режим доступа: - http://asu.tusur.ru/graduate/spec051301/spec051301-pract1.pdf

4 Резник В.Г. Современные компьютерные технологии. Методические указания по самостоятельной работе аспирантов по направлению «09.06.01 Информатика и вычислительная техника» (профиль 05.13.01 «Системный анализ, управление и обработка информации (информация и информационные системы, экономика, энергетика, промышленность, образование)». – Томск, ТУСУР, 2018. – 18 с. [Электронный ресурс]: - Режим доступа: http://asu.tusur.ru/graduate/spec051301/spec051301-work2.pdf

5 Резник В.Г. Учебный программный комплекс кафедры АСУ на базе ОС ArchLinux. Учебно-методическое пособие. – Томск, ТУСУР, 2017. – 38 с. [Электронный ресурс]. – Режим доступа: - http://asu.tusur.ru/learning/books/b13.pdf

6 Бройдо В.Л., Ильина О.П. Архитектура ЭВМ и систем: Учебник для вузов. – СПб.: Питер, 2006. - 712с.: Библиотека ТУСУР.

7 ГОСТ Р ИСО/МЭК 26300-2010 - Информационная технология. Формат Open Document для офисных приложений (OpenDocument) v1.0: [Электронный ресурс]. – Режим доступа: - http://www.gostedu.ru/50328.html.

8 ГОСТ Р ИСО/МЭК 10032-2007 - Эталонная модель управления данными: [Электронный ресурс]. – Режим доступа: - http://www.docnorma.ru/gost_lib/47691.htm.

9 ГОСТ Р ИСО/МЭК 10032-2007 - Эталонная модель управления данными: [Электронный ресурс]. – Режим доступа: - http://files.stroyinf.ru/Data2/1/4293832/4293832367.htm.

10 Бройдо В.Л. Вычислительные системы, сети и телекоммуникации: Учебное пособие для вузов. – СПб.: Питер, 2006. - 702с.: Библиотека ТУСУР.

11 ГОСТ Р ИСО/МЭК 7498-1-99 - Информационная технология. Взаимосвязь открытых систем. Базовая эталонная модель. Часть 1. Базовая модель: [Электронный ресурс]. – Режим доступа: - http://protect.gost.ru/v.aspx?control=7&id=132355

12 Павлов С.Н. Системы искусственного интеллекта : учеб. пособие. В 2-х частях. / С. Н. Павлов. — Томск: Эль Контент, 2011. — Ч. 1. — 176 с.

13 Павлов С.Н. Системы искусственного интеллекта : учеб. пособие. В 2-х частях. / С. Н. Павлов. — Томск: Эль Контент, 2011. — Ч. 2. — 194 с.

14 Достоверный и правдоподобный вывод в интеллектуальных системах : учебное пособие для вузов / В. Н. Вагин, Е. Ю. Головина, А. А. Загорянская, М. В. Фомина; Ред. Д. А. Поспелов. - М. : Физматлит, 2004. - 704 с. : Библиотека ТУСУР.

15 Антамошин А. Н. Интеллектуальные системы управления организационно-техническими системами. - М. : Горячая линия-Телеком, 2006. - 160 с. : Библиотека ТУСУР.

16 Андрейчиков А.В. Интеллектуальные информационные системы: Учебник для вузов. - М.: Финансы и статистика, 2006. - 423с. : Библиотека ТУСУР.

Учебное издание

Резник Виталий Григорьевич

СИСТЕМНЫЙ АНАЛИЗ, УПРАВЛЕНИЕ И ОБРАБОТКА ИНФОРМАЦИИ. ЧАСТЬ II

Методические указания определяют порядок выполнения практических работ и организацию самостоятельной работы аспирантов по дисциплине «Системный анализ, управление и обработка информации» для направленности (профиля) 05.13.01 – Системный анализ, управление и обработка информации (информация и информационные системы, экономика, энергетика, промышленность, образование) подготовки кадров высшей квалификации, осуществляемой в рамках направления подготовки 09.06.01 – Информатика и вычислительная техника. Практикум по части II этой дисциплины является продолжением практикума по части I. В части II изучаются компьютерные технологии обработки информации. Методические указания определяют также регламент самостоятельной работы при подготовке к выполнению практических работ. Для оценки степени освоения закрепленной за дисциплиной компетенции предложены тестовые задания.

Учебно-методическое пособие

Усл. печ. л. . Тираж . Заказ .
Томский государственный университет
систем управления и радиоэлектроники
634050, г. Томск, пр. Ленина, 40