Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное
образовательное учреждение
Высшего образования
Алтайский государственный технический университет
им. И.И.Ползунова

НАУКА И МОЛОДЕЖЬ – 2017

XIV Всероссийская научно-техническая конференция студентов, аспирантов и молодых ученых

СЕКЦИЯ

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

подсекция

ПРОГРАММНАЯ ИНЖЕНЕРИЯ

Барнаул – 2017

УДК 004

XIV Всероссийская научно-техническая конференция студентов, аспирантов и молодых ученых "Наука и молодежь – 2017". Секция «Информационные технологии». Подсекция «Программная инженерия». / Алт. гос. техн. ун-т им. И.И.Ползунова. – Барнаул: изд-во АлтГТУ, 2017. – 127 с.

В сборнике представлены работы научно-технической конференции студентов, аспирантов и молодых ученых, проходившей 27 апреля 2017 г.

Редакционная коллегия сборника:
Кантор С.А., заведующий кафедрой «Прикладная математика» АлтГТУ – руководитель секции, Крючкова Е.Н., профессор, зам. зав. кафедрой ПМ, Сорокин А.В., доцент каф. ПМ, ответственный за НИРС на кафедре ПМ

Научный руководитель подсекции: к.ф.-м.н., профессор, Кантор С.А.
Секретарь подсекции: к.т.н., доцент, Сорокин А.В.

Компьютерная верстка: Сорокин А.В.

© Алтайский государственный технический университет им. И.И.Ползунова

СОДЕРЖАНИЕ

Ананьев Т.П. Модификация классов на уровне байткода Java 5

Ананьев Т.П., Фаст А.С. Организация системы проведения тренировок по олимпиадному программированию 8

Борисов В.В., Казаков М.Г. Горизонтально масштабируемая система развёртывания сервисов, под управлением микрооперационной системы 13

Гавриченков Е.А., Крючкова Е.Н. Проектирование социальной сети «Tradebook» 17

Елисеев А.Г., Крайванова В.А. Разработка системы визуализации мультимодальной информации об исторических событиях 20

Корней А.О., Крючкова Е.Н. Автоматический анализ эмоционального состояния автора в коротких текстах на естественном языке 22

Метелкин К.О., Астахова А.В. Особенности разработки серверного ПО для системы навигационного мониторинга грузовых перевозок 26

Коновальчук Е.С., Козликина Е.Ю., Копылова Д.А., Астахова А.В. Разработка программного обеспечения по учету кадров 29

Метелкин К.О., Шкирков А.Ю., Васильева О.В., Астахова А.В. Некоторые вопросы управления IT-проектами 31

Савченко И.В., Крючкова Е.Н. Реализация автоматизированной системы анализа плагиата программного кода 36

Самойлов С.С., Крючкова Е.Н. Описание механизма встраивания готового решения в сайт клиента 38

Сидоренко Е.А., Ловцкая О.В., Бубнова Н.Д. Разработка программных модулей расчёта морфометрических и гидрологических характеристик территории для ГИС с открытым кодом 40

Станько И.В., Крючкова Е.Н. Децентрализованная система хранения и потверждения подлиности портфолио студента на основе технологии Blockchain 43

Ткаченко Е.С., Крайванова В.А. Классификация фрагментов текстов по их функциональному назначению 46

Токарев В.И., Крайванова В.А. Иерархическая модель функциональной структуры научных текстов 48

Чарапкин Я.С., Старикова В.К., Троицкий В.С. Автоматизирование учета оборудования в акционерном обществе «РОССЕЛЬХОЗБАНК» 50

Шахов Д.Е. Разработка архитектуры системы идентификации предаварийных ситуаций технологических процессов 51

Шкирков А.Ю., Крючкова Е.Н., Кочанов И.А. Разработка модуля web-приложения для обеспечения обмена данными в режиме реального времени с использованием технологии Websocket 53

Ренёв Н.С., Крайванова В.А. Система визуализации модульной структуры программных продуктов по их текстовым описаниям на примере студенческих работ..... 55

Леденева Е.А., Липатов В.А., Реутов А.С., Сорокин А.В., Рогозин К.И. Опыт адаптации учебных материалов для школьников под персональные мобильные цифровые устройства..... 57

Волков Е.А., Крайванова В.А. Извлечение информации о событиях из статей Википедии..... 59

Волков Е.А., Крайванова В.А. Задача парсинга «прозрачного» синтаксиса на примере вики-разметки..... 61

Аксёнов Р.З., Старолетов С.М. Проектирование системы анализа звукового трафика мобильных разговоров..... 63

Ложкина Д.Д., Старолетов С.М. Online портал для моделирования и верификации распределенного программного обеспечения..... 66

Пасюта А.А., Старолетов С.М. Целесообразность использования модели деревьев поведения по сравнению с автоматной моделью при программировании действий игровых персонажей..... 73

Шевелева А.Г., Старолетов С.М. Целесообразность применения методологии разработки MDD в компании по производству программного обеспечения 76

Козлов Н.С., Крючкова Е.Н. Архитектура интеллектуальной системы по созданию музыкальных произведений 81

Мальков С.А., Крючкова Е.Н. Комплекс клиент-серверных приложений Price - Chekker ... 87

Некрасов Д.В., Старолетов С.М. Разработка самообучающихся алгоритмов для трекинга объектов в видеопотоке..... 89

Дубко А.В., Старолетов С.М. Проектрование верифицирующего расширения компилятора для формальной проверки распределенных приложений 92

Лаптев М.А., Старолетов С.М. Система распознавания марки автомобилей по фотографиям с камер фотофиксации как задача классификации 97

Данильченко Д.Б. Разработка мобильного приложения на платформе UWP для составления графика работ 102

Амосов М.С., Старолетов С.М. Программное обеспечение для статической обработки данных об изменениях файлов проекта в распределённой системе контроля версий GIT 108

Головин М.А., Крючкова Е.Н. Проблемы при разработке диалоговых систем на естественном языке..... 112

Шабанов П.А., Старолетов С.М. Модель для моделирования поведения живых объектов на основе взаимодействующих расширенных вероятностных автоматов..... 116

Галкин Р.Е., Андреева А.Ю. Разработка кроссплатформенного текстового редактора с комбинированным тексто-рукописным вводом 118

Зяблицкий А.С., Кантор С.А., Кошелев К.Б. Исследование возможностей и ограничений конечно-объемной библиотеки Openfoam при разработке решателя квазиодномерных уравнений Сен-Венана 122

МОДИФИКАЦИЯ КЛАССОВ НА УРОВНЕ БАЙТКОДА JAVA

Ананьев Т.П. – студент
Алтайский государственный технический университет (г. Барнаул)

Актуальность

Java – современный объектно-ориентированный язык программирования, получивший широкое распространение во многих сферах разработки. Одной из главных особенностей языка является переносимость разработанных на нём приложений, обеспечиваемая компиляцией не в машинный код, а в байткод – набор инструкций, исполняемых виртуальной машиной Java.

Работа на уровне байткода [1] предоставляет возможность программной модификации уже скомпилированных классов без необходимости иметь доступ к их исходному коду. Эта технология применяется в профайлерах, анализаторах кода, фреймворках и JVM-based языках (Scala, Groovy). Оперируя с байткодом вы можете динамически генерировать классы, обфусцировать код или отдельно выделять сквозной функционал (аспектно-ориентированное программирование), такой как ведение логов, обработку исключений, аутентификацию.

Широкая распространённость технологии и её возможности делают манипуляцию байткодом актуальной в Java-разработке. Опыт работы с ней даёт представление о внутреннем устройстве Java, что может быть очень полезно и при обычной разработке.

Постановка задачи

Java предоставляет широкие возможности для создания модульных приложений. При использовании модулей от сторонних разработчиков может потребоваться их модификация с целью достижения необходимого функционала приложения. Не всегда возможно выполнить эту модификацию с помощью конфигурирования и использования средств, предоставленных разработчиками. Исходный код модулей так же не всегда доступен, а декомпиляция может внести ошибки в случае её применения к большим классам со сложной логикой. Может потребоваться поддержка нескольких версий модулей, отличающихся в необходимых классах, но незначительно отличающихся в участках кода, требующих модификации. В данных ситуациях самым простым решением является модификация на уровне байткода.

С использованием различных вариаций применения данной технологии были решены следующие отдельные задачи:

1. Реализована корректная отрисовка шрифта для русского языка в клиенте игры путем исключения из кода ограничения на ширину символов.
2. Вставка в нужные места (согласно конфигурационным файлам) модулей, реализованных с применением одного API, вызовов событий из другого API для их корректной совместной работы на одном сервере в случаях, когда имеющиеся в сервере методы не могли организовать связь этих API.
3. Замена заданных методов сервера на вызовы собственных более оптимальных реализаций на большом количестве различных версий.
4. Корректная работа модуля на нескольких версиях с классами, содержащими в пути версию сервера, но при этом не отличающимися между версиями описаниями методов и логикой их работы (пример: net.application.server.v1_7_R3.Item, где 1_7_R3 – версия сервера).
5. Профайлинг обращений приложения к сети через HttpURLConnection.
6. Внесение изменений в сложный класс проверки решений системы проведения контестов, где недопустимы неточности и ошибки, которые могли бы возникнуть при декомпиляции.

Получение доступа к байткоду классов

Можно использовать в приложении класс, который был заранее изменён с помошью готовых или специально разработанных инструментов, либо встроить эти инструменты в само приложение и динамически изменять классы в момент их загрузки JVM. При решении задач использовалась только динамическая модификации и далее будет говориться о ней.

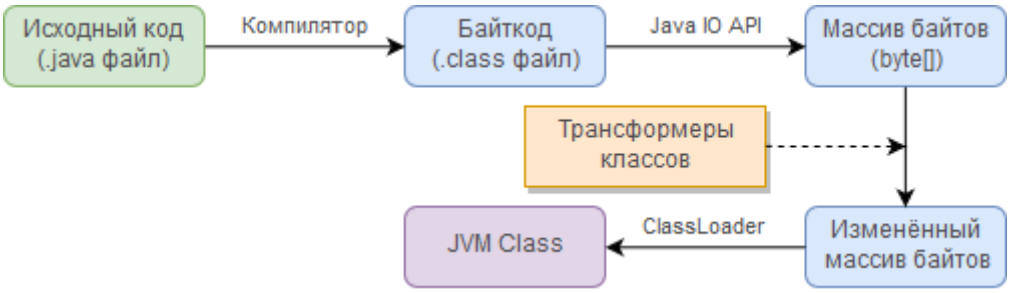


Рисунок 1 – Процесс компиляции в класс и загрузки этого класса JVM

Массив байтов, содержащий байткод класса, может быть изменён (или сгенерирован) перед его попаданием в загрузчик классов (ClassLoader). Доступ сразу же после запуска приложения может быть получен на двух уровнях в зависимости от загрузчика:

1. *Системный загрузчик классов.* Необходимо подключение инструмента как Java Agent [2] с помощью опции JVM -javaagent, затем через предоставленный Instrumentation можно добавить трансформер классов, который будет иметь доступ ко всем загружаемым классам, в том числе стандартным библиотекам Java.
2. В любом месте программы может быть создан *собственный загрузчик классов*, который получает классы из определённого места (например, загрузчик модулей из jar-файлов). Перед передачей загруженного массива байтов методу defineClass загрузчика этот массив может быть изменён, но требуется реализация собственной системы регистрации трансформеров (или использование единственного) и могут быть изменены загружаемые только этим загрузчиком классы.

Трансформер классов

В независимости от используемого загрузчика трансформеры имеют единую структуру, состоящую из главного метода, получающего на вход название класса и байты для изменения и возвращающего байты для загрузчика или следующего в очереди трансформера.

Среди встроенных возможностей Java кроме регистрации трансформеров не представлены средства для работы с байткодом, однако существует большое количество сторонних библиотек и подходов к реализации, рассмотрим некоторые их них, которые использовались при решении поставленных задач.

Работа напрямую с потоком байтов

Самый низкоуровневый подход. Из потока байтов согласно формата считываются отдельные элементы и инструкции, необходимым образом модифицируются и записываются обратно. Требуется написание собственных методов чтения/записи, знание внутренней структуры класса и требований к ней. Не нужно подключение каких-либо библиотек.

Использовался при реализации работы модуля со схожими классами, расположенными в различных пакетах. Для этого из класса считывалась служебная информация и пул констант, в строковых значениях которого через регулярные выражения осуществлялся поиск версии сервера, которая заменялась на текущую. После чего изменённый пул записывался в выходной поток, а остальная часть класса переписывалась без изменений.

Работа на уровне инструкций

Наиболее оптимальный из подходов, требуется знание инструкций байткода, но не принципы их хранения в классе. Наиболее распространённым фреймворком, работающим на этом уровне, является ASM от OW2. Он предоставляет два API для работы – Core API (события для каждого элемента класса) и Tree API (строится представление класса в виде дерева элементов). Core API работает быстрее и менее ресурсоемкий, однако Tree API более удобен и гибок, поэтому при решении задач использовался именно он.

Основной подход, который применялся при модификации, заключается в нахождении в требующем модификации методе заданной инструкции (например, вызов определённого метода или возврат значения) путем обхода и анализа всех инструкций и привязка дальнейшей адресации к ней (или началу/концу метода). Далее, используя эту адресацию, можно определить в какое место необходимо вставить новые инструкции, либо же модифицировать или удалить имеющиеся, и произвести эти действия.

Пример работы с инструкциями с использованием ASM:

```
m.instructions.insertBefore(n, new JumpInsnNode(Opcodes.IFNE, label));
m.instructions.insertBefore(n, new VarInsnNode(Opcodes.ALOAD, 0));
m.instructions.insertBefore(n, new LdcInsnNode("disconnect.endOfStream"));
m.instructions.insertBefore(n, new InsnNode(Opcodes.ICONST_0));
```

Для IDE Eclipse доступен плагин Bytecode Outline, который позволяет посмотреть, как будет представлен в байткоде весь класс или выделенный его фрагмент. Это очень помогает в разработке, позволяя понять, к каким инструкциям осуществлять привязку или же как сгенерировать инструкции, реализующие заданный функционал.

Высокоуровневый фреймворк

Javassit от JBoss позволяет работать на уровне байткода без необходимости знать что-либо о нём. Разработчику, например, предоставляется возможность вставить после определённой строки (задаётся её номером) фрагмент кода, написанный на Java, который будет скомпилирован фреймворком с учетом окружения и вставлен в нужное место.

Пример вставки фрагментов в заданный метод с использованием Javassist:

```
method.insertBefore("if(this.previousTestsFail) return false;");
method.insertAt(576, "if(!passed) this.previousTestsFail = true;");
```

К плюсам подхода можно отнести простоту, малое количество кода для достижения результата и наличие обширной документации. Но он имеет и следующие недостатки: модификация данным методом работает дольше, чем другими, не используются возможности байткода. Если компилируемый фрагмент кода использует обращение к другими классам, то необходимо, чтобы они уже были загружены, что вызывает проблемы, если эти классы также должны быть трансформированы.

Заключение

Таким образом, манипуляция байткодом предоставляет разработчикам различные подходы, которые могут быть использованы для решения обширного круга задач. Знание этой технологии является полезным навыком Java-разработчика.

Список литературы

1. Living in the Matrix with Bytecode Manipulation [Электронный ресурс]. – Режим доступа: https://www.infoq.com/articles/Living-Matrix-Bytecode-Manipulation
2. Java Agent на службе JVM / Хабрахабр [Электронный ресурс]. – Режим доступа: https://habrahabr.ru/post/230239/

ОРГАНИЗАЦИЯ СИСТЕМЫ ПРОВЕДЕНИЯ ТРЕНИРОВОК ПО ОЛИМПИАДНОМУ ПРОГРАММИРОВАНИЮ

Ананьев Т.П., Фаст А.С. – студенты
Алтайский государственный технический университет (г. Барнаул)

На сегодняшний день невозможно представить олимпиаду по программированию без какой-либо автоматизированной системы тестирования решений. Существует большое разнообразие тестирующих систем, сайтов и сервисов, которые позволяют проверять решения на определенном наборе тестов по заданным правилам. Поэтому очевидно, что для подготовки к олимпиадам и использованию таких систем удобно и необходимо использовать одну из них при проведении тренировок.

Перед нами стояла задача организовать систему, которая бы обладала свойствами большинства популярных систем, позволяла добавлять большое количество подготовленных тренером задач, разбивать их на категории по темам, сдавать решения как отдельным участникам, так и командам, в удобном виде следить за результатами членов сборной.

От идеи использования таких сайтов, как Codeforces, в качестве основного средства проведения тренировок мы отказались ввиду очевидной зависимости возможности проведения тренировки от доступности и работоспособности удаленного ресурса. Исходя из этого, было решено взять какую-либо из существующих систем, настроить её и при необходимости изменить под свои нужды.

Тестирующая система в общем случае представляет собой набор программ и библиотек, позволяющих развернуть её на компьютере. В настоящее время популярны такие системы, как Ejudge, Contester, PCMS2 и другие. Из всех доступных систем мы выбрали систему PC2 (Programming Contest Control) [1], разработанную в Калифорнийском Государственном Университете, Сакраменто. Данная система свободно распространяется для проведения соревнований и тренировок на некоммерческой основе. Система постоянно обновляется и поддерживается, хорошо документирована. В связи с этим, она и была выбрана в качестве той системы, которую мы будем устанавливать и настраивать.

Структура системы

Основная особенность системы - это её модульность. В ней выделяют такие компоненты, как Server, Judge, Admin, Scoreboard, JavaServer и Web-интерфейс. Все они могут быть расположены на любом компьютере в интернете, единственное условие – это доступность главного сервера, на котором работает *Server*, объединяющий и координирующий работу остальных модулей. Для запуска всех программ необходимо

указать, по какому адресу располагается сервер, и на какой порт осуществляется подключение, после чего все компоненты системы могут быть запущены и работать вместе, обмениваясь необходимыми данными.

В любой момент времени в системе существует ровно один контест - сущность, объединяющая набор задач, время начала, продолжительность и другие параметры. Для нашей цели - проведения тренировок - мы установили контест длительностью чуть более года (в случае необходимости можно продлить), а задачи условно группируются по названиям - первые два символа названия - номер группы, к которой относится задача.

Admin - это программа для управления всеми параметрами системы. С помощью неё можно, например, добавлять и изменять участников тренировки, задачи, изменять время начала, продолжительность и так далее.

Judge представляет собой программу для тестирования решений участников. Server может обслуживать несколько Judge и распределять решения между ними для ускорения тестирования. Каждому Judge может быть назначен любой набор задач, при наличии нескольких тестирующих компьютеров можно распределять задачи между ними.

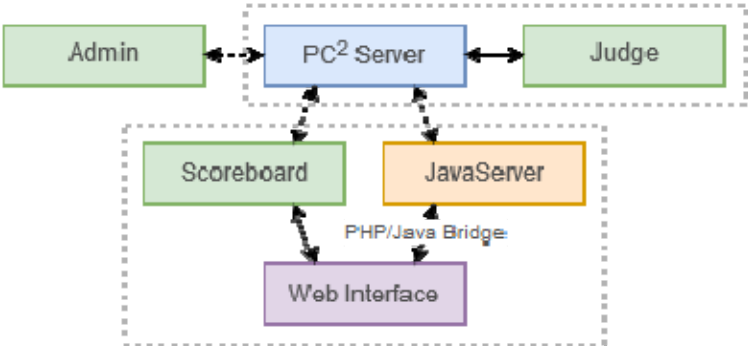


Рисунок 1 – Используемая структура системы

Scoreboard позволяет генерировать “монитор” - таблицу, в которой отражена текущая информация о проведении контеста: время до окончания, положение участников, их посылки, результаты тестирования и так далее.

JavaServer и Web-*интерфейс* используются для предоставления возможности работы с системой с использованием любого веб-браузера без необходимости установки дополнительного программного обеспечения.

Установка и настройка системы

В первую очередь был настроен Server на удаленной машине с “белым” IPv4 адресом под управлением MS Windows Server 2012R2, позволяющий производить подключение других частей системы из любой точки планеты [2].

Затем, на этой же машине был настроен Judge: установлены необходимые компиляторы и интерпретаторы - MS Visual C++ 2015, GNU C, GNU C++, GNU C++11, Java 8, Python 3. Прописаны строки компиляции и запуска, обеспечивающие работу оптимизатора, встроенного в компилятор (при его наличии) и при возможности - добавление опции компиляции ONLINE_JUDGE - как это принято на многих соревнованиях по программированию.

Чекеры

Чекер - это программа, которая по ответу программы участника, входным данным и ответу жюри может решить, правильный ли ответ дала программа участника. В PC^2 чекеры называют валидаторами, но суть от этого не изменяется. Согласно архитектуре системы,

чекер на каждом тесте должен генерировать xml файл определенного формата, в котором записана информация о решении чекера на этом тесте.

По умолчанию в системе есть пять чекеров, позволяющих сравнивать выходные файлы программ участника и жюри на полное совпадение с различными параметрами игнорирования пробелов и переводов строк. Но реальность такова, что во многих задачах этого недостаточно, например, в таких задачах, где ответ не единственный. Сегодня самым популярным способом написания чекеров является использование TestLib - testlib.h для c++, либо testlib.jar для Java. Система же, к сожалению, по умолчанию не поддерживает testlib, но в большом количестве задач используется именно он для реализации чекеров и переписывать каждый чекер вручную займет очень много времени.

Было найдено решение этой проблемы, основанное на изменении testlib.h так, чтобы программа с его использованием при определенных параметрах запуска генерировала xml нужного формата, и реализации прослойки между системой и чекером, которая обеспечивает при необходимости компиляцию исходного кода чекера и запуск его с нужными параметрами. Также мы изменили testlib.jar и при запуске чекера, написанного с его использованием, из classpath подгружаются измененные нами классы, генерирующие правильные xml файлы.

Подготовка задач

Для добавления задачи в систему необходимо указать её название, краткое название, ограничение на время выполнения на каждом тесте, выбрать чекер и прописать строку его запуска, добавить файлы с тестами.

Авторы соревнований и задач обычно распространяют задачи в известном формате - в виде каталога с определенной структурой файлов, где находятся файлы с тестами, с ответами жюри, решениями, чекером с другими данными, связанными с задачей. У разных авторов данная структура разная, это вызвано использованием разных тестирующих систем, но, как правило, у одного автора от контеста к контесту она не меняется.

Мы изучили файловые структуры нескольких авторов и реализовали программу, позволяющую в автоматическом режиме преобразовывать связанные с задачей данные в формат, поддерживаемый системой.

В среднем, при использовании данной программы можно до четырёх раз сократить время на добавление одного контеста в систему, так как проводятся только рутинные операции, используются уже подготовленные файлы с тестами и чекеры, нет необходимости их переименовывать, перемещать и так далее.

Модификация системы

Некоторые возможности, которыми обладают большинство систем проведения контестов, не реализованы в PC^2, для их добавления потребовалась её модификация. Система написана на Java, внесение изменений в её классы производится динамически при загрузке с помощью фреймворка манипуляции байткодом Javassist. Нами была реализована система загрузки патчей, подключаемая через Java Agent, позволяющая полностью или частично заменять классы для внесения изменений в них.

Было реализовано сохранение для последующего отображения пользователям информации о порядковом номере первого теста, который не был пройден решением, и наибольшее время выполнения программы среди всех выполненных тестов. Для сохранения этой информации были использованы сообщения участникам от жюри, задание которых не было предусмотрено в интерфейсе системы, но передача между модулями, сохранение и отображение было реализовано. Это позволило хранить и передавать данные без необходимости вносить изменения в передачу данных по сети или формат хранения данных, реализованный через сериализацию/десериализацию классов.

Вместо сохранения вывода программы и чекера на последнем запущенном тесте реализовано сохранение и отображение на первом, который обычно известен пользователям из условия задачи и эта информация позволяет проверить работу ввода/вывода решения и определить возможные особенности компиляторов или интерпретаторов, используемых в системе. Размер сохраняемых файлов был ограничен для уменьшения занимаемого места и передаваемого между модулями трафика, в случае отсечения части файла в его конец добавляется информация об этом.

В механизм тестирования были внесены изменения, останавливающие тестирование после первого непройденного решением теста. Добавление вывода времени работы позволило нам обнаружить ошибки в имеющемся алгоритме проверки этого параметра. После анализа были найдены причины и способы их исправления. Алгоритм позволял решению работать на 1-2 секунды дольше допустимого лимита, это было связано с очень большим шагом таймера, проверяющего время выполнения, а также неверного вычисления и проверки этого времени, работавшего с секундами вместо миллисекунд.

Web-интерфейс

Взамен предоставляемого с PC^2 Web-интерфейса от Восточного Вашингтонского Университета, нами был разработан собственный на его основе. Для снижения нагрузки на основной сервер тестирования Web-интерфейс и связанные с ним модули были перенесены на другой сервер под управлением операционной системы SUSE Linux Enterprise Server 11.

Серверная часть интерфейса реализована на PHP и работает на связке Nginx + Apache. Скрипты через библиотеку PHP/Java Bridge связаны с реализованной на Java обёркой над API PC^2 (JavaServer), позволяющей устанавливать соединение пользователя с сервером тестирования, запрашивать с него данные и отправлять решения задач. Также на сервере запущен модуль, осуществляющий построение таблицы результатов.

В первую очередь изменена система авторизации. Она создаёт для пользователя соединение с сервером тестирования при первом входе, а затем поддерживает его без разрывов. Такая реализация позволяет пользователю оставаться авторизованным продолжительное время без необходимости повторного ввода пароля, работать одновременно из нескольких браузеров. На сервере тестирования исключаются предупреждения о переподключении пользователей. В случае если при проверке авторизации обнаружится, что пароль для данного пользователя был изменён, то соединение не разрывается, но будет выдан запрос на повторный ввод пароля.

< Hide all contests >

AltSTU #1 Infinite Training Contest

Last updated Sat Apr 08 19:42:33 NOVT 2017

< Full monitor >

01 - Waterloo - 1 October, 2016

#	Name	Score	01.A 3/4	01.B 3/6	01.C 4/6	01.D 4/5	01.E 4/14	01.F 2/4	Time
1	judge	6	+	+	+	+	+	+	171684
2	Dasha	6	+	+	+1	+	+5	+2	234673
3	Timofey	5	+	+3	+1	+1	+3	–	105573
4	Artem	3	-1	–	+	+	+2	–	118463

02 - Waterloo - 19 June, 2016

03 - PTZ 2017w - Jagiellonian U

#	Name	Score	03.A 1/1	03.B 1/1	03.C 1/1	03.D 1/1	03.E 1/1	03.F 1/1	03.G 0/1	03.H 2/2	03.I 1/1	03.J 1/1	03.K 0/0	03.L 2/2	Time
1	judge	10	+	+	+	+	+	+	–	+	+	+	–	+	726056
2	Timofey	1	–	–	–	–	–	–	–	–	–	–	–	+	74982
3	Dasha	1	–	–	–	–	–	–	-1	+	–	–	–	–	75854

Рисунок 2 – Таблицы с результатами тестирования решений

Клиентская часть была полностью переделана, разработан новый более легкий и удобный для использования интерфейс. Используется фреймворк jQuery [3]. С использованием подхода AJAX реализована проверка данных при авторизации, отправка решений, обновление монитора и подгрузка результатов тестирования в фоновом режиме без обновления страницы. Проверка результатов осуществляется с различными интервалами в зависимости от ситуации: при просто открытом мониторе установлен большой интервал, чтобы была возможность увидеть отправленные из другого браузера посылки, но при этом не нагружать сеть; после отправки решения частота обновления повышается, чтобы пользователь раньше узнал результаты, после получения которых интервал возвращается к своему стандартному значению.

На основе монитора, генерируемого модулем Scoreboard, выполняется автоматическое построение нового, включающего отдельные таблицы результатов для каждой категории задач. В конфигурационных файлах задаются префиксы для категорий, их названия и ссылки на условия. Имеется возможность свернуть определённые таблицы, а в случае, если с момента сворачивания, в них появилась информация о новых посылках, отображается специальный индикатор.

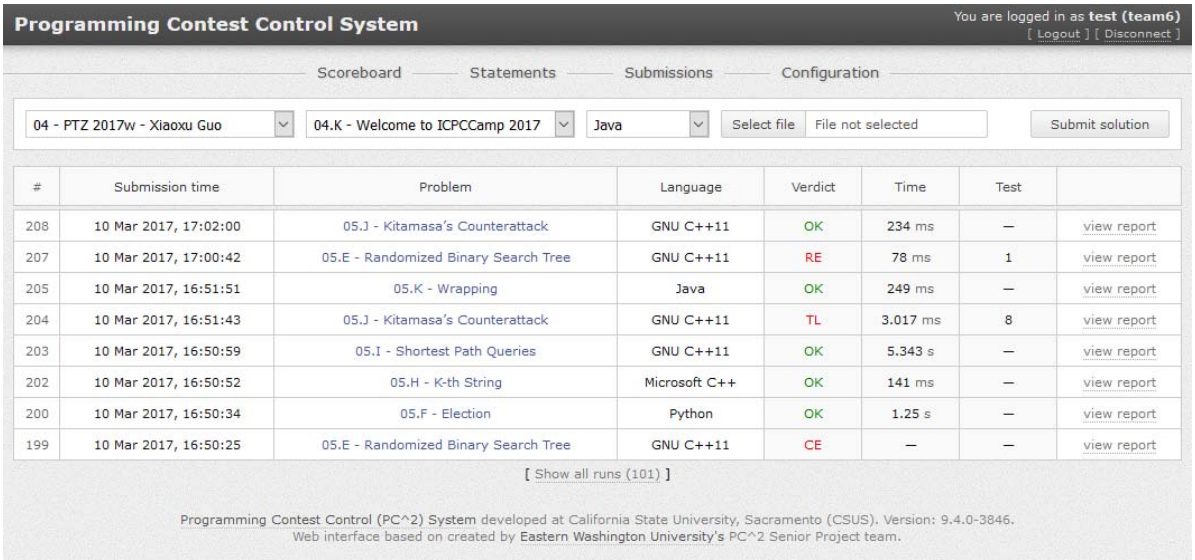


Рисунок 3 – Страница отправки и просмотра результатов тестирования

На совмещённой странице отправки и просмотра результатов тестирования решений отображается в удобном виде вся необходимая участникам информации, добавлена возможность запросить с сервера тестирования и посмотреть файлы, связанные с посылкой, такие как исходный код программы, который отображается с подсветкой синтаксиса соответствующего языка, логи компиляции и выполнения на первом тесте. Запрошенные файлы сохраняются в памяти для ускорения доступа к ним при повторных открытиях отчета.

Заключение

В результате работы нами была организована система для проведения тренировок по олимпиадному программированию, отвечающая поставленным к ней требованиям. Существенных изменений в архитектуру не вносилось, но были изменены многие отдельные аспекты, которые повысили стабильность и качество работы системы, её удобство для пользователей.

Список литературы

1. PC^2 Version 9.4 Contest Administrator's Installation and Configuration Guide [Электронный ресурс]. – Режим доступа: https://pc2.ecs.csus.edu/doc/v9/9.4.0/pc2v9AdminGuide.pdf
2. Таненбаум Э. Компьютерные сети. 5-е изд. / Э. Таненбаум, Д. Уэзеролл. – СПб.: Питер, 2016. – 960 с.
3. Хольцнер С. jQuery. Практическое применение / Стивен Хольцнер ; [пер. с англ.]. – М.: Эксмо, 2010. – 224 с.

ГОРИЗОНТАЛЬНО МАСШТАБИРУЕМАЯ СИСТЕМА РАЗВЁРТЫВАНИЯ СЕРВИСОВ, ПОД УПРАВЛЕНИКМ МИКРООПЕРАЦИОННОЙ СИСТЕМЫ

Борисов В.В. - студент, Казаков М.Г. – к.т.н., ст. преподаватель
Алтайский государственный технический университет (г. Барнаул)

Во многих IT компаниях, для обработки однотипной информации в больших объёмах используют микросервисы [1]. Архитектурный стиль микросервисов — это подход, при котором единое приложение строится как набор небольших сервисов, каждый из которых работает в собственном процессе и коммуницирует с остальными используя легковесные механизмы, как правило HTTP. Эти сервисы построены вокруг бизнес-потребностей и развертываются независимо с использованием полностью автоматизированной среды. Существует абсолютный минимум централизованного управления этими сервисами [2]. Сами по себе эти сервисы могут быть написаны на разных языках и использовать разные технологии хранения данных.

Сервис [3] представляет из себя программный продукт, выполняющий узконаправленные действия, такие как: конвертирование аудио файла из одного формата в другой, поиск на изображении лиц и т.д. Сервис запускается в реальной операционной системе (Windows или Unix подобных) на виртуальной машине. Для запуска, контролирования виртуальных машин, а также масштабирования всей системы используют гипервизор.

В 2016 году на конференции CppCon [4] был представлен рабочий прототип проекта одномодульной микрооперационной системы IncludeOS [5]. IncludeOS предоставляет минимальное необходимое самодостаточное окружение, которое взаимодействует непосредственно с гипервизором и предоставляет загрузчик, ядро системы, минимальный набор библиотек, модулей и драйверов, достаточный для выполнения кода на языке C++, написанный с использованием стандартной библиотеки классов [6]. Окружение компонуется с предназначенным для выполнения приложением и оформляется в виде загрузочного образа виртуальной машины, образуя готовый облачный сервис. Из систем виртуализации, в которых могут работать подобные окружения, поддерживаются KVM/Linux, VirtualBox и QEMU. Схематичное представление построение загрузочного образа виртуальной машины с готовым микросервисом изображён на рисунке 1.

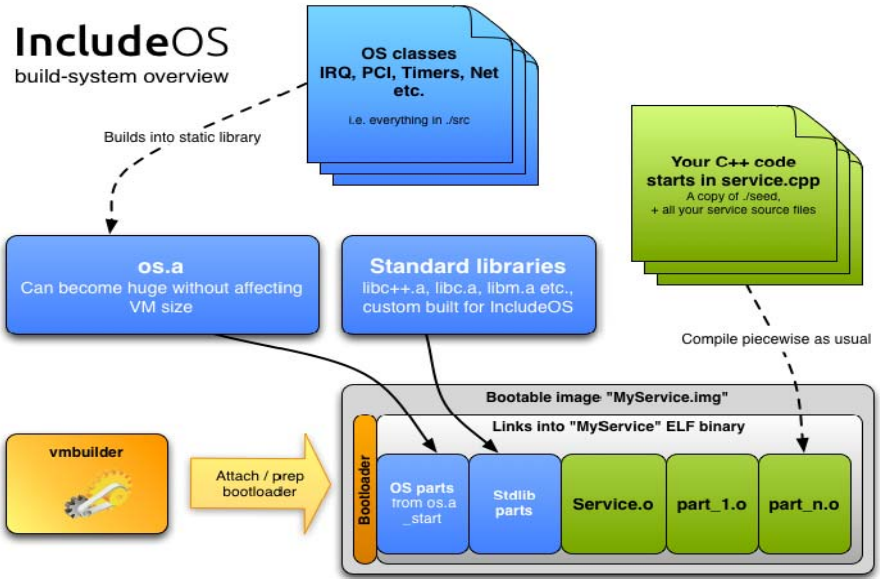


Рисунок 1 - Построение образа виртуальной машины

IncludeOS, в основном, разрабатывается для выполнение web сервисов. Суммарный размер библиотек и компонентов скомпилированной операционной системы составляет около 1 Мб [7]. На тестах с операционной системой Ubuntu, IncludeOS показал значительно меньшое потребление ресурсов реального хост – компьютера, а именно: занимаемое пространство на жёстком диске меньше в 300 раз; использование оперативной памяти меньше в 280 раз [8]. Сравнительная диаграмма используемых ресурсов хост – компьютера для Ubuntu и IncludeOS изображена на рисунке 2.

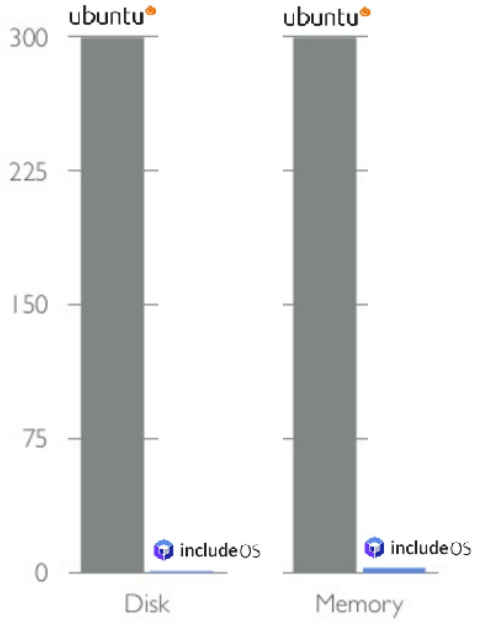


Рисунок 2 - Используемые ресурсы Ubuntu и IncludeOS

В связи с этим использование IncludeOS в качестве операционной системы для вируальной машины в настоящее время в разы эффективней, чем с любой другой

операционной системы, т. к. на одних и тех-же аппаратных мощностях можно запускать в разы больше виртуальных машин.

Так как IncludeOS относительно новый проект и на данный момент поддерживаются только системы виртуализации: KVM/Linux, VirtualBox и QEMU, то запустить виртуальные машины под управлением данной операционной системы, для работы сервисов на наиболее известных гипервизорах, таких как VMware ESX Server [9], XenServer [10], Citrix [11] и других не получится. Исходя из изложенного, отказываться от преимуществ IncludeOS будет ошибкой, в связи с этим было принято решение о разработке собственного менеджера виртуальных машин с возможностью запуска, остановкой, просмотра состояния нагрузки виртуальных машин и т. д.

Постановка задачи

Необходимо разработать программный комплекс, в который будет входить: клиентский компонент системы, для формирования данных для обработки сервисами (передача данных для обработки и получение результата обработки); сервер для мониторинга виртуальных машин, который будет в зависимости от загруженности виртуальных машин принимать решения о запуске или остановки сервисов, а также будет посредником для передачи данных для обработки от клиента сервисам; контроллер будет установлен на хост-машину и предназначен для запуска, остановки, сбора информации о загрузки виртуальной машины; консоль администратора, будет устанавливаться параллельно с клиентским компонентом для просмотра состояния загруженности виртуальных машин .

Поставленная выше задача была разбита на следующие подзадачи:

- Проектирование механизмов взаимодействие компонентов системы. Включает в себя четыре механизма взаимодействия:
  1. Проектирование протокола запроса на исполнение задачи;
  2. Проектирование протокола запроса на выдачу информации о состоянии виртуальной машины;
  3. Проектирование протокола управления состоянием виртуальной машины;
  4. Проектирование протокола запуска и остановки экземпляров виртуальных машин на хост-компьютере;
- Разработка серверного компонента. Включает в себя четыре подсистемы:
  1. Разработка подсистемы перенаправления запросов на исполнения задачи;
  2. Разработка подсистемы сбора данных о состояниях виртуальных машин;
  3. Разработка подсистемы принятия решений о горизонтальном масштабировании;
  4. Разработка подсистемы отображения информации о текущей загрузки системы;
- Разработка компонента виртуальных машин. Включает в себя три подсистемы:
  1. Разработка веб-сервиса, исполняемого в виртуальной машине;
  2. Разработка обработчика запросов на предоставление информации о системе и запросов на управление;
  3. Разработка контроллера виртуальной машины;
- Разработка клиентского компонента.
- Разработка консоли администратора.

Планируемый метод решения поставленной задачи

Для взаимодействия клиента с системой будет реализован **клиентский компонент системы**. С помощью данного компонента пользователь сможет предоставлять исходные данные для обработки сервисами, а также получать и просматривать обработанные данные. Данный компонент будет написан на языке программирования Java.

Сервер будет использоваться для следующих возможностей:

1. Для принятия решений о запуске и остановки виртуальных машин на хост —

компьютере.
2. Для передачи исходных данных от клиентского компонента системы для обработки сервисами.
3. Для получения обработанных данных от витальных машин и передачи этих данных клиентскому компоненту.

Сервер будет представлять из себя Java сервлет, принимающий запросы и возвращающий ответа как от клиента, так и от контроллера.

Запуск **контроллера виртуальных машин** планируется на Unix подобной операционной системе, так как сборку образа IncludeOS с необходимым сервисом можно проводить через командную оболочку, а именно запуск всех скриптов по сборке и запуску виртуальных машин. Контроллер будет представлять из себя программное обеспечение написанное на языке программирования java, установленное на хост – компьютере. Контроллер будет взаимодействовать с виртуальными машинами следующим образом:

- При инициализации всей системы на хост машине должна быть запущена хотя-бы одна виртуальная машина.
- При поступлении новой задачи на обработку, сервер будет опрашивать все доступные виртуальные машины, поле обнаружения менее загруженной виртуальной машины ей будут отправлены новы данные на обработку или запрос на запуск/остановку виртуальных машин.
- Контроллер через перенапрвление вывода консоли виртуальной машины считывает полученные данные.
- При получении новых данных, контроллер обрабатывает их и совершает одно из трёх действий:
  - Запустить ещё n образов виртуальных машин с новыми данными;
  - Остановить n образов виртуальных машин;
  - Ничего не делать.

Консоль администратора разрабатывается для слежением за состоянием виртуальных машин и отображение графиков их состояний. Данный компонент будет написан на языке программирования Java.

Сервис каждой виртуальной машины будет дополнен модулем, который выполняет следующие действия:

1. В определённый промежуток времени отправляет информацию о себе, а именно:
   1. Использование вычислительной мощности реального процессора, занимаемое виртуальной машиной;
   2. Занимаемый объём реальной оперативной памяти и т.д.
2. Запрашивает данные на обработку и запрос на запуск/остановку виртуальных машин у сервера
3. Выгружает данные в свою консоль для чтения этих данных контроллером.

Список литературы

1. Микросервисы (Microservices) [Электронный ресурс]. - Режим доступа: https://habrahabr.ru/post/249183/
2. Преимущества и недостатки микросервисной архитектуры [Электронный ресурс]. - Режим доступа: http://eax.me/micro-service-architecture/
3. Микросервисы на практике - SmartMe University [Электронный ресурс]. - Режим доступа: http://smartme.university/course/microservices-in-practice/
4. CppCon 2016 [Электронный ресурс]. - Режим доступа: https://cppcon2016.sched.com/
5. CppCon 2016: #Include <os>: from bootloader to REST API with the new C++ [Электронный ресурс]. - Режим доступа:

https://cppcon2016.sched.com/event/7nLe/include-ltosgt-from-bootloader-to-rest-api-with-the-new-c
6. В рамках проекта IncludeOS развивается ядро для обособленного запуска C++ приложений [Электронный ресурс]. - Режим доступа: https://www.opennet.ru/opennews/art.shtml?num=43444
7. IncludeOS [Hatred's Log Place] — Programming [Электронный ресурс]. - Режим доступа: https://htrd.su/wiki/zhurnal/2016/09/22/includeos
8. #Include os [Электронный ресурс]. - Режим доступа: https://www.slideshare.net/IncludeOS/include-ltos-from-bootloader-to-rest-api-with-the-new-c
9. VMware ESX [Электронный ресурс]. - Режим доступа: http://www.vmware.com/ru/products/esxi-and-esx.html
10. XenServer | Open Source Server Virtualization [Электронный ресурс]. - Режим доступа: https://xenserver.org/
11. Citrix [Электронный ресурс]. - Режим доступа: https://www.citrix.ru/

ПРОЕКТИРОВАНИЕ СОЦИАЛЬНАЯ СЕТЬ «TRADEBOOK»

Гавриченков Е.А. – студент, Крючкова Е.Н. – к.ф.-м.н., профессор
Алтайский Государственный Технический Университет им. И.И. Ползунова

Издавна люди старались объединиться в различные организации и сообщества по интересам. Развитие же информационных технологий позволяет общаться людям, объединенным общими идеями и мыслями, со всего света. Зачастую общение людей с разных точек планеты приносит не только моральное удовлетворение, но и непосредственную материальную выгоду. Одним из форматов такого общения являются социальные сети. [1]

В частности в данной статье описывается процессы описания предметной области, анализа требований и проектирования архитектуры серверной части социальной сети трейдеров, получившей название «Tradebook». Сервер должна уметь закачивать необходимую информацию с внешних площадок и предоставлять API для клиентов с фронтенда.

Предполагается, что с момента запуска системы количество пользователей в системе достигнет 10000 в течение 6 месяцев.

Для унификации были приняты следующие определения:

Пост – отдельно взятое сообщение пользователя.

Акция - ценная бумага.

Котировка - цена (курс, процентная ставка) товара, которую объявляет продавец или покупатель и по которой они готовы совершить покупку или продажу.

Площадка - внешний источник данных (котировок), на котором торгуется та или иная акция.

Тег - краткий идентификатор внешней сущности, используемый в теле поста: могут быть для пользователя, акции и хештеги.

Хештег — тип тега, облегчающий поиск сообщений по теме или содержанию. Представляет собой слово или объединение слов, которому предшествует символ #, например: #искусство, #техника, #видео.

Подписка - список тегов, интересующие пользователя

Взаимосвязи между данными сущностями представлены на рисунке 1.

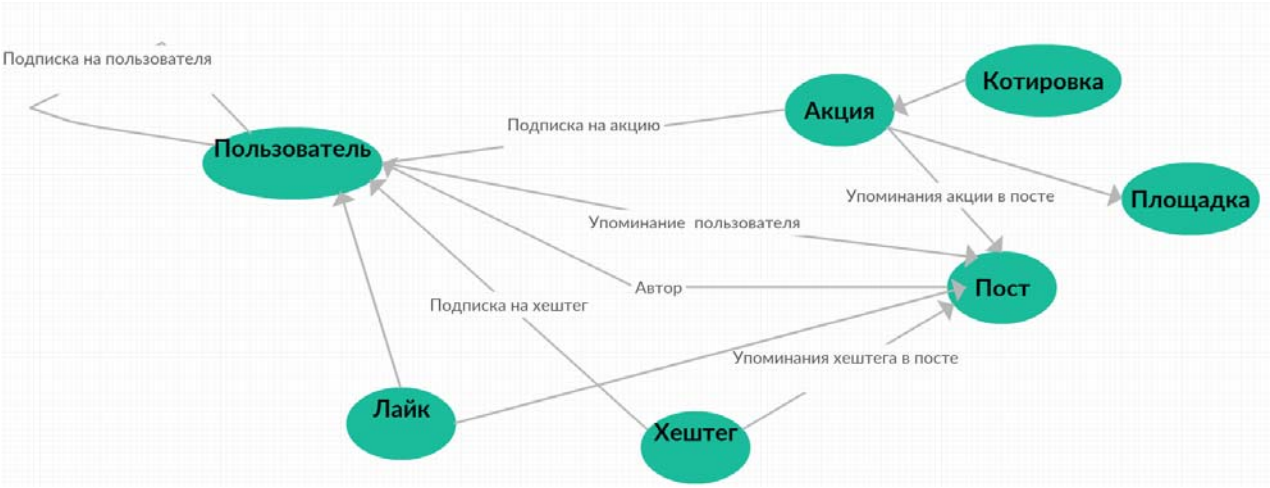


Рисунок 1 - Диаграмма сущностей системы и их взаимосвязей

Данные об акциях и котировках на сервер поступают с различных внешних торговых площадок (пр. YahooFinance [2]), причем изменение котировки происходит несколько раз в секунду. Одним из требований является хранение некоторой истории котировок, вследствие чего количество данных в системе будет очень большим.

Более того, будут сохраняться все переписки, что со временем также приведет к разрастанию базы данных, вследствие чего вопрос правильного выбора модели БД и самой СУБД становится особенно актуальным.

Выбор модели БД шел между традиционными реляционными и графовыми. Преимуществом первых является поддержка транзакций и независимость данных, недостатками же – зачастую низкая скорость работы (особенно при большом количестве операций join) и больший расход памяти.

Преимуществами графовых СУБД считается универсальность и возможность доступа к данным через отношения, недостатком же – сложность в проектировании.

Учитывая особенности требований и данных моделей БД, на первом этапе была выбрана реляционная СУБД PostgreSQL[3].

Важным моментом является формирование удобной схемы базы данных, которая будет максимально отражать реальные процессы, и сможет впоследствии расшириться. Для ее составления был проведен анализ тех понятий и их свойств, которые используются в предметной области. Схема базы данных представлена на рисунке 2.

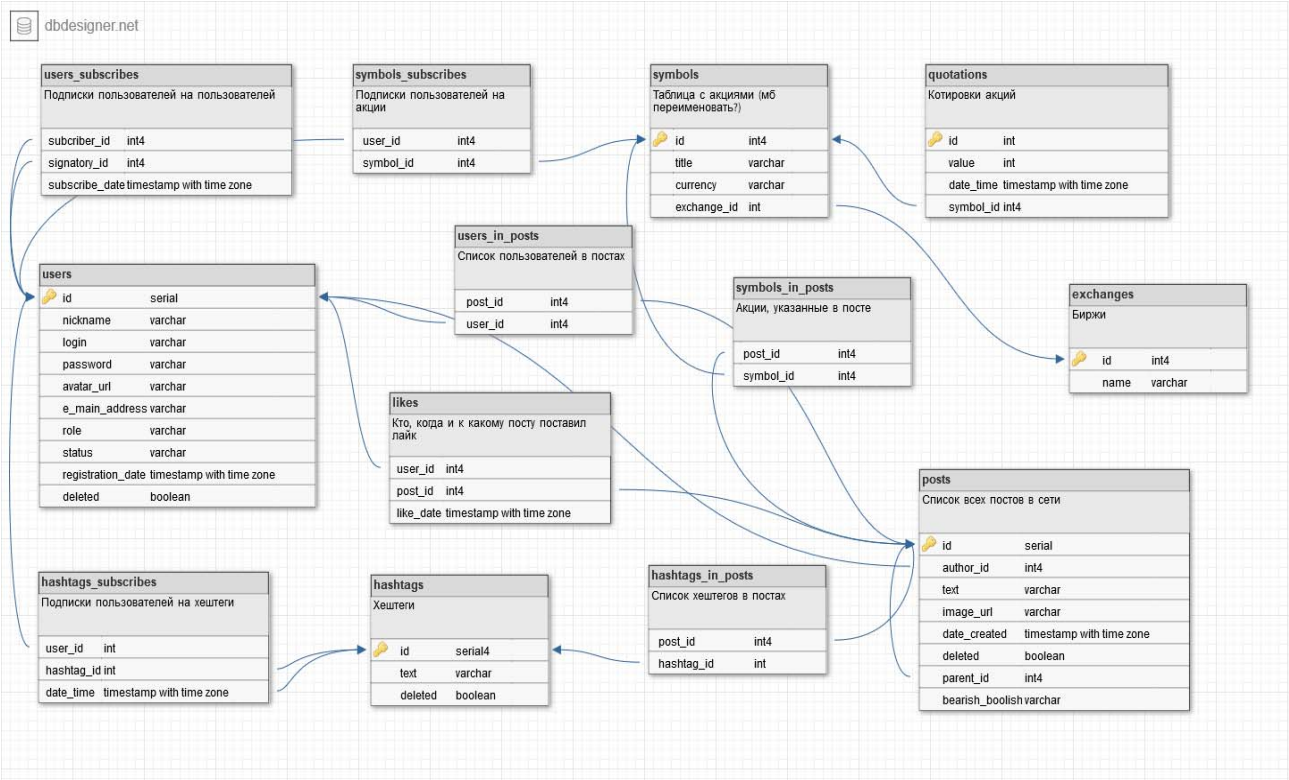


Рисунок 2 – схема базы данных системы.

Для взаимодействия с клиентом было решено использовать два механизма.

Первый – REST API – будет использоваться для приема каких-то сообщений со стороны клиента и ответа на них. Примерами является написание сообщения, регистрация и авторизация и т.п.

Вторым механизмом является Web Sockets - протокол полнодуплексной связи поверх TCP-соединения, предназначенный для обмена сообщениями между браузером и веб-сервером в режиме реального времени. Он будет использоваться для оповещения клиентов о случившихся событиях, таких как отправка сообщений другим пользователем, изменение котировки акции и т.д.

Для написания данной системы будет использована технология Java вкупе с фреймворком Spring[4]. Web Sockets предполагается реализовывать с помощью фреймворка Jetty (входящей в состав Spring), для взаимодействия с базой данных – jOOQ.

Таким образом, в данной статье были описаны процессы формирования предметной области для социально сети трейдеров «Tradebook», проанализированы требования к системе, составлен первичный вариант модели данных и выбраны средства для реализации. В настоящее время система находится на этапе дальнейшего уточнения требований и подготовки базовых механизмов программы.

Список литературы

1. Hendrickson, Mark. Nine Ways to Build Your Own Social Network. [Электронный ресурс]. – TechCrunch. - July 24, 2007. – Режим доступа: http://www.techcrunch.com/2007/07/24/9-ways-to-build-your-own-social-network/
2. Описание взаимодействия с торговой площадкой YahooFinance [Электронный ресурс]. – Режим доступа: http://financequotes-api.com/
3. Документация PostgreSQL [Электронный ресурс]. – Режим доступа: https://postgrespro.ru/docs/postgresql/9.6/
4. Spring documentation [Электронный ресурс]. – Режим доступа: https://spring.io/docs

РАЗРАБОТКА СИСТЕМЫ ВИЗУАЛИЗАЦИИ МУЛЬТИМОДАЛЬНОЙ ИНФОРМАЦИИ ОБ ИСТОРИЧЕСКИХ СОБЫТИЯХ

Елисеев А.Г. – студент, Крайванова В.А. – к.ф.-м.н., доцент
Алтайский государственный технический университет (г. Барнаул)

К настоящему времени человечество накопило и агрегировало огромное количество информации. С расширением влияния интернета объем неструктурированных, «сырых» данных растет. Однако, эффективно работать с такими объемами данных человек не может. Нужно сначала автоматизированными средствами отфильтровать данные, выбрать нужные, выявить среди них зависимости, например, с помощью методов машинного обучения. Но не менее важной является задача визуализации полученных результатов, для представления их пользователю в удобном для человеческого восприятия виде. В данной работе рассматривается задача визуализации выявленных зависимостей между различными событиями.

Для реализации системы отображения в качестве платформы был выбран Web. Такое построение сервиса позволит реализовать отображение в качестве библиотеки, которая будет охватывать большое количество устройств, включая мобильные, вне зависимости от операционной системы, а позже может быть оформлена в качестве отдельного REST сервиса.

Ключевые особенности выбранной задачи:

- Число фактов, событий велико
- Связи между ними носят неоднозначный характер, часто могут пересекаться и быть довольно запутанными (например, формальным поводом для начала Первой мировой войны является убийство эрцгерцога Франца Фердинанда, однако общепризнано, что убийство послужило лишь «толчком» к войне и истинные причины лежат гораздо глубже)
- События упорядочены по времени (например, Первая мировая война началась и закончилась раньше Второй)
- Каждое событие помимо информации о связи с другими событиями содержать дополнительные сведения об участниках, ключевых местах и др. (например, битва при Бородино может содержать сведения о численности армий, командующих и другие)
- События могут быть сгруппированы (например, можно рассмотреть Великую Отечественную войну, как большое событие и попытаться выявить предпосылки к ней, но война содержит в себе вложенные события меньшего масштаба, например, конкретные сражения такие как битва за Москву

В рассмотренной задаче события представляют собой отрезки на временной шкале, размеченные дополнительными свойствами. Временная шкала и определенный порядок между событиями (по датам) является опорной точкой при построении системы. Также характерной особенностью является вложенность событий, которую необходимо учесть.

На рынке присутствует много готовых решений для отображения timeline графиков, самые известные из них Google Charts[1], Vis.js[2]. Но у данных решений нет поддержки отображения зависимостей между элементами и вложенных элементов.

Разрабатываемая библиотека взаимодействует с хранилищем данных о событиях через REST-интерфейс, который реализует следующие сценарии:

- Подгрузка событий, которые принадлежат указанному промежутку времени
- Подгрузка причин и следствий для заданного события
- Подгрузка вложенных событий

Для описания каждого события используется структура Event{id, name, start_date,end_date}. Где id – идентификатор события, name – название, start_date – дата начала, end_date – дата окончания.

Структура работы библиотеки визуализации:

1. Запросить у сервера события, принадлежащие заданному интервалу
2. Получить данные от сервиса распознавания в виде массива событий Event
3. Запросить у сервера причины и следствия для события с заданным id
4. Получить ответ от сервера в структуре:
   {
   reason: [{id,start_date,end_date,name }],
   result: [{id,start_date,end_date,name }]
   }
   Где reason - массив причин типа Event, result - массив следствий типа Event
5. Запросить у сервера для события с заданным id вложенные
6. Получить ответ от сервера в виде
   {
   nested: [{id,start_date,end_date,name }]
   }
   Где nested - массив вложенных событий типа Event

Сценарии работы в пользовательском интерфейсе:

1. Если событие содержит вложенные, по двойному клику график зуммируется и отображаются вложенные события
2. Клик по событию вызывает отрисовку связей (стрелки для причин и следствий выбранного события)
3. Скроллинг вызывает масштабирование линии времени, мелкие события исчезают или, если возможно, группируются в более крупные
4. Наведение на событие вызывает полное отображение названия, продолжительность события, даты начала и конца

Прототип разработанного интерфейса представлен на рисунке 1. На данный момент ведется программная реализация разработанного приложения.

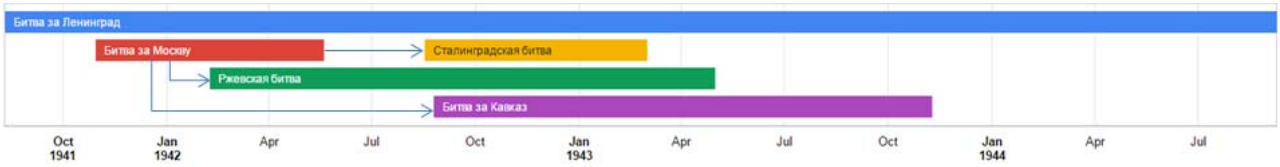


Рисунок 1 – Прототип интерфейса

Список литературы

1. Google chart documentation [Электронный ресурс]. – Режим доступа: https://developers.google.com/chart/interactive/docs/gallery/timeline
2. Vis.js documentation [Электронный ресурс]. – Режим доступа: http://visjs.org/docs/timeline/

АВТОМАТИЧЕСКИЙ АНАЛИЗ ЭМОЦИОНАЛЬНОГО СОСТОЯНИЯ АВТОРА В КОРОТКИХ ТЕКСТАХ НА ЕСТЕСТВЕННОМ ЯЗЫКЕ

Корней А.О. – студент, Крючкова Е.Н. – к.ф.-м.н., профессор
Алтайский государственный технический университет

В настоящее время наблюдается экспоненциальный рост объемов информации, доступной в сети Интернет. Социальные сети стали неотъемлемой частью повседневной жизни. Любое значимое событие мгновенно становится предметом обсуждений, а при выборе товара покупатели руководствуются опубликованными отзывами.

Доступ к такому количеству данных представляет собой огромный интерес как с точки зрения науки, так и с точки зрения бизнеса и политики. На сегодняшний день основное внимание сосредоточено на извлечении и анализе информации об эмоциях, мнениях и взглядах, выражаемых в текстах. Представители бизнеса могут получить необходимую для дальнейшего развития обратную связь. Представители власти, в свою очередь, имеют возможность анализировать реакцию населения на вводимые меры и принимать оптимальные решения.

Очевидно, что вручную извлекать информацию из онлайн-текстов практически невозможно. Поэтому существует особый класс методов называемый анализом тональности или сентимент-анализом. При анализе тональности применяются различные подходы, каждый из которых имеет свои сильные и слабые стороны. Выбор конкретного подхода целиком определяется поставленной задачей.

Извлечение мнений из большого количества текстов не является единственной задачей, для решения которой необходим анализ эмоциональной окраски. Анализ тональности может быть применен для определения каких-либо целевых эмоций в текстах (будь то депрессия или агрессия и нетерпимость). При создании алгоритмов взаимодействия человека и компьютера так же необходимо применять анализ тональности, и в данном случае требования к точности и адекватности применяемых методик куда более высоки.

Любая система анализа тональности текстов представляет собой модель для классификации эмоций и алгоритмы извлечения информации из текстов. Очевидно, что с ростом сложности системы растут и требования к отдельным ее компонентам.

1. Проблемы анализа тональности

При анализе любого текста, написанного на естественном языке, возникают сложности, обусловленные природой языка и особенностями его использования. Так, с точки зрения анализа тональности наиболее значимы следующие проблемы:

- Искажение тональности из-за сарказма [1]
- Влияние контекста на тональность
- Влияние отрицаний на тональность
- Утрата информации из-за ошибок разного уровня
- Утрата или искажение информации из-за использования смайлов
- Утрата или искажение информации из-за неверной трактовки местоимений
- Утрата или искажение информации из-за использования хештегов [2]

Каждая из перечисленных проблем решается в той или иной мере, однако полностью избавиться от ошибок, связанных с самой природой естественного языка, невозможно.

При работе над сложными задачами, такими как организация взаимодействия человека и компьютера, выявление сложных эмоциональных состояний, необходимо применять комплексный междисциплинарный подход. Для описания эмоциональных состояний необходима достаточно полная математическая модель, основанная на знаниях о психологии

эмоций. Алгоритмы извлечения информации должны сочетать в себе лексический, семантический анализ и элементы искусственного интеллекта (машинное обучение). Гибридный подход к извлечению тональности позволит минимизировать возможные ошибки, обусловленные природой естественных языков, а использование сложных моделей позволит выявлять целевые эмоции (настроения) в коротких текстах.

2. Математическая модель

Математическая модель, пригодная для решения сложных задач анализа тональности, должна достаточно полно и достоверно описывать эмоциональные состояния, присущие человеку и важные в контексте решаемой задачи. Рассмотрим основные модели, используемые для классификации эмоций в задачах анализа тональности.

Бинарная шкала. Оперирует простыми понятиями – «позитивный» и «негативный». Возможно использование третьего, «нейтрального» класса. Такая модель хорошо годится для статистического анализа отзывов о товаре. Описывается в ранних работах Терни[3] и Панга[4]. Возможны модификации модели с усложением шкалы и введением баллов, где низший балл соответствует негативу, а высший – позитиву [5,6].

Модель, основанная на понятиях объективности и субъективности. Использование понятия о субъективности позволяет отедлить части текста, которые несут в себе эмоциональный компонент, и сосредоточиться на их анализе. Такая модель успешно применяется в задачах, которые не связаны с маркетингом[7], однако ее слабость заключена в самом способе определения субъективности.

Модели, основанные на сложных эмоциональных состояниях. Психология эмоций такова, что понятий «позитивно» и «негативно» недостаточно, чтобы описать весь спектр эмоций, различаемых человеком. В задачах, где необходимо выйти за рамки бинарной шкалы, можно использовать более сложные модели. На рисунке 1 изображена визуализация двумерной модели (Russell, 1980)[8], и все эмоции описаны при помощи полярности и силы.

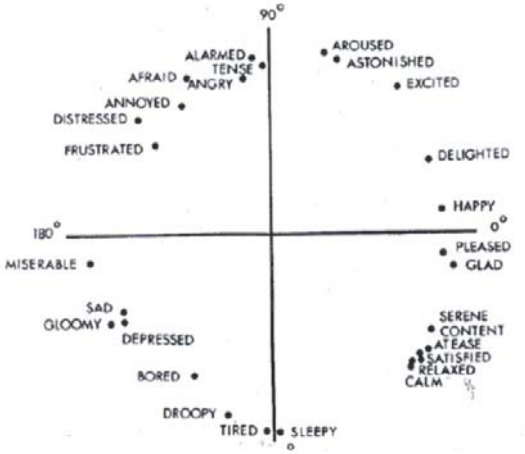


Рисунок 1 – двумерная модель эмоций Расселла (1980)

Следущим шагом в сторону усложнения модели можно назвать «Wheel of emotions» (Plutchik, 2001)[9] и производную модель «Hourglass of Emotions», описанную в работе[10]. Визуальное представление данных моделей изображено на рисунке 2. В основе обеих моделей 8 основных эмоций и 8 дополнительных, полученных попарным смешением основных. Достоинство обеих моделей в достаточно подробной структуре, однако существенным минусом является отсутствие специально определенного «нейтрального» класса.

Очевидно, для решения сложных задач необходима математическая модель, подобная Hourglass of Emotions. Однако, необходимо бороться с отсутствием нейтрального класса. Это возможно сделать несколькими способами:

- Особым образом интерпретируя слабовыраженные эмоции, используя адекватно подобранный порог интенсивности (считать, что текст нейтрален, если интенсивность эмоций в нем не превышает порог);
- Делая вывод о нейтральности текста на основе данных о его объективности (исходя из предположения, что оъективный текст не содержит никакой эмоциональной окраски).

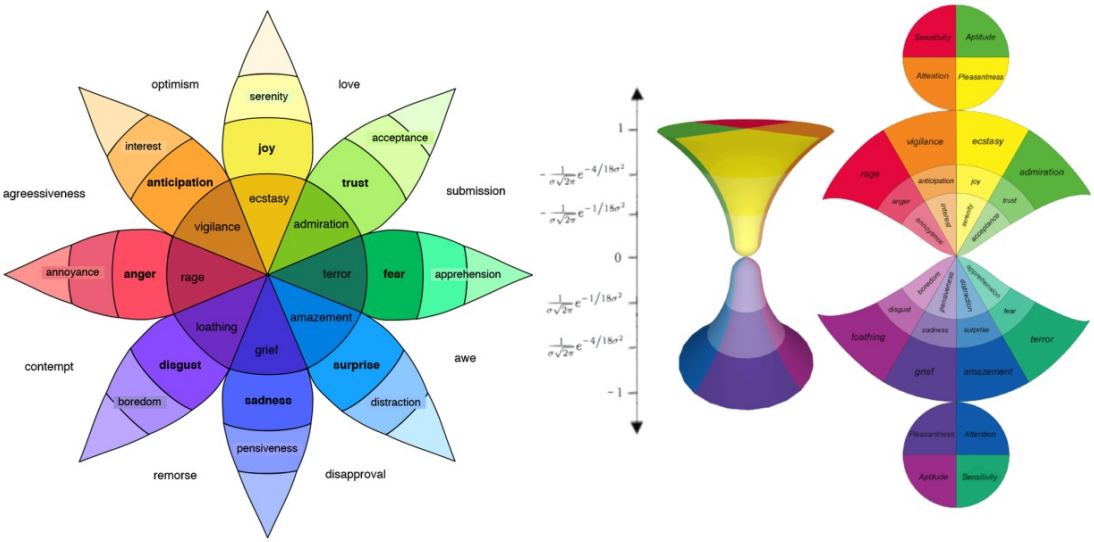


Рисунок 2 – модели Wheel of Emotions (слева) и Hourglass of Emotions (справа).

Выбор той или иной модели для описания эмоциональных состояний оказывает значительное влияние на перечень используемых методов.

3. Методы анализа тональности

Все методы, применяемые для анализа тональности, можно условно разделить на три группы:

- Методы, основанные на словарях
- Методы, основанные на машинном обучении
- Гибридные методы

Методы, основанные на словарях, используют лексикон, в котором каждому слову или понятию поставлена в соответствие некая метка эмоционального состояния. Итоговая эмоциональная окраска текста представляет собой совокупность тональностей отдельных лексических единиц. Основной проблемой данного подхода является зависимость тональности слова от контекста [11]. Построить универсальный словарь для всех предметных областей крайне сложно. Тем не менее, существуют достаточно хорошие решения – WordNetAffect, SentiWordNet, SenticNet. В основе последней системы лежит модель Hourglass of Emotions, описанная ранее.

Методы на основе машинного обучения требуют формирования тренировочного набора данных, размеченного в ручную или автоматически. Возможно обучение как с учителем, так и без. Обучение с учителем возможно лишь в том случае, если классификация эмоциональных состояний определена заранее и известно число классов. В противном случае необходимо обучение без учителя.

В последнее время наибольший интерес представляет применение нейронных сетей различной структуры для решения задач анализа тональности. В работе [12] показана эффективность рекурсивных нейросетей, а в работе [13] применены конволюционные нейронные сети. Алгоритмы, основанные на нейронных сетях, оказываются достаточно эффективными. Однако построение гибридной системы позволит использовать плюсы обоих подходов сразу и извлекать максимум информации о тональности текста. В качестве лексикона предложена семантическая сеть с аффективными метками для каждого понятия. Связи, существующие между отдельными понятиями семантической сети, несут в себе дополнительную информацию, которая может помочь при уточнении тональности текста или его фрагмента. Однако, это не решит всех проблем, связанных с определением тональности.

Заключение

Для решения задач, связанных с выявлением сложных эмоциональных состояний, необходимо построение гибридной системы, где в качестве лексикона используется семантическая сеть с аффективными метками, а машинное обучение выполняется с применением современных нейронных сетей. Особое внимание следует уделить минимизации ошибок, связанных с природой естественного языка и особенностями интернет-общения (употребление смайлов и хэштегов, пренебрежение правилами орфографии и пунктуации).

Список литературы

1. Riloff E., Sarcasm as Contrast between a Positive Sentiment and Negative Situation. Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pp. 704–714.
2. Тутубалина Е.В., Тестирование методов анализа тональности, основанных на словарях. Russian Digital Libraries Journal. 2015. V. 18. No 3-4
3. Turney, P. (2002). Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. Proceedings of the Association for Computational Linguistics. pp. 417–424.
4. Pang, Bo; Lee, L.; Vaithyanathan, S. (2002). Thumbs up? Sentiment Classification using Machine Learning Techniques. Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 79–86.
5. Pang, Bo; Lee, Lillian (2005). Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. Proceedings of the Association for Computational Linguistics (ACL). pp. 115–124.
6. Snyder, B., Barzilay, R. (2007). Multiple Aspect Ranking using the Good Grief Algorithm. Proceedings of the Joint Human Language Technology/North American Chapter of the ACL Conference (HLT-NAACL). pp. 300–307.
7. Denecke, K., & Nejdl, W. (2009). How valuable is medical social media data? Content analysis of the medical web. *Information Sciences*, 179(12), 1870-1880
8. Russell, James (1980). A circumplex model of affect. *Journal of Personality and Social Psychology*. 39: 1161–1178.
9. Plutchik, R. The nature of Emotions. *American Scientist*, vol. 89, Issue 4, p.344
10. Cambria E., Livingstone A., Hussain A., The hourglass of emotions. Proceedings of the 2011 international conference on Cognitive Behavioural Systems, February 21-26, 2011, Dresden, Germany
11. D'Andrea A., Ferri F., Grifoni P., Guzzo T., Approaches, Tools and Applications for Sentiment Analysis Implementation. *International Journal of Computer Applications* (0975 – 8887) Volume 125 – No.3, September 2015, pp 26-33

12. Tarasov D.S., Deep Recurrent Neural Networks for Multiple Language Aspect-based Sentiment Analysis of User Reviews. Computational Linguistics and Intellectual Technologies Papers from the Annual International Conference “Dialogue” (2015) Issue 14 Volume 2 of 2, pp 53-64
13. Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014), 1746–1751.

ОСОБЕННОСТИ РАЗРАБОТКИ СЕРВЕРНОГО ПО ДЛЯ СИСТЕМЫ НАВИГАЦИОННОГО МОНИТОРИНГА ГРУЗОВЫХ ПЕРЕВОЗОК

Метелкин К.О. – студент, Астахова А.В. – к.э.н., профессор
Алтайский государственный технический университет (Барнаул)

Многие предприятия, производственно-экономическая деятельность которых связана с транспортными средствами, сталкиваются с проблемами контроля перевозок и расхода топлива. Эти проблемы ведут к финансовым потерям, что отражается на результатах финансово-экономической деятельности предприятия в целом.

Для решения названных проблем существуют приемлемые по стоимости средства, разрабатываемые на основе спутниковых навигационных систем, которые позволяют в любой момент времени определить местонахождение конкретного автомобиля без временных затрат руководителей предприятия.

Мониторинг транспорта в режиме онлайн, дает уникальную возможность всегда иметь точную и достоверную информацию о реальном местоположении и маршрутах движения транспорта. Появляется возможность сверить маршрутные листы с реальным маршрутом, отображаемым на географической карте, с отчетом, на котором перечислены точки маршрута, либо с полным списком пройденных адресов. Можно легко сделать выводы о нецелевом использовании транспортных средств, принадлежащих компании (доставка "левых" грузов, отклонение от маршрутов, использование служебного транспорта в личных целях), или о хищениях и повреждении груза, топлива.

В связи с вышесказанным было принято решение разработать клиент-серверное приложение мониторинга транспортных средств на основе ГЛОНАСС для условий предприятия «Вкусная жизнь». Работа выполняется по заявке названного предприятия.

Основные функции, реализуемые программным обеспечением:

- повышение эффективности контроля грузовых перевозок;
- автоматизация отчетности по грузовым перевозкам;
- контроль и учет расхода горюче-смазочных материалов (ГСМ);
- контроль использования рабочего времени водителями во время грузовых перевозок.

При реализации данного проекта потребовалось решить следующие задачи:

- выполнить обзор и анализ используемых навигационных систем;
- определить параметры, получаемые с устройства, установленного на автомобиль, необходимые для мониторинга;
- разработать архитектуру системы ПО мониторинга транспортных средств;
- спроектировать и разработать базу данных;
- разработать серверную часть системы;

- спроектировать интерфейс для клиентской части системы;
- разработать клиентскую часть системы;
- согласовать проект с заказчиком, внести изменения в структуру данных и программное обеспечение, в случае необходимости;
- провести развертывание разработанного ПО и опытную эксплуатацию проекта на подмножестве реальных данных;
- оформить руководство пользователя;
- передать ПО и документацию по проекту заказчику.

Ниже подробнее остановимся на программном обеспечении серверной части системы, место которого видно из схемы рис. 1. На вход данного ПО поступает закодированная информация в виде сигналов с устройств мониторинга, установленных на автомобилях; информация регистрируется, декодируется, фильтруется и записывается в виде учетных показателей в базу данных для решения прикладных задач.

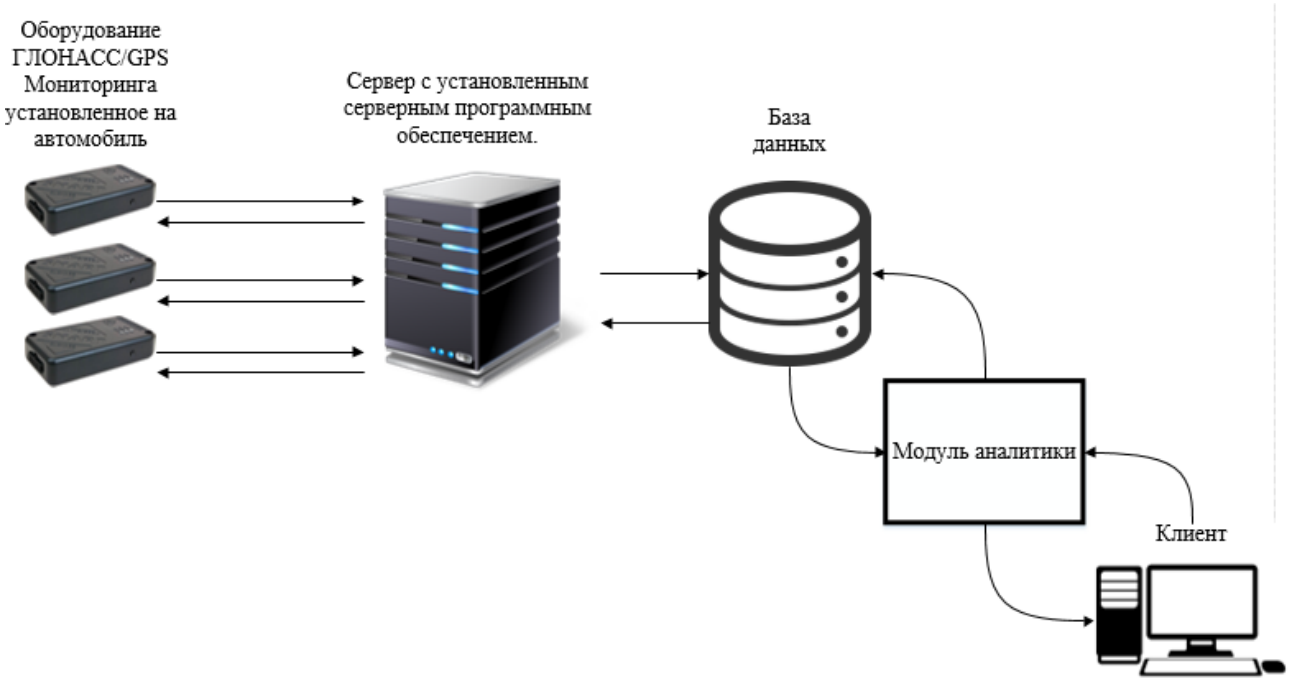


Рисунок 1 – Архитектура системы мониторинга

Для решения поставленной задачи требуется в заданном режиме получать с оборудования ГЛОНАСС/GPS [1] (устройства мониторинга) следующие параметры, регистрируемые на сервере системы:

- код события (событие – факт полученного сообщения);
- время события (время полученного сообщения);
- статус устройства (режим работы устройства);
- уровень GSM (уровень сигнала сотовой сети);
- состояние GPS (включен/выключен);
- время получения последних валидных координат до произошедшего события;
- угол широты в десятичных долях минуты;
- угол долготы в десятичных долях минуты;
- высоту относительно уровня моря в дециметрах;
- скорость, зафиксированную при получении последних валидных координат, в км/ч;
- курс, зафиксированный при получении последних валидных координат;
- пробег, зафиксированный на момент события, во время получения последних валидных координат;

- последний отрезок пути (пробег, рассчитанный между данным событием и предыдущим, то есть между двумя точками трека);
- моточасы (время работы генератора двигателя автомобиля).

Функции, реализуемые разработанным серверным программным обеспечением:

- реализация ожидания подключения устройств к определенному порту;
- контроль настроек зарегистрированного оборудования;
- получение данных с устройства мониторинга и проверка их целостности;
- преобразование полученных данных в пригодные для хранения в базе данных и отправка их в базу данных;
- формирование и отправка ответа на полученные сообщения от устройств мониторинга;
- ведение логов подключения и отключения устройств от сервера;
- реализация единовременного подключения нескольких устройств к серверу.

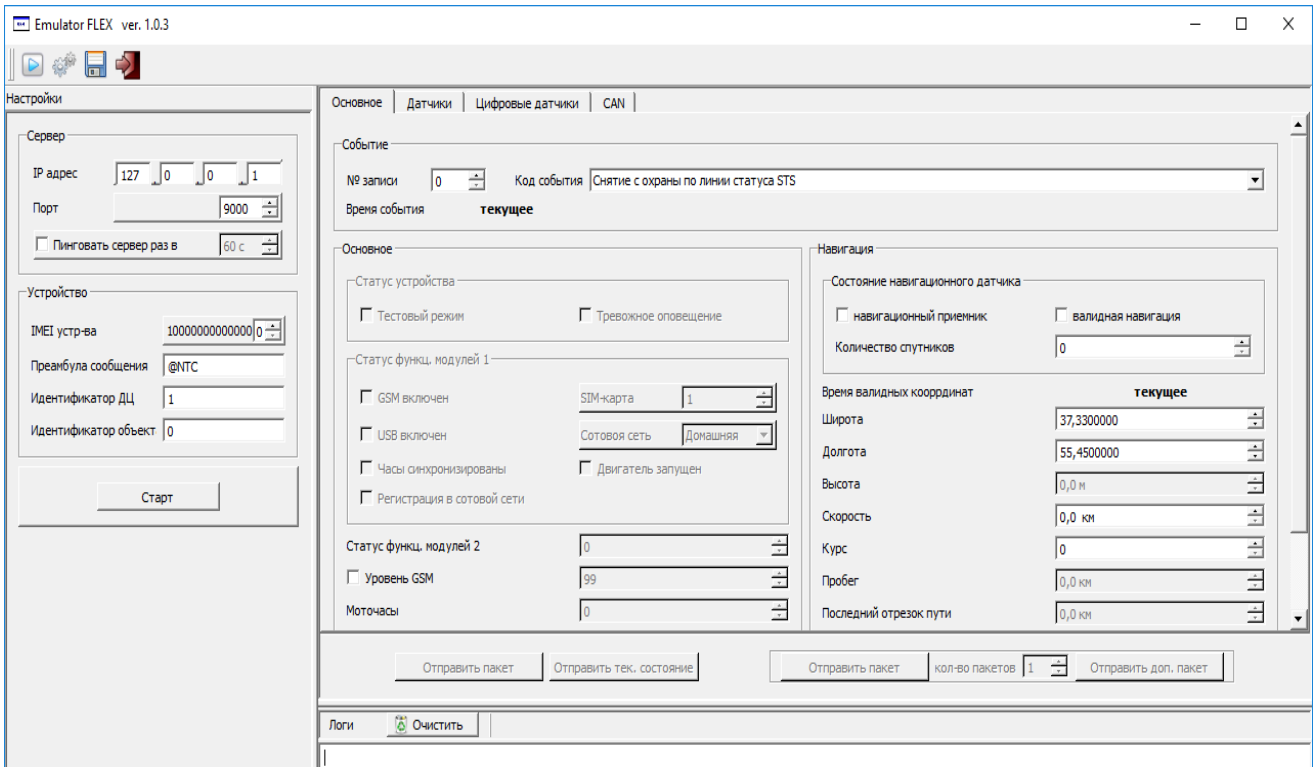


Рисунок 2 – Экранная форма эмулятора телематического устройства

Разработка серверной части системы осуществлялась с использованием эмулятора телематического устройства NTC FlexEmulator [2] (см. рис.2), который имеет графический интерфейс для настройки и консоль вывода, в которой отслеживаются отправленные на сервер данные и получение от сервера ответы. Это существенно упростило отладку и ускорило разработку серверного программного обеспечения.

При реализации алгоритмов обработки информации с устройства мониторинга потребовала решения проблема декодирования информации. Данные, поступающие с устройства это просто массив байтов, в котором хранятся все данные, в том числе, – выбранные нами параметры для поставленной задачи, – для хранения используются различные коды (UTF-8, UTF-16, UTF-32), что следует учесть при декодировании информации.

Программное обеспечение разработано на языке программирования C++ в среде разработки Qt Creator с использованием кроссплатформенного инструментария разработки

программного обеспечения Qt [3], предоставляющийся по лицензии GNU Lesser General Public License (LGPL).

Список литературы

1. Протокол информационного обмена оборудования ООО «Навтелеком» [Электронный ресурс]. – http://www.navtelecom.ru/tehpodderjka/razrabotchikam – Съем информации 04.03.2017 г.
2. NTC FlexEmulator – эмулятор телематического устройства – [Электронные данные]. – www.navtelecom.ru/programmnoe-obespechenie/ntc-flex-emulator-emulyator-telematicheskogo-ustrojstva – Съем информации 04.03.2017 г.
3. Qt Documentation [Электронный ресурс]. – Режим доступа: http://doc.qt.io/qt-5/qbytearray-members.html – Съем информации 04.03.2017 г.

РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ПО УЧЕТУ КАДРОВ

Коновальчук Е.С., Козликина Е.Ю., Копылова Д.А. – студенты
Астахова А.В. – к.э.н., профессор
Алтайский государственный технический университет (г. Барнаул)

В современных условиях развития информационно-коммуникационных технологий использование персонального компьютера, оснащенного автоматизированным рабочим местом (АРМ), использующим базу данных, является необходимым условием эффективной деятельности соответствующего специалиста. В этой связи актуальной является задача привития навыков практической работы с базами данных еще на этапе подготовки вузовских специалистов.

Одной из наиболее распространенных на практике информационных технологий является автоматизация кадрового делопроизводства. Современные автоматизированные системы управления персоналом призваны совершенствовать оперативную кадровую работу, в первую очередь, руководства и персонала организации. В частности, менеджеры по персоналу, используя такие системы, избавляются от множества рутинных операций по поиску информации при работе с персоналом. Автоматизированное хранение и обработка достаточно полной кадровой информации позволяет эффективно осуществлять подбор и перемещение сотрудников; своевременно контролировать сроки повышения квалификации, переизбрания профессорско-преподавательского состава; планировать и учитывать награды и поощрения и проч.

В настоящее время разработано достаточное количество программных продуктов, которые автоматизируют основные функции кадрового производства. Однако многие организации предпочитают иметь свою собственную индивидуально разработанную систему, которая отвечает всем требованиям конкретного производственного процесса и, как правило, является более дешевой, чем стандартные системы.

С учетом сказанного было принято решение разработать учебный проект автоматизации учета кадров для его использования студентами технического университета при изучении дисциплины «Информационные технологии». В качестве объекта исследования был выбран вуз (технический университет).

Для выделения автоматизированных процедур в отделе кадров университета был проведен анализ задач кадрового производства. Основными задачами являются следующие:

– осуществление эффективной кадровой политики руководства университета в целях обеспечения максимального качества и эффективности труда профессорско-преподавательского состава и иных категорий работников университета;

– комплектование структурных подразделений университета кадрами требуемых профессий, специальностей и соответствующей квалификации;
– планирование потребности в кадрах;
– оформление трудовых договоров (контрактов) с работниками университета, а также движения личного состава, ведение личных дел, персонального и статистического учёта профессорско-преподавательского состава и иных категорий работников университета;
– организация и координация кадровой работы в структурных подразделениях университета;
– управление дисциплинарными отношениями.

Результаты исследования и анализ документооборота, на основе которого реализуются названные выше задачи, позволил выделить оперативный учет как первоочередную задачу и соответствующий документ, содержащий учетную информацию о преподавателях и сотрудниках. Таким документом является личная карточка сотрудника. На основе личных карточек формируется картотека персонала. В зависимости от количества сотрудников личные карточки находятся в картотеке, либо в алфавитном порядке, либо по структурным подразделениям. В качестве бланка личной карточки используется унифицированная форма № Т-2, которая отражает всю необходимую информацию о личности и трудовой деятельности сотрудника.

Для проектирования базы данных, отображающей содержание формы № Т-2 и содержащей учетную информацию о сотрудниках, предварительно были спроектированы справочники нормативно-справочной информации (НСИ).

В разрабатываемом проекте выделены следующие основные справочники нормативно-справочной информации: пол, гражданство, языки, образование, состояние в браке, родство, квалификация по документу об образование, справочники по воинским сведениям, справочник должностей, образовательные учреждения, структурные подразделения, награды, отпуска, вид паспорта.

На рисунке 1 приведен фрагмент спроектированной авторами данных тезисов базы данных (БД), предназначенной для автоматизации работ кадрового производства, таких, как заполнение личной карточки работника (форма Т-2), в том числе, – ведение информации о воинском учете, приеме на работу сотрудника, его переводе и увольнении, аттестации, повышении квалификации, профессиональной переподготовке, наградах и отпусках.

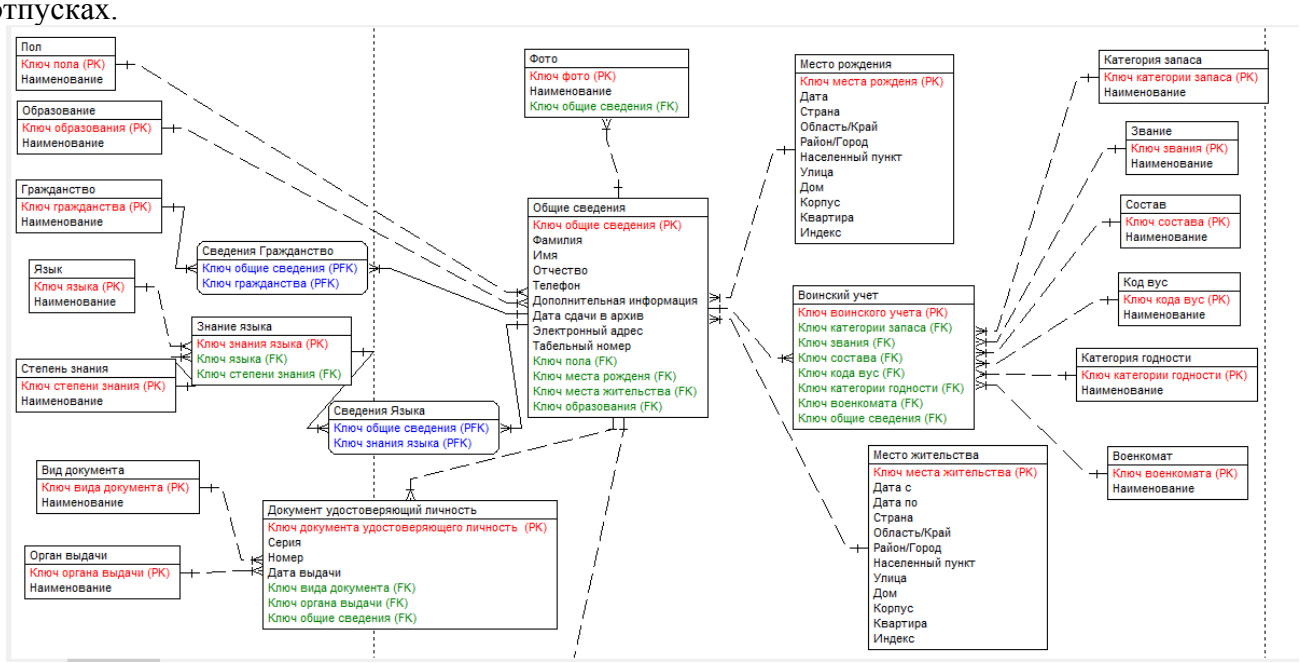


Рисунок 1 – Фрагмент полной атрибутивной модели данных

На основе данной БД в настоящее время авторами данных тезисов разрабатывается АРМ инспектора отдела кадров. Разрабатываемый проект, как было сказано выше, нацелен на его использование в учебном процессе вуза.

Студенты во время лабораторных работ, используя программное обеспечение (ПО) на основе разработанной базы данных смогут, в частности, выполнить такие работы как:

- формирование справочников НСИ;
- ведение личных карточек работников на основе примеров первичных документов с учетом всех разделов карточки (создание новой карточки; редактирование записей в личных карточках работников; удаление личных карточек работников);
- поиск личных карточек работников по различным признакам.

Программное приложение разрабатывается на языке программирования C#, Microsoft Visual Studio 2013 Community с использованием технологии Windows Form. В качестве СУБД выбрана система Oracle. Программное обеспечение реализует архитектуру «Клиент-сервер». Все используемые программы – лицензионные. Развертывание ПО и опытная эксплуатация проекта планируется с начала 2017-2018 учебного года. Экспериментальная проверка проектных решений запланирована в конце текущего учебного года.

НЕКОТОРЫЕ ВОПРОСЫ УПРАВЛЕНИЯ IT-ПРОЕКТАМИ

Метелкин К.О., Шкирков А.Ю., Васильева О.В.– студенты
Астахова А.В. – к.э.н., профессор
Алтайский государственный технический университет (г. Барнаул)

Управление IT-проектами невозможно назвать лёгким, его нельзя подогнать под шаблоны и рамки, поскольку информационные технологии пребывают в процессе постоянного развития. В этой связи современные IT-проекты даже средней и малой величины, разрабатываемые для иерархических систем управления предприятиями и организациями, отличаются значительной трудоемкостью и стоимостью. Так в [1, с. 81] приведены следующие характеристики таких проектов: «от трех до четырех уровней; от 15 до 40 пакетов работ; от 40 до 80 часов на средний пакет работ; от 3 до 7% общего бюджета рабочих часов на средний пакет работ». Трудности разработки и внедрения IT-проектов обусловлены многими причинами, в том числе:

- проблемами в координации управления большим количеством взаимосвязанных работ по проекту, в том числе, – выполняемых параллельно;
- нерациональным использованием системных ресурсов на этих работах;
- недостаточно высокой квалификацией некоторых исполнителей работ;
- ошибками на ранних стадиях разработки проектов.

Попытки ускорить сроки разработки и уложиться в бюджет могут привести к некачественным проектным решениям, базирующимся на:

- размытости представлений у заказчика задач проектирования: отсутствии ясных целей и четких требований,
- несогласованности IT-стратегии с корпоративной стратегией,
- не достаточной полноте обследования предметной области системными аналитиками и IT-специалистами – участниками проекта.

В результате увеличиваются риски проектов. Так, согласно статистике по международного агентства ИТ-услуг The Standish Group CHAOS Summary 2009, «32%

проектов завершились успешно; 44% испытали различные трудности (превысили бюджет, выпали из сроков и прочее); 24% проектов просто провалились; во всех завершенных проектах только 61% требуемых свойств были реализованы» [2, с. 16].

Сказанное выше обусловливает актуальность управления IT-проектами, и в первую очередь, – актуальность планирования, как основополагающей функции управления. Следует отметить, что данная компетенция должна осваиваться еще на этапе вузовской подготовки специалистов. Четкое календарное планирование с закреплением за этапами плана исполнителей с расчетом трудоемкости работ и ресурсов, необходимых для выполнения работ на всех фазах жизненного цикла IT-проекта, является необходимым условием его успешности. В этой связи подготовка IT-специалистов требует овладение студентами навыков разработки перспективных или текущих планов и оперативных планов-графиков выполнения работ, связанных с проектированием и внедрением IT-проектов.

Управление проектами – это искусство и наука организации, планирования и управления различными процессами, обладающими, как правило, индивидуальными особенностями, в условиях ограниченных ресурсов, времени и затрат. Для облегчения процесса управления проектами разработаны и продолжают разрабатываться и модернизироваться множество самых разнообразных методов. Одним из математических методов в управлении проектами является сетевое планирование и управление (СПУ).

Методика СПУ – это развитая система планирования и управления разработками, предусматривающая как отражение логических связей между отдельными работами, так и оперативную корректировку плана проектных работ, а также возможность прогнозирования и предупреждения возможных срывов в ходе выполнения проекта.

Основная цель СПУ - сокращение до минимума продолжительности проекта, а иногда – сокращение затрат, связанных с выполнением работ по проекту. Метод СПУ позволяет графически, наглядно и системно отобразить и оптимизировать последовательность и взаимозависимость работ, действий или мероприятий, обеспечивающих своевременное и планомерное достижение конечных целей.

Сетевое планирование и управление включает в себя множество методов и моделей. В качестве наиболее распространенных методов СПУ можно назвать:

- Метод CPM (Critical Path Method) – метод критического пути.
- Метод PERT (Program Evaluation and Review Technique) – метод оценки и пересмотра планов.
- Метод MPM (Metra Potential Method) – метод потенциальных величин.
- Метод GERT (Graphical Evaluation and Review Technique) – метод графической оценки и анализа.

Любая сетевая модель, используемая в рамках СПУ состоят из трех следующих элементов:

работа – это процесс, который необходимо выполнить для получения определенного (заданного) результата, как правило, позволяющего приступить к последующим действиям;

событие – результат выполнения работы или дата в ходе осуществления проекта; событие используется для отображения состояния завершенности тех или иных работ;

связь – это логическая зависимость между сроками выполнения отдельных работ и наступления событий; если для начала выполнения какой-либо работы необходимо завершение другой работы, говорят, что эти работы соединены связью (связаны).

По способу представления информации существуют два принципиально различных вида сетевых моделей и соответствующих сетевых графиков (СГ).

Первая. Сеть вида "вершина – событие": вершины соответствуют событиям, а соединяющие их дуги – работам. Связи представлены пунктирными стрелками, которые так же, как и работы, являются направленными дугами графа.

Вторая. Сеть вида "вершина – работа": вершины соответствуют работам, а дуги – связям. События (главным образом вехи) при необходимости отображаются какими-либо фигурами, например – треугольниками.

Практика использования метода СПУ показывает, что для управления IT-проектами чаще всего используется метод сетевого планирования и управления PERT-CPM.

Проект представляется в виде списка работ (отдельных операций), между которыми установлены связи различных типов:

- связь типа «начало – окончание»: следующая работа не может быть начата, если не будет завершена предыдущая;
- связь типа «один ко многим»: завершение одной работы связано с началом нескольких других работ. Таким образом, проект представляется в виде сетевого графика работ, где работы – это вершины графа, а связи между ними – направленные дуги графа, имеющие «веса», соответствующие вектору ресурсов, в том числе, – по времени.

Критическим путём называется такая цепочка связанных работ от начального до конечного события СГ, для которой характерна наибольшая длительность выполнения.

Длительность критического пути определяет минимальное время выполнения IT-проекта. При этом длительность выполнения работ, лежащих вне критического пути можно увеличивать или уменьшать, и это не скажется на общей продолжительности осуществления IT-проекта. При сокращении сроков работ, лежащих на критическом пути возможна ситуация, когда критическим путём может стать другая цепочка работ.

Для выполнения проекта задействуются ресурсы различного вида. В их число входят материальные ресурсы, рабочее время специалистов, оборудование, финансовые средства. Единица любого ресурса имеет свою стоимость, а в совокупности все эти затраты определяют бюджет проекта.

Авторами данных тезисов разработан план работ; сформирован и реализован в MS Project соответствующий ему график выполнения работ (в виде сетевого графика и в форме диаграммы Ганнта) для IT-проекта «Информационная системы обслуживания клиентов в отделении банка».

Использование метода СПУ в рамках MS Project позволяет решить следующие основные задачи планирования и оптимизации:

- сокращение до минимума продолжительности IT-проекта;
- сокращение до минимума материальных затрат IT-проекта;
- построение диаграмм использования ресурсов;
- логическое отображение комплекса работ;
- прогнозирование возможных срывов в ходе выполнения IT-проекта;
- анализ отклонений расчетных показателей от запланированных.

Сетевая модель представляет собой план выполнения заданного комплекса взаимосвязанных работ и изображается с помощью графа [3]. Вершины графа соответствуют событиям (моменты начала или окончания работ), ориентированные рёбра – работам. Над каждым ребром удобно указывать продолжительность работы (можно также – требуемые для работы ресурсы).

Проанализировав сетевой график по критерию времени, устанавливают общую продолжительность всего комплекса работ –длину критического пути, которая определяется работами, лежащими на критическом пути. Критический путь – это цепочка работ, у которой наибольшая длительность. К задержке завершения комплекса работ ведёт задержка любой критической работы, поэтому в первую очередь именно критические работы нужно обеспечить всеми необходимыми ресурсами, в том числе, – финансами. Логистический анализ в СПУ – в том, чтобы выявить возможности сокращения времени некритических работ. Для этого используются модели оптимизации. Длительность выполнения работ, лежащих вне критического пути можно увеличивать или уменьшать, и это не скажется на

общей продолжительности осуществления IT-проекта. При сокращении сроков работ, лежащих на критическом пути возможна ситуация, когда критическим путём может стать другая цепочка работ.

Существует несколько методов оптимизации сетевого графика.

Один из них, рассмотренный ниже на конкретном проекте авторов данных тезисов, заключается в уменьшении стоимости комплекса запланированных работ за счёт увеличения продолжительностей выполнения некритических работ сетевого графика, имеющих свободный резерв времени.

Рассмотрим для примера 17 работ, относящихся к предпроектной стадии и частично к стадии разработки техно-рабочего проекта информационной системы (ИС) обслуживания клиентов в отделении банка (см. рис.1, на котором жирной ломаной линией выделен найденный критический путь).

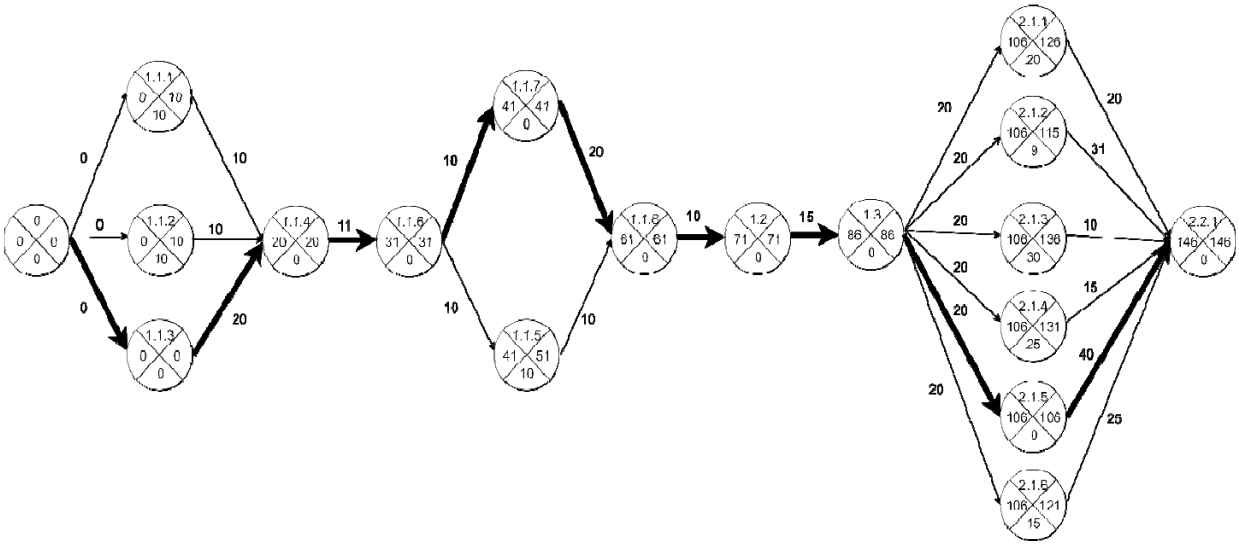


Рисунок 1– Фрагмент сетевого графика

Здесь представлены следующие работы:

- 0 – точка начала фрагмента проекта;
- 1.1.1 – 1.1.3 – анализ задач системы обслуживания клиентов и путей автоматизации с учётом выявленных проблемных ситуаций;
- 1.1.4 – анализ списка услуг банка и содержания услуг;
- 1.1.5 – разработка требований к алгоритмам задач ИС;
- 1.1.6 – 1.1.7 – разработка требований к кодированию информации, к составу и структуре данных;
- 1.1.8 – корректировка требований с руководителем отделения банка;
- 1.2 – технико-экономическое обоснование целесообразности разработки;
- 1.3 – разработка технического задания на проектирование ИС;
- 2.1.1 – 2.1.8 – проектирование обеспечивающих и функциональных подсистем ИС;
- 2.2.1 – точка окончания фрагмента проекта.

Значения параметров сетевой модели для некритических работ приведены в таблице 1. Использованы обозначения:

A_{ij}, T_{ij}, B_{ij} – начальная продолжительность работы (*i,j*) в сутках (минимальная, наиболее вероятная и максимальная соответственно);

$R^{ij}_{св}$ – свободный резерв времени работы в сутках;

C_{ij}– стоимость работы в тыс. руб;

(C^{ij}_{max}- C^{ij}_{min}) – интервал изменения стоимости работы в тыс. руб;

h_{ij} – коэффициент затрат на ускорение сроков выполнения работы в тыс. руб./сутки;

T_{new} – отредактированная продолжительность работы в сутках;

ΔC_{ij} – уменьшение стоимости проекта в тыс. руб;

$$h_{ij} = \frac{C_{max}^{ij} - C_{min}^{ij}}{B_{ij} - A_{ij}}$$

$$\Delta C_{ij} = \left(B_{ij} - T_{new}\right) * h_{ij}$$

Таблица 1 – Значения параметров сетевой модели для некритических работ

Название	**A_{ij}**	**T_{ij}**	**B_{ij}**	**R_{ij}**	**C_{ij}**	**C^{ij}_{max} – C^{ij}_{min}**	**H_{ij}**	**T_{new}**	**ΔC_{ij}**
1.1.1.	5	10	20	10	13600	20400	1360	15	6800
1.1.2.	5	10	15	10	13600	13600	1360	13	2720
1.1.5.	5	10	15	10	21600	21600	2160	14	2160
2.1.1.	15	20	24	20	48000	21600	2400	22	4800
2.1.2.	25	31	40	9	111600	54000	3600	37	10800
2.1.3.	8	10	20	30	44400	53280	4440	17	13320
2.1.4.	10	15	22	25	64000	51200	4267	20	8533
2.1.6.	20	25	30	15	40000	16000	1600	29	1600
C =					**356800**			**ΔC =**	**50733**

Для критических работ таблица их стоимости приведены в табл.2.

Таблица 2 – стоимости критических работ

Название	1.1.3	1.1.4	1.1.6	1.1.7	1.1.8	1.2	1.3	2.1.5	Сумма
Cij	27200	36960	29600	59200	57600	50400	99200	16000	376160

Стоимость первоначального варианта плана работ равна сумме стоимостей всех работ (включая работы, не имеющие резервов и не включенные в таблицу):

C = 356 800 + 376 160 = 732 960 рублей

Стоимость нового плана равна:

C – ΔC = 732 960 – 50 733 = 682 227 рублей

В результате оптимизации сетевого графика работ получен новый график, позволяющий выполнить комплекс работ в запланированный срок

T' = T = 146 дней при стоимости C = 682 227 руб.

В итоге без изменения продолжительности критического пути нами получено сокращение затрат времени на выполнение работ проекта на 7%.

Следует отметить, что продукт MS Project, поддерживающий совместное управление длительными проектами и ресурсами организации, в редакции Project 2016 позволяет реализовать алгоритм, описанный выше, а также планировать даты проектов и их задач до 31.12.2149 г., что подтверждает целесообразность использования программы Project для практических целей.

Список литературы

1. Грекул В.И. Проектное управление в сфере информационных технологий. [Электронный ресурс] — Электрон. дан. — М. : Издательство "Лаборатория знаний", 2013. — 336 с. — Режим доступа: http://e.lanbook.com/book/8809 − Загл. с экрана (съем информации 01.01.2017 г.)
2. Авдошин С.М. Информатизация бизнеса. Управление рисками. [Электронный ресурс] : Учебник / С.М. Авдошин, Е.Ю. Песоцкая. — Электрон. дан. — М. : ДМК Пресс, 2011. — 176 с. — Режим доступа: http://e.lanbook.com/book/3028 − Загл. с экрана (съем информации 01.01.2017 г.)
4. Исследование операций в экономике : Учебн. пособие для вузов / Н.Ш. Кремер, Б.А. Прутко, И.М. Тришин, М.Н. Фридман/ под ред. проф. Н.Ш. Кремера. − М.: Банки и биржи, ЮНИТИ, 1997. − С. 286-332.

РЕАЛИЗАЦИЯ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ АНАЛИЗА ПЛАГИАТА ПРОГРАММНОГО КОДА

Савченко И.В. – студент, Крючкова Е.Н. – к.ф.-м.н., профессор

Алтайский государственный технический университет (г. Барнаул)

Актуальность работы

Возможность легкого копирования информации, представленной в электронном виде, породила множество проблем, связанных с нарушением авторских прав. Как правило единственным способом улаживания конфликтов являются судебные разбирательства.

Статья 146 УК РФ. Нарушение авторских и смежных прав (в ред. Федерального закона от 08.04.2003 N 45-ФЗ) говорит, о том что при нарушении авторских прав предусматриваются наказания, максимальные из которых: до 200 тысяч рублей штраф и до шести месяцев ареста[1].

На сегодняшний день существуют системы, проверяющие текст на плагиат, в том числе и системы, анализирующие исходные тексты программ [2]. Однако методы, которые используют такие системы, не всегда срабатывают. Поэтому в данной работе предлагаются методы, направленные на ликвидацию существующих недостатков.

Анализ характеристик программы

Любая программа имеет определенные составы структур, которые могут быть выявлены, измерены, а затем использованы в качестве характеристик программы. Такие характеристики должны незначительно меняться в случае модификации исходного кода. Имена процедур или переменных, текстовые строки и тому подобное могут быть легко изменены злоумышленником и поэтому не должны использоваться для полного сравнения. Так же легко могут быть модифицированы и операторы в программе. Однако зачастую их количественные характеристики изменяется не существенно.

Характеристиками программы в данном случае выступают метрики программного обеспечения (ПО). Существуют различные типы метрик, такие как: количественные метрики, метрики цикломатической сложности, объектно-ориентированные метрики и т.д. [3,4]. Следовательно, чем больше метрик будет использовано, тем, вероятно, более точный результат будет получен.

При маскировке нелегально заимствованного исходного кода программы злоумышленник может изменить некоторые операторы или добавить новые, но в целом изменения, вероятно, будут настолько малы, что распределение частот останется практически тем же. Конечно, близкие значения частот во фрагментах различных программ еще не являются доказательством факта плагиата, но зато дают повод это подозревать.

Кроме того, предлагается анализировать не только метрические показатели программы, но и размещение операторов в теле программы. Необходимо сравнить две последовательности операторов, чтобы получить взаимную корреляцию операторов этих последовательностей. Следует отметить, что необходимо анализировать не полное совпадение последовательностей, а их схожести, так как злоумышленник может незначительно видоизменить последовательность операторов.

Кроме закономерной корреляции может возникнуть и заметная случайная, поэтому каждый вариант высокого значения корреляции следует анализировать другими методами.

Как уже отмечалось, близкие значения характеристик двух различных программ не означают, что в одной программе присутствуют фрагменты другой. Данное обстоятельство является недостатком описанных методов.

Статический и динамический анализ программного обеспечения

Методы статического анализа программного кода предполагают проведение анализа без запуска программы на исполнение с помощью автоматического построения модели программы и последующего анализа построенной модели. Модель программы предлагается построить таким образом, который позволит проводить частичный или параллельный анализ. Такой подход позволяет существенно сократить время анализа программы. Как правило, модель программы имеет некоторую степень приближения к реальному поведению программы в процессе запуска. В связи с этим поиск ошибок с помощью статического анализа может иметь как ложные срабатывания, так и не обнаруживать некоторые ошибки, имеющиеся в программе. Статический анализ используется для вычисления характеристик программы. При статическом анализе характеристики могут отличаться несущественно, поэтому в данной работе предлагается дополнительно использовать динамический анализ.

Динамический анализ [5,6] основан на запуске исследуемого продукта на исполнение, что позволит выявить дополнительные характеристики программы. Основной минус динамического анализа состоит в том, что для получения качественного покрытия анализируемой программы, как правило, требуется неоднократный запуск программы на исполнение, что связано с немалыми временными затратами.

Так как динамический анализ требует больших временных затрат, то он будет использоваться в режиме тотальной проверки.

Структура системы

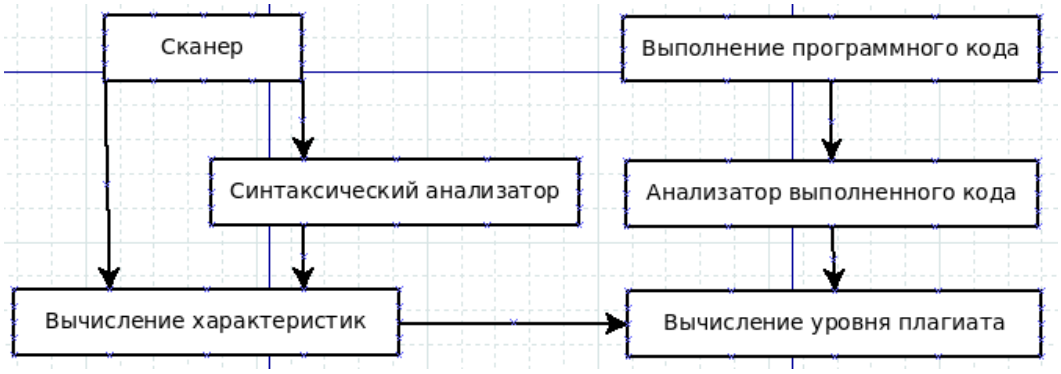


Рисунок 1 - “Структура системы”

На уровне статического анализа структурные характеристики программы вычисляются на основе анализа синтаксических конструкций. Для этого необходим лексический сканер, из которого будут получены комментарии и переданы в блок вычисления характеристик. В синтаксическом анализаторе выявляются синтаксические структуры программы, которые передаются в блок вычисления характеристик. Блок вычисления характеристик получает данные от сканера и синтаксического анализатора, и вычисляет необходимые метрики.

Динамический анализ проводится посредством внешней утилиты, которая выполняет исходный код. Затем данные полученные от внешней утилиты анализируются.

Теперь когда имеются характеристики полученные с помощью статического и динамического анализа вычисляется уровень плагиата.

Выводы

Предложенные методы выявления плагиата программного кода универсальны, и могут выявить схожести даже у текстов разных языков программирования. Необходимо лишь выбрать в системе по каким именно характеристикам программы проводить сравнение.

Реализация представленной системы должна лишь помочь в выявлении плагиата программного кода и не гарантирует 100% успеха.

Список литературы

1. КонсультантПлюс [Электронный ресурс]. – Режим доступа:http://www.consultant.ru/document/cons_doc_LAW_10699/b683408102681707f2702cff05f0a3025daab7ab/
2. Обзор алгоритмов обнаружения плагиата в исходных кодах программ [Электронный ресурс]. – Режим доступа:http://rain.ifmo.ru/cat/data/theory/unsorted/plagiarism-2006/article.pdf
3. Метрики кода и их практическая реализация [Электронный ресурс]. – Режим доступа:http://cmcons.com/articles/CC_CQ/dev_metrics/mertics_part_1/
4. Метрики программного обеспечения [Электронный ресурс]. – Режим доступа:https://www.viva64.com/ru/a/0045/
5. Использование статического и динамического анализа для повышения качества продукции и эффективности разработки [Электронный ресурс]. – Режим доступа:http://www.swd.ru/print.php3?pid=828
6. Динамический анализ кода [Электронный ресурс]. – Режим доступа:https://www.viva64.com/ru/t/0070/

ОПИСАНИЕ МЕХАНИЗМА ВСТРАИВАНИЯ ГОТОВОГО РЕШЕНИЯ В САЙТ КЛИЕНТА

Самойлов С.С. – студент, Крючкова Е.Н. – к.ф.-м.н., профессор
Алтайский государственный технический университет (г. Барнаул)

Разрабатывать системы под все нужды бизнеса дорого и долго. В условия жёсткой конкуренции побеждают те проекты, которые используют наработки других. Программисты часто используют внешние библиотеки вместо того, чтобы писать весь требуемый функционал самостоятельно. Разработчики сайтов используют web фреймворки и CMS, чтобы ускорить получение результата. Некоторые системы необходимо встраивать в сайт клиента, например, карту [1] или систему онлайн бронирования отелей.

Способ встраивания системы должен быть простым. Для встраивания в сайт клиента можно открыть необходимое API, однако реализация этого API может занять много

времени. Можно подготовить готовое решение так, чтобы клиенту осталось только вставить три строчки в код страницы своего сайта. После чего готовое решение будет полностью готово, а его функционал будет доступен через фрейм.

В коде страницы сайта клиента подключается скрипт готового решения [2]. Также в месте расположения фрейма в коде страницы ставится якорь в виде элемента с определённым id. После чего остаётся только показать скрипту готового решения, что вместо якоря нужно установить фрейм. Для этого нужно вызвать определённую функцию из загруженного скрипта готового решения. В параметры этой функции можно передать какие-либо настройки страницы в фрейме.

Страница готового решения никак не может напрямую влиять на сайт клиента, однако часто это необходимо. Разработчики браузеров запретили такое взаимодействие в целях безопасности. Но они разработали механизм, с помощью которого родительский сайт и сайт во фрейме могут обмениваться сообщениями. Отправлять такие сообщения можно с помощью метода postMessage [3]. Получать их можно подписавшись на событие onmessage.

В нашем случае приёмом, обработкой и отправкой сообщений будет заниматься скрипт готового решения. Если странице фрейма А нужно будет передать сообщение в страницу фрейма В, то скрипт готового решения будет выступать в качестве посредника. Схема этого взаимодействия показана на рисунке 1.

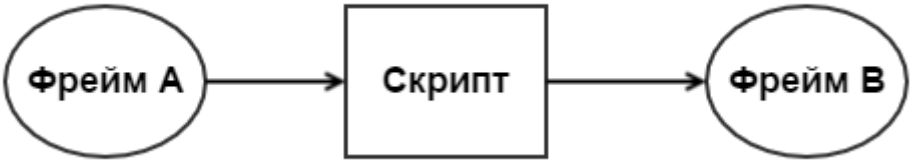


Рисунок 1 Схема взаимодействия фреймов

Кроме переотправки сообщений между фреймами скрипт готового решения может принимать другие сообщения. Далее будут описаны несколько возможных типов сообщений.

- Отправка сообщения всем фреймам. Кроме отправки сообщения конкретному фрейму может потребоваться рассылка сообщений всем фреймам системы.
- Изменение размера фрейма. Страница в фрейме может управлять только собственным размером. При этом сам фрейм не будет под него подстраиваться. Вместо этого появится пустое место на странице клиента или прокрутка. Однако фрейм может сообщать о своём размере скрипту готового решения, который в свою очередь будет подстраивать размер фрейма.
- Переход на другую страницу в фрейме. Если страница в фрейме самостоятельно перейдёт на другую страницу, то скрипт готового решения утратит над ней контроль. Поэтому нужно отправить сообщение скрипту готового решения, который поменяет страницу фрейма, заменив атрибут src в теге iframe.
- Переход на другую страницу сайта клиента. Фрейм разумеется не может напрямую поменять адрес родительской страницы. Но это можно сделать через сообщение.
- Открыть ещё один фрейм. Странице в фрейме может понадобиться открыть ещё один фрейм на сайте клиента, например, всплывающее окно. Это можно также организовать через посылку сообщений.
- Уничтожить фрейм. Если страница фрейма была вспомогательной, например, всплывающим окном, то после соответствующий фрейм необходимо уничтожить после окончания работы с этой страницей.
- Оправить событие. На странице в фрейме может произойти что-то такое, на что нужно отреагировать сайту клиента, например, пользователь авторизовался. Можно послать сообщение скрипту готового решения, чтобы тот вызвал необходимое событие.

Помимо создания фреймов и обработки сообщений от них скрипт готового решения может предоставлять полезные методы. Далее будут описаны несколько возможных дополнительных методов скрипта готового решения.

- Метод, в который вызывает callback после того, как полностью инициализируется скрипт готового решения.
- Метод, который вернёт информацию, например, данные авторизированного пользователя.
- Метод, который повесит определённый обработчик на определённую кнопку. С помощью такого метода можно предоставить разработчику сайта клиента использовать собственные кнопки для взаимодействия с готовым решением. Можно дать ему возможность создавать и уничтожать фреймы готового решения, залогинить и разлогинить пользователя готового решения.

Скрипт готового решения по возможности не должен использовать какие-либо внешние библиотеки. Даже использование JQuery, которая стала заменой стандартной библиотеки JavaScript, нежелательно. Поэтому приходится использовать стандартную библиотеку JavaScript.

Список литературы

1. API Яндекс.Карт [Электронный ресурс]. Режим доступа: https://tech.yandex.ru/maps/mapsapi/, свободный
2. Флэнаган Д. JavaScript. Подробное руководство. – Москва: Символ-Плюс, 2013. – 1080 с.
3. Общение окон с разных доменов: postMessage [Электронный ресурс]. Режим доступа: https://learn.javascript.ru/cross-window-messaging-with-postmessage, свободный.

РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ РАСЧЁТА МОРФОМЕТРИЧЕСКИХ И ГИДРОЛОГИЧЕСКИХ ХАРАКТЕРИСТИК ТЕРРИТОРИИ ДЛЯ ГИС С ОТКРЫТЫМ КОДОМ

Сидоренко Е.А. – студент, Ловцкая О.В. – с.н.с.*, Бубнова Н.Д. – ст. преподаватель
Алтайский государственный технический университет (г. Барнаул)
* Институт водных и экологических проблем СО РАН (г. Барнаул)

Проведение расчетов для неизученных рек и речных участков весьма актуально для нашей страны, в которой менее 3000 речных гидрологических постов на более чем 2,5 млн. рек. В настоящее время нет официальных специализированных программ для автоматизации гидрологических расчетов при отсутствии гидрометрических данных [1]. Некоторым исключением можно считать программы Snipcalc 2.0 для расчета морфометрических характеристик [2] и FloodHigh [3] для расчета максимального стока весеннего половодья и дождевых паводков неизученных рек.

Расчеты в условиях отсутствия данных наблюдений требуют привлечения уточненных морфометрических, гидрографических и ландшафтных характеристик речных водосборов. В недавнем прошлом эти картометрические работы выполнялись вручную в соответствии с рекомендациями и формулами, изложенными в [4, 5]. В последнее время произошел существенный технологический прорыв, связанный, во-первых, с появлением различных программных комплексов, реализующих ГИС-технологии, во-вторых, с появлением новых источников пространственных данных (данные дистанционного зондирования Земли; готовые ЦМР на все территории), большей открытостью топографических карт, обновлением и появлением новых отраслевых карт.

Современные проприетарные ГИС (ArcGIS® for Desktop) имеют инструменты, позволяющие получать гидрографические характеристики водосборов, однако их использование затруднено высокой стоимостью коммерческих программных продуктов и их функциональной избыточностью. В последние годы наметилась тенденция к использованию открытого программного обеспечения с настраиваемой функциональностью.

QuantumGIS (QGIS) – свободное, бесплатно распространяемое программное обеспечение для работы с географическими (пространственными) данными, которое позволяет просматривать, создавать, редактировать и анализировать пространственные данные в различных растровых и векторных форматах, а также использовать различные форматы баз данных. В настоящее время QGIS – это кроссплатформенное приложение, т.е. программа, которая может работать на разных операционных системах. Открытость исходных кодов QGIS, возможность обмениваться данными с другими ГИС и расширять функциональность с помощью плагинов делает весьма перспективным использование этой программы в образовательных, научных и прикладных целях [6]. Все вышеизложенное и определило выбор ГИС для выполнения данной работы.

Целью работы является разработка плагина для открытой ГИС QGIS, который позволяет определить морфометрические и гидрологические характеристики речного водосбора. Основные требования к функционалу плагина:

- определить границы, площадь, среднюю высоту, уклон речного водосбора;
- определить длину исследуемой реки и общую протяженность русловой сети, коэффициент густоты речной сети;
- определить площадь озер, болот, залесенных территорий.
- Исходными данными для плагина являются:
- растровый файл с привязанной топографической картой;
- набор тематических шейп-файлов, полученных при оцифровке топографических и соответствующих тематических карт;
- цифровая модель рельефа на исследуемый регион (далее – ЦМР).

Для расчета необходимых величин используются расчетные формулы, приведенные в [7].

Площадь водосбора – часть земной поверхности и толщи почвогрунтов, ограниченная водораздельной линией, с которой вода поступает в данный водный объект: $F = \sum Fi$, где $\sum Fi$ – сумма площадей всех измеренных водосборов.

Гидрографическая длина водотока – наибольшая протяженность основного русла водотока (системы водотоков), измеряемая от истока притока, составляющего с основным водотоком наибольшую длину. Вычисляется как длина русла водотока.

Средневзвешенный уклон водотока: $I = \prod Ij$, где $\prod Ij$ – произведение частных средних уклонов водосбора.

Относительная заболоченность – отношение суммарной площади болот на водосборе к общей площади водосбора: $fз = \sum Fб / F$, где $\sum Fб$ – суммарная площадь болот, F – общая площадь водосбора.

Относительная лесистость – отношение суммарной площади лесных угодий на водосборе к общей площади водосбора: $fл = \sum Fл / F$, где $\sum Fл$ – суммарная площадь лесов, F – общая площадь водосбора.

Озерность водосбора – отношение суммарной площади водной поверхности водоемов, расположенных в пределах водосбора, к общей площади водосбора: $fo = \sum Fo / F$, где $\sum Fo$ – суммарная площадь озер, F – общая площадь водосбора.

Коэффициент густоты речной сети: $Gp = \sum l / F$, где $\sum l$ – сумма длин всех водотоков, F – общая площадь водосбора [7].

Программная реализация плагина выполнена в виде встраиваемого модуля для ГИС QGIS, написанного на языке программирования общего назначения Python. Плагин представляет собой программно зависимую реализацию описанных функций для QGIS. Дополнительно в плагине реализовано определение структурного деления территории.

Для проведения расчетов на исследуемом водотоке пользователем должны быть указаны: слой для ограничения зоны интересов, слои для определения площади водотока, длины речной сети и других морфологических характеристик.

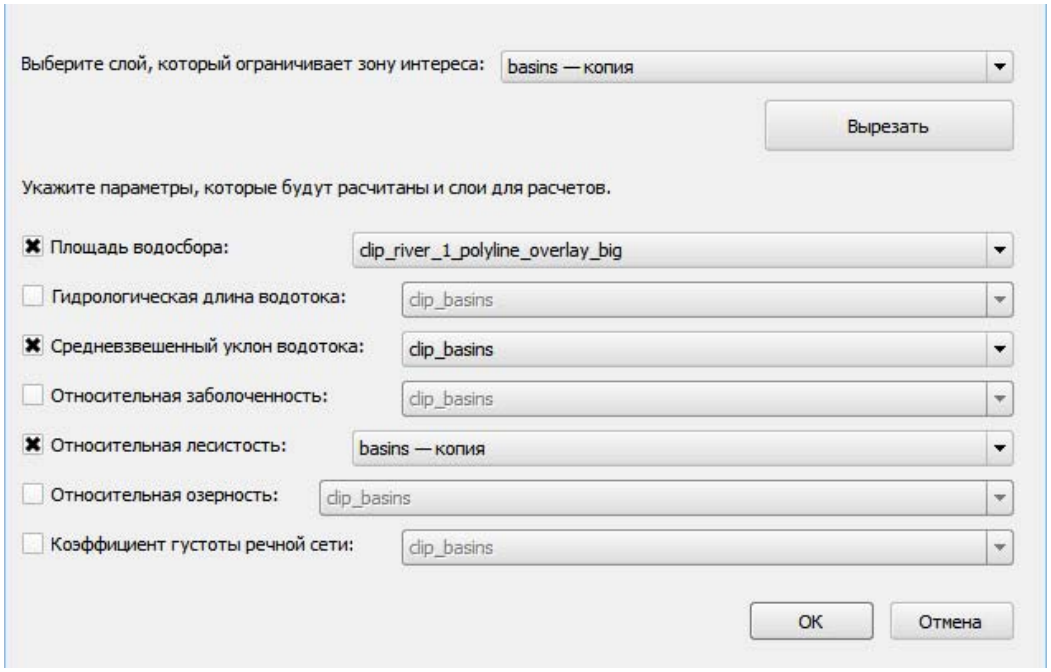


Рисунок 1 – Интерфейс плагина HydroCalc

Далее с учетом пользовательской информации определяются границы зоны интересов, площади водосбора, озер, лесов, болот и длина речной сети в выбранной зоне. С помощью плагина рассчитываются параметры территории и необходимые коэффициенты – характеристики водосбора.

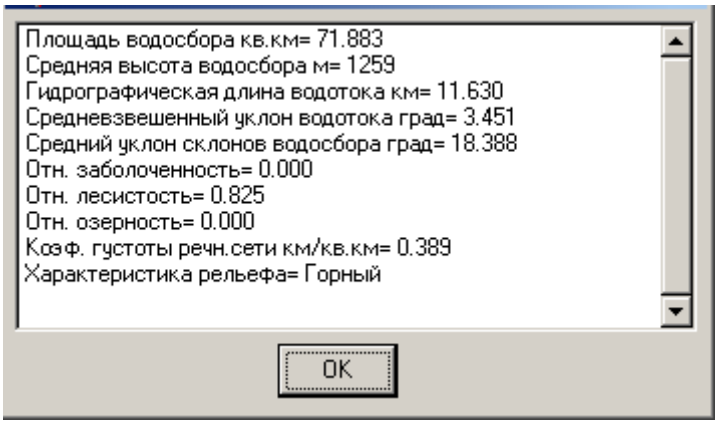


Рисунок 2 – Результаты работы плагина HydroCalc

Список литературы

1. Магрицкий Д.В. Речной сток и гидрологические расчеты: практические работы с выполнением при помощи компьютерных программ. – М.: Изд-во Триумф, 2014. – 184 с.
2. Расчет максимального стока весеннего половодья и дождевых паводков неизученных рек (FloodHigh): свидетельство о регистрации № 2001611052 Рос. Федерация / Жоров В.А., Воробьев Е.К., Ловцкая О.В., Яковченко С.Г.; правообладатель Научно-исследовательское учреждение Институт водных и экологических проблем Сибирского отделения Российской Академии Наук (ИВЭП СО РАН). – Заявка № 2001610547; дата поступления 4 мая 2001 г.; зарегистрирована в Реестре программ для ЭВМ 20.08.2001. – 1с.
3. Snipcalc v.2.0: свидетельство о регистрации № 2012619797 Рос. Федерация / Яковченко С.Г.; правообладатель Общество с ограниченной ответственностью «Центр инженерных технологий». – Заявка № 2012617603; дата поступления 11 сентября 2012 г.; зарегистрирована в Реестре программ для ЭВМ 30.10.2012. – 1с.
4. Клибашев К.П., Горшков И.Ф. Гидрологические расчеты. – Л.: Гидрометеоиздат, 1970. – 463 с.
5. СП 33-101-2003. Свод правил по проектированию и строительству. Определение основных расчетных гидрологических характеристик. Издание официальное. – М.: Госстрой России, 2004. 72 с.
6. Телегина М.В., Янников И.М. Геоинформационные системы и экологическое картографирование в подготовке студентов по специальности «Техносферная безопасность» // Научный журнал «Современные наукоемкие технологии» [Электронный ресурс]. – Электрон. текст. дан. – Режим доступа: https://www.top-technologies.ru/ru/article/view?id=31809. – Загл. с экрана
7. Руководство по определению гидрографических характеристик картометрическим способом // Электронный фонд правовой и нормативно-технической документации [электронный ресурс]. – Электрон. текст. дан. – Режим доступа: http://docs.cntd.ru/document/1200077599. – Загл. с экрана

ДЕЦЕНТРАЛИЗОВАННАЯ СИСТЕМА ХРАНЕНИЯ И ПОТВЕРЖДЕНИЯ ПОДЛИНОСТИ ПОРТФОЛИО СТУДЕНТА НА ОСНОВЕ ТЕХНОЛОГИИ BLOCKCHAIN

Станько И.В. – студент, Крючкова Е.Н. – к.ф.-м.н., профессор
Алтайский государственный технический университет (г. Барнаул)

Целью данной работы является описание реализации децентрализованной системы хранения и подтверждения подлинности портфолио студента на основе технологии blockchain. Почему именно blockchain?

В жизни мы сталкиваемся с ситуациями, когда мы вынуждены доверять посредникам, чтобы они подтвердили подлинность различных документов или договоров. Но почему мы должны доверять этим посредникам? Есть ли способ обеспечить достоверность заключения сделок, минуя посредников как таковых? Да, это может обеспечить blockchain.

Blockchain – это технология надежного децентрализованного хранения достоверных записей, о документах, договорах и даже криптовалюте, например Bitcoin. Все, что обычно записано на бумаге, можно записать и в blockchain с одним лишь отличием – в blockchain просто невозможно подделать или подменить эти записи [4].

1. Blockchain и традиционные системы хранения данных

Первое очень важное отличие от традиционных систем хранения данных - это децентрализованность. Все пользователи blockchain образуют собой сеть компьютеров, на каждом из которых хранится копия данных blockchain. Обычно это полная копия всех блоков, но, можно хранить лишь нужные на конкретном компьютере данные.

Благодаря этому выключить или сломать blockchain практически невозможно, поскольку для этого надо выключить или сломать все компьютеры. Пока есть хоть один пользователь, blockchain существует. Каждый новый пользователь расширяет и укрепляет эту сеть. Причем все компьютеры равноправны, там нет организаторов, модераторов, контролеров и менеджеров. Каждый отвечает за себя сам.

Второе отличие - в blockchain есть возможность устанавливать правила транзакций (бизнес-логику), которые привязаны к самой транзакции. Это отличает данную технологию от традиционных баз данных, в которых правила часто устанавливаются на уровне всей базы данных или в программном приложении, но не на уровне отдельной транзакции.

2. Структура проекта

Чтобы технологию blockchain можно было применить в нашем проекте, её нужно немного модифицировать. Во-первых, откажемся от блоков; у нас будет цепочка транзакций. Во-вторых, новые транзакции в цепочку могут добавлять только проверяющие узлы (далее валидаторы), которые принадлежат университету, они и проверяют законность такой транзакции. Таким образом, мы создадим управляемый распределённый реестр записей о студентах.

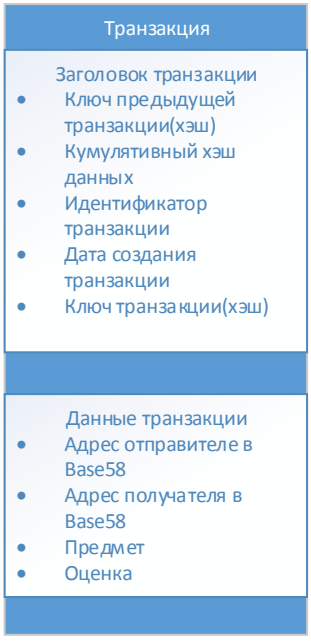


Рисунок 1 – Транзакция

Транзакция у нас будет иметь вид, представленный на рисунке 1. Рассмотрим ее элементы.

Адрес пользователя в сети формируется следующим образом. При регистрации в сети blockchain генерируется ключевая пара: закрытый ключ, открытый ключ. Закрытый ключ известен только владельцу, а открытый может быть опубликован. Далее берётся открытый ключ и производится SHA-256 хэширование. Потом результат хэширования снова хэшируем SHA-256 и результат конвертируется в Base58 [1]. Это и есть адрес в сети.

Хэш транзакции вычисляется по формуле:

$$\mathrm{H}_{i+1} = SHA-256(SHA-256(M_{i+1}) + SHA-256(H_i)), \quad (1)$$

где H_{i+1} - текущие транзакция, M_{i+1} - дерево Меркла для текущий транзакции [1].

А это значит, что в ключе любой транзакции закодированы не только записи этой транзакции, но и все предыдущие транзакции. Любое, даже самое незначительное, изменение данных в любой транзакции вызывает полное изменение его ключа, что, в свою очередь, потребует изменения ключей всех последующих блоков [4].

Схема работы приложения выглядит следующим образом. Студент после сдачи зачёта/экзамена называет преподавателю свой адрес в сети blockchain. Преподаватель в приложении составляет транзакцию: свой адрес в сети, адрес студента, оценка и предмет. Эта транзакция отправляется на узлы-валидаторы, которые принадлежат университету, там проверяется “законность” такой транзакции и вычисляет хэш транзакции.

Транзакция считается подтвержденной, если ее одобрили по крайней мере 60% проверяющих узлов. Для достижения консенсуса применим алгоритм PBFT (Practical Byzantine fault tolerance), основанный на задаче о византийских генералах [3]., хотя существует возможность подключения любого алгоритма

Рассмотрим задачу о византийских генералах немного подробнее. Византийская армия, состоящая из нескольких частей, окружила город. Каждой из частей армии командует генерал, и всем им нужно принять коллективное решение; часть генералов голосуют за атаку, остальные — за отступление. При этом все понимают, что атака только частью сил захлебнется, поэтому все согласны принять мнение большинства. Однако принятие решения усложняется тем, что среди генералов могут быть предатели, вводящие всех остальных в заблуждение, чтобы в итоге было принято неверное решение [5].

В качестве «генералов» у нас выступают проверяющие узлы, которые участвуют в процессе подтверждения транзакций. Предположим, что в сеть поступает транзакция, после её проверки узлы-валидаторы обмениваются между собой информацией о результатах работы, эти данные сравниваются и принимается решение о подтверждении транзакции. При обмене информацией появляется возможность определить «предателя», который всем остальным участникам процесса сообщил разные данные, — этот вредоносный узел помещается в карантин и в дальнейшем не учитывается, затем производится синхронизация данных среди всех участников сети blockchain.

Если кому-то из участников сети интересно просмотреть портфолио студента, то достаточно просто “пройтись” по всей цепочке транзакций и найти все записи соответствующие публичному ключу студента.

Выводы

Данная технология позволит крайне быстро взаимодействовать различным государственным структурам, которые связаны с обучением, так как они всегда будут иметь актуальную информацию. Работодателям легко и удобно проверять соискателей на должности. А благодаря децентрализованности можно не опасаться, что данные пропадут в результате несчастного случая или поломки оборудования.

Список литературы

1. Making Blockchain Real for Business. — Explained. [Электронный ресурс] — 2016. — URL:https://www-01.ibm.com/events/wwe/grp/grp308.nsf/vLookupPDFs/Blockchain Explained/$file/Blockchain Explained.pdf

3. Bitcoin: A Peer-to-Peer Electronic Cash System. [Электронный ресурс] — 2008. — URL: https://bitcoin.org/bitcoin.pdf
4. Земцов А.А Блокчейн для всех. [Электронный ресурс] — 2016. — URL: https://www.osp.ru/os/2016/04/13050989/
5. Худорожков Р.А Как blockchain изменит нашу жизнь? [Электронный ресурс] — 2016. — URL: http://rb.ru/opinion/blockchain/
6. Чащин Д. Алгоритм византийских генералов. [Электронный ресурс] — URL: http://blog.artlives.ru/programming/%d0%b0%d0%bb%d0%b3%d0%be%d1%80%d0%b8%d1%82%d0%bc-%d0%b2%d0%b8%d0%b7%d0%b0%d0%bd%d1%82%d0%b8%d0%b9%d1%81%d0%ba%d0%b8%d1%85-%d0%b3%d0%b5%d0%bd%d0%b5%d1%80%d0%b0%d0%bb%d0%be%d0%b2/

КЛАССИФИКАЦИЯ ФРАГМЕНТОВ ТЕКСТОВ ПО ИХ ФУНКЦИОНАЛЬНОМУ НАЗНАЧЕНИЮ

Ткаченко Е.С. – студент, Крайванова В.А. – к.ф.-м.н., доцент
Алтайский государственный технический университет (г. Барнаул)

На текущий момент для обеспечения аналитической работы с большими объемами данных создаются хранилища корпоративных и открытых документов. Важным элементом, позволяющим автоматизировать обработку и анализ документов, в таких хранилищах, являются средства автоматической классификации [1]. При этом обычно рассматривается задача классификации текстов целиком без выделения в них значимых фрагментов. Во многих практических случаях это является существенным ограничением. В первую очередь это длинные, неоднородные по внутренней структуре тексты, такие как законы, научные статьи и монографии, техническая документация.

Задача классификации фрагментов текстов по их функциональному назначению может быть разделена на три подзадачи [2]:

- классификация – определение принадлежности текстов к одной или нескольким заранее определенным категориям (это необходимо для определения того, какие в принципе функциональные фрагменты имеет смысл искать в данном тексте);
- сегментация – разделение текстов на тематически однородные фрагменты (выделение в тексте групп предложений, относящихся к одной теме или выполняющих одну функцию);
- реферирование – выделение значимых предложений в тексте с целью построения его краткого изложения (выделение не всех значимых предложений, а только относящихся к определенной тематике).

Текст на естественном языке представляет собой систему со сложной, неравномерной внутренней структурой, с разной степенью формализованности. Именно поэтому прежде чем применять алгоритмы классификации необходимо выделить из текста фрагменты, которые затем иерархически будут объединены в функциональные единицы. Каждый новый уровень в этой иерархии семантически представляет собой нечто большее, чем просто сумму его составляющих. Рассмотрим уровни функциональных фрагментов в тексте:

- Самостоятельные части речи: существительное, глагол, прилагательное, наречие, причастие, деепричастие.
- Словосочетания, классифицируемые по вопросу: определительные (предмет и его признак), объектные (называют предметы, действия, признаки и т. д., но более точно,

более конкретно, чем слова: читать — читать вслух), обстоятельственные (указывают на действие и его признак)

- Фрагменты внутри предложения, представляющие собой различные связки из словосочетаний. Такие фрагменты формируются с помощью служебных частей речи: предлог, союз, частица, междометие (например, пространственные предлоги, соединительные союзы, простые частицы).
- Предложения с учетом их структуры, а именно видами подчинительной и сочинительной связи (при наличии таковой).
- Более крупные элементы: внутренне связанные последовательности предложений (определения, перечисления, описания объектов и процессов, повествования), которые могут как совпадать, так и не совпадать с формальными фрагментами в тексте: абзацами, разделами, главами и т.д.

Выполняя семантический разбор текстов важно учитывать различные факторы, такие как:

- омонимичные и синонимические структуры, а также явление полисемии [3];
- возможные орфографические (включая опечатки), пунктуационные ошибки;
- вставки на иностранных языках.

При работе в качестве корпуса текстов выбрана русскоязычная Википедия, потому что она содержит большое число текстов разнообразной тематики и при этом имеет частично формализованную структуру, что позволяет использовать ее для алгоритмов обучения с учителем. Преимущества при работе с данным корпусом следующие.

Во-первых, тексты статей составляются и размечаются с учетом правил и указаний. Это позволяет говорить о достаточной однородности стиля текстов, а также о единых правилах создания заголовков и других формальных элементов страницы. Во-вторых, страницы Википедии содержат в себе формализованные объекты - шаблоны инфоблоков - в которых в виде фрейма сконцентрирована самая важная информация статьи. На основе этой разметки можно получить частичную информацию о различных типах фрагментов, которые возможно выделить в той или иной статье. В-третьих, существуют правила именования статей, которые подчиняются реалиям энциклопедического и научного стиля речи, принципам организации Википедии. И наконец, выделенные в текстах внутренние разделы отражают их логическую структуру и повторяются в статьях одного типа.

Для извлечения функциональных фрагментов нижних уровней из текста на естественном языке при помощи контекстно-свободных грамматик и словарей ключевых слов используется “Томита-парсер”. Он позволяет писать свои грамматики и добавлять словари для нужного языка. Исходный код проекта открыт и выложен на GitHub. После извлечения функциональные фрагменты необходимо преобразовать в элементы формальной модели текста. Для этого проводится кластеризация выделенных фрагментов, и последующая классификация выделенных кластеров. На этом этапе каждому выделенному функциональному фрагменту ставится в соответствие его тип и другая необходимая информация.

После этого формируется формальное представление для более крупных функциональных фрагментов на основе полученных данных. Затем кластеризуются и они. Процесс продолжается, пока не будет достигнут верхний уровень иерархии - целый текст. Таким образом, формируется иерархия, на верхних уровнях которой окажутся уже группы предложений - функциональные фрагменты последнего уровня.

Ведется разработка ПО для анализа полученных данных с использованием парсера, где реализуются как логика выделения сложных функциональных фрагментов, так и способы их хранения и представления.

Список литературы

1. Webber B., Egg M., and Kordoni V. Discourse structure and language technology, Natural Language Engineering, 18(4), 437-490, 2012
2. Васильев В.Г. Выделение фрагментов в текстах при классификации [Электронный ресурс] - Режим доступа: http://www.dialog-21.ru/digests/dialog2009/materials/html/10.htm, свободный (дата обращения: 13.04.17)
3. Большой энциклопедический словарь. Языкознание. Гл. ред. ... Ярцева В.Н. М.: Большая Российская энциклопедия, 1998. – 344 с.

ИЕРАРХИЧЕСКАЯ МОДЕЛЬ ФУНКЦИОНАЛЬНОЙ СТРУКТУРЫ НАУЧНЫХ ТЕКСТОВ

Токарев В.И. – студент, Крайванова В.А. – к.ф.-м.н., доцент
Алтайский государственный технический университет (г. Барнаул)

В последнее время области информационных технологий, связанных с анализом естественных языков, становятся очень популярными, благодаря большому количеству прикладных исследований, позволяющих решать всё более широкий круг насущных задач. Суть анализа естественных языков заключается в том, чтобы на основе взаимной встречаемости слов или словосочетаний находить закономерности в тексте, и преобразовывать эти закономерности в некоторую формальную модель, реализуя «понимание» смысла текста [1] в пределах конкретной предметной области. То есть, компьютер не понимает, что такое деньги, но знает, что оно встречается с такими словами как банк, кредит, купить, продать и подобное, так как в процессе анализа текста были обнаружены связи между этими сломами и словосочетаниями.

В нашей работе стоит цель реализовать инструмент, который на основе частично размеченных корпусов текстов будет стоить граф из фрагментов текстов, что позволит осуществлять семантический поиск по небольшому фрагменту текста. То есть, человек пишет небольшой тезис, включающий в себя краткое описание процесса, явления, научного исследования или события, а затем пропускает его через обученную иерархическую модель фрагментов текстов. Данный инструмент поможет проверить уникальность работы, в плане похожих исследований, либо найти более подробные исследования того, что было упомянуто в анализируемой статье. Необходимо, что бы система могла дообучаться и встраивать новый текст в уже сформированный граф. Это будет экономить вычислительные ресурсы и время.

Проблемы, которые необходимо решить в процессе реализации парсера предлагаемой модели функциональной структуры текстов – это омонимия различных видов и произвольный порядок слов в предложении русского языка. Это свойство порождает большое разнообразие конструкций, которые необходимо унифицировать. Также, в научных статьях зачастую используется большое количество вербоидов (отглагольных существительных и прилагательных, причастий и деепричастий) вместо глаголов. В связи с этим, перед тем, как анализировать текст, его необходимо обработать, и привести слова к нужной форме. Для построения модели нижнего уровня можно использовать разные инструменты [2], например, word2vec [3, 4]. Это не является какой-то технологией или алгоритмом, это набор алгоритмов и инструментов, разработанных специалистами Google, для собственных нужд и выложенный в открытый доступ в качестве open source инструмента для других разработчиков.

Связным фрагментом текста назовем некоторый составной элемент, который семантически значит больше, чем элементы, его составляющие. Например, смысл

предложения – это больше, чем смысл набора составляющих его слов, перечисленных через запятую. Чтобы не нарушать общность математической модели, минимальным фрагментом текста будем считать словоформу одной из основных частей речи (глагол, существительное и т.д.). Формально, фрагментом текста будем называть граф, вершины которого представляют собой фрагменты текста, а ребра – связи между этими фрагментами, оформленные с помощью морфологического согласования, служебных частей речи (предлогов, союзов, местоимений и других анафорических конструкций) или только семантически. Таким образом, каждый фрагмент на своем уровне иерархии связан функциональными отношениями с соседними фрагментами, и отношениями вложенности с фрагментами следующего и предыдущего уровней. Разделение текста на такие фрагменты неоднозначно. Конкретный способ разбиения определяется результатами обучения модели.

Данную модель можно обобщить на коллекцию текстов. В этом случае фрагменты связаны функциональными связями внутри текста, из которого они извлечены, и тематическими связями с фрагментами из других текстов. Внутри общей иерархической модели коллекции текстов все тексты связаны в единый граф, вершинами которого уже являются графы из фрагментов самого текста. То есть в итоге получается граф, каждая вершина которого тоже некий граф, позволяющий каким-то образом систематизировать информацию. Организованные в такой модели данные не являются онтологией или семантической сетью в их классическом понимании. Скорее можно провести аналогию с результатом работы сверточных рекурсивных нейронных сетей. Данные можно визуализировать для пользователя и показать наглядно, какие статьи (или фрагменты статей) можно прочитать и почему они, по мнению системы, относятся к той или иной статье, то есть модель обладает инструментами логического вывода.

Возникает вопрос, где брать данные для обучения системы и насколько разнообразными могут быть данные. Касательно источника данных, если речь идет о научных статьях, то это могут быть открытые базы тезисов с конференций, статей из научных журналов [6], или выпускных квалификационных работ, которые с 2016 года должны быть выложены в открытый доступ после защиты работы, а также учебные материалы из открытых источников [5]. Источником предварительного обучения модели может являться Википедия. Предлагаемый метод может быть применен для коллекций новостных статей, научных публикаций, исторических заметок и многого другого, что можно как-то связать друг с другом. Граф, составленный из источников различной тематики можно дополнительно разметить тематическими маркерами, что должно повысить скорость и качество итогового поиска.

Список литературы

1. Ефремова М.И. Автоматический разбор и аннтирование статей // Фундаментальные исследования. 2015. № № 2-22 / 2015. С. 4866 – 4870.
2. Gensim [Электронный ресурс]. // Gensim – Режим доступа: http://radimrehurek.com/gensim/index.html, свободный.
3. Word2Vec в примерах [Электронный ресурс]. //Хабрахабр – Режим доступа: https://habrahabr.ru/post/249215, свободный.
4. Немного про word2vec: полезная теория [Электронный ресурс]. //NLPx – Режим доступа: http://nlpx.net/archives/179, свободный.
5. Единое окно доступа к информационным ресурсам [Электронный ресурс]. // Единое окно – Режим доступа: window.edu.ru, свободный.
6. Научная электронная библиотека [Электронный ресурс]. // Научная электронная библиотека – Режим доступа: elibrary.ru, свободный.

АВТОМАТИЗИРОВАНИЕ УЧЕТА ОБОРУДОВАНИЯ В АКЦИОНЕРНОМ ОБЩЕСТВЕ «РОССЕЛЬХОЗБАНК»

Чарапкин Я.С., Старикова В.К. – студенты, Троицкий В.С. – к.ф.-м.н., доцент
Алтайский государственный технический университет (г. Барнаул)

Инвентарный учет оборудования - это мутный процесс, превращающийся в головную боль для любого человека, которому его поручили. Да и сам вопрос, что такое инвентаризация и какие задачи перед ней стоят, однозначного ответа не имеет. Есть две противоположных точки зрения. IT-администраторы считают, что это учет начинки внутри компьютера и для решения этой задачи привлекают всякие системы вроде Aida64 или подобные способы учета и мониторинга. Бухгалтер ожидает от инвентарного учета, ответ на простой вопрос: на месте ли предмет с инвентарным номером ABC123? И соответствует ли инвентарному номеру ABC123 то изделие, которому этот номер был присвоен? Таким образом, бухгалтеров вообще не интересует то, что установлено внутри компьютера, и инвентаризация для них — это элемент бухучета в специализированных бухгалтерских программах. Но чаще всего, под инвентаризационным учетом понимается нечто среднее, автоматизации чего и посвящена наша работа.

Несмотря на большой выбор систем по инвентаризации, готовой системы, удовлетворяющей требованиям заказчика, не нашлось. И этому есть тысяча причин (например, оригинальный учет, изменять который заказчик отказался). Т.е. готовую систему учёта придётся дорабатывать своими руками, что резко уменьшает привлекательность этого подхода. По этой причине было принято решение о разработке оригинальной системы учета. В АКЦИОНЕРНОМ ОБЩЕСТВЕ «РОССЕЛЬХОЗБАНК» уже имеется софт, в котором ведется учет материальных ценностей. Функционал его весьма ограничен, а технические решения, положенные в его основу, не позволяют без существенных затрат, его дорабатывать и развивать. Поэтому существующее решение мы рассматриваем только как источник данных о соответствии описаний единиц хранения их инвентарным номерам. В рамках данной работы мы сделаем так, чтобы эта информация была наглядно представлена и могла быть легко использована при инвентарном учете оборудования.

На наш взгляд, система должна быть простой, иначе её будут просто игнорировать. В идеале это выглядит так: у каждого сотрудника, который занимается инвентаризацией на руках есть смартфон (планшет), которым они могли бы сканировать коды поднося смартфон (планшет) к устройству. Результатом сканирования было бы получение на экране смартфона (планшета) информации об объекте инвентаризации и отметок в актуальном списке подлежащего учёту оборудования. С таким инструментарием инвентаризацию смогут провести даже самые неопытные сотрудники с низкой мотивацией. К сожалению, мы создаем систему не для нового предприятия, а для уже работающего. Где традиционно используется распечатка из БД с сортировкой по кабинетам, обход и отметка карандашом проверенных объектов и последующее внесение изменений в БД. Да и старые этикетки с инвентарными номерами машиночитаемых кодов не содержат. Поэтому, в создаваемой системе останется поддержка и такой технологии.

Система учета создается на современной WEB-платформе (apache, mysql, php), позволяющей ее дальнейшее развитие. Интерфейс соответствует пожеланиям заказчика, имеет адаптивный дизайн. В работе использованы только продукты имеющие бесплатную лицензию. Кроме непосредственно учета оборудования предусмотрено хранение информации о его месторасположении, материально ответственном лице, ремонте, подсистема отчетности и импорт данных из существующего решения. По нашим оценкам, разработка системы учета должна быть завершена до конца мая 2017 года.

РАЗРАБОТКА АРХИТЕКТУРЫ СИСТЕМЫ ИДЕНТИФИКАЦИИ ПРЕДАВАРИЙНЫХ СИТУАЦИЙ ТЕХНОЛОГИЧЕСКИХ ПРОЦЕССОВ

Шахов Д.Е. - студент
Алтайский государственный технический университет (г. Барнаул)

Современное развитие технологий позволяет производителям различной техники внедрять в нее программные модули, отвечающие за сбор и передачу необходимой информации в центры по обработке данных. Собранная информация может использоваться для оценки предоставляемых услуг, в маркетинговых целях, для улучшения технических характеристик существующих и разрабатываемых устройств, для своевременного выявления и предотвращения аварийных ситуаций. Обеспечение надежной работы технологического оборудования промышленных предприятий достигается за счет своевременного проведения его технического обслуживания и ремонта. Для поддержания высокого уровня безопасности и экологической чистоты производства используются системы, позволяющие заблаговременно идентифицировать возможный переход процесса в аварийный режим и предотвратить его остановку. В подобных системах должен проводиться логический и количественный анализ информации о функционировании технологических объектов, идентификация и прогнозирование состояний, расчет соответствующих режимов работы. Описанные системы используются для мониторинга различных технических объектов, от смартфона и кухонной плиты до атомных реакторов.

Для разработки программного обеспечения, позволяющего решать задачи прогнозирования аварийных и предаварийных ситуаций, необходима эффективная методика их идентификации. Создание эффективной системы идентификации предаварийных ситуаций технологического процесса, позволяющей повысить его безопасность и безаварийность, является актуальной научной и практической проблемой. Для построения системы идентификации предаварийных ситуаций необходимо решение следующих задач:

- рассмотрение предметной области задачи идентификации предаварийных ситуаций [1];
- построение модели системы идентификации предаварийных ситуаций [2, 3];
- разработка системы идентификации предаварийных ситуаций и соответствующей базы знаний.

Рисунок 1 - Архитектура автоматизированной системы мониторинга

На базе ОАО "Научно-инженерный центр электротехнического университета" была разработана автоматизированная система мониторинга надежности и безопасности сложных технических объектов [4]. На рисунке 1 представлена архитектура разработанной системы. Из доступной информации не ясна стоимость поставляемой, ОАО "Научно-инженерный центр электротехнического университета", системы и её технические характеристики.

В связи с изложенными выше причинами было решено разработать модульную систему, осуществляющую прием, обработку, сохранение и выдачу информации, поступающей от технических объектов.

Система агрегирует в хранилище информацию, получаемую от большого числа типовых клиентов-поставщиков. Информационные сообщения могут поставляться несколькими типами авторизуемых клиентов-поставщиков. В зависимости от типа клиента-поставщика состав и структура поставляемой им информации различаются и со временем могут изменяться. Информация от клиентов-поставщиков может поступать в нескольких режимах:

- регулярные передачи данных по сети;
- передача данных по сети, только во время технического обслуживания объекта;
- внесение данных, собранных на информационный носитель, при помощи интерфейса взаимодействия.

В роли клиентов-потребителей могут выступать пользователи и программные подсистемы. Основные совершаемые пользователями операции – поиск/фильтрация информации и построение динамических отчетов. Программные подсистемы на основе хранящейся информации могут осуществлять мониторинг общего состояния существующих технических объектов, их настройку и усовершенствование. Программный анализ большого объема информации с применение математических моделей позволяет производить прогнозирование предаварийных ситуаций и предотвращение аварийных ситуаций.

Наличие подобной модульной системы позволит предприятию повысить качество сборки, обслуживания и работы технических объектов, что заметно скажется на конкурентоспособности производителя на рынке предоставляемых им услуг.

Список литературы

1. Жедунов Р. Р. Моделирование предметной области задачи управления технологическим процессом / Теоретический анализ и математическое моделирование информационных систем: отчет по НИР (госбюд., промежут., за 2004 год института информационных технологий и коммуникаций) рег. № РК 0120.0 504515. – Астрахань, 2005. – 140 с.
2. Жедунов Р. Р. Модель распределенной системы идентификации предаварийных ситуаций технологических процессов / Вестн. Астрахан. гос. техн. ун-та. – 2006. – № 1. – С. 152–157.
3. Жедунов Р. Р. Моделирование процесса распознавания предаварийных ситуаций системы управления технологическим процессом / Компьютерное моделирование-2005: тр. VI Междунар. техн. конф. – СПб.: Изд-во политехн. ун-та, 2005. – С. 365–367.
4. Интеллектуальные системы мониторинга и контроля состояния технически сложных объектов [Электронный ресурс] / Научно-инженерный центр электротехнического университета. – Режим доступа: http://www.nicetu.ru/ru/reshenija-i-produkty/intellektualnye-sistemy-monitoringa-i-kontrolja-sostojanija-tekhnicheski-slozhnykh-obektov/

РАЗРАБОТКА МОДУЛЯ WEB-ПРИЛОЖЕНИЯ ДЛЯ ОБЕСПЕЧЕНИЯ ОБМЕНА ДАННЫМИ В РЕЖИМЕ РЕАЛЬНОГО ВРЕМЕНИ С ИСПОЛЬЗОВАНИЕМ ТЕХНОЛОГИИ WEBSOCKET

Шкирков А.Ю. – студент, Крючкова Е.Н. – к.ф.-м.н., профессор,
Кочанов И.А. – технический директор*
Алтайский государственный технический университет им. И.И. Ползунова (г. Барнаул)
ООО «ЛинксСофт»* (г. Барнаул)

Разработка данного модуля велась в рамках реального проекта для компании, занимающейся организацией вертолётных экскурсий по острову.

В целом разработка проекта была связана с автоматизацией логистики процесса оказания услуг. Одна итерация оказания услуги состоит из следующих укрупнённых этапов:

1. Диспетчер формирует расписание полётов вертолётов, исходя из предварительно поданных заявок пассажиров.
2. Водитель забирает пассажиров из 5 различных зон, по очереди проезжая каждую из них. При этом предварительно водителю должно присылаться уведомление от диспетчера о количестве пассажиров в каждой из зон. Водитель же должен уведомлять диспетчера о том, что все пассажиры были забраны из зоны, или же возникли какие-то проблемы. По окончании сбора всех пассажиров водитель сообщает, что поехал на аэродром.
3. Наземная служба получает уведомление, что водитель движется на аэродром и начинает подготавливать вертолёт к вылету. После прибытия пассажиров их необходимо взвесить и принять платёжные документы.
4. Операторы принимают информацию о пассажирах, рассчитывают баланс вертолёта, при необходимости сообщают наземной службе, что необходимо пересадить пассажиров, и в дальнейшем управляют процессом полёта.
5. После полёта водитель развозит пассажиров обратно по зонам. После того, как все пассажиры развезены, он уведомляет об этом диспетчера.

Целью разработки модуля для обмена данными является реализация механизма коммуникации между работниками компании. Критическим моментом всего процесса является время. К примеру, сбор пассажиров водителем занимает около 20 минут, и за это время наземной службе необходимо должным образом подготовить вертолёт. Таким образом, крайне важно, чтобы все уведомления между работниками компании приходили как можно быстрее.

Исходя из цели разработки было выведено несколько вариантов автоматизации процесса.

Первый вариант – использование услуг SMS-агрегатора для рассылки уведомлений каждому из участников процесса. Плюсом этого метода является достаточно невысокая стоимость и время интегрирования в бизнес-процессы компании. Однако этот метод имеет несколько существенных недостатков. Во-первых, появляется внешняя зависимость от компании, предоставляющей услуги рассылки SMS. Во-вторых, появляются довольно существенные затраты на рассылку сообщений. Например, одно сообщение на мобильный телефон оператора T-Mobile стоит $0.01[1], а в день может быть отправлено несколько тысяч таких сообщений. Таким образом, только за одну неделю стоимость такого обслуживания составит около 200$.

Второй вариант – разработка модуля, периодически опрашивающего web-сервер на предмет изменения статусов участников процесса. Плюсом такого варианта является отсутствие оплаты за отправку или получение уведомлений. Но очевидные проблемы связаны с тем, что невозможно получить уведомление в режиме реального времени. В любом случае необходимо установить некоторый интервал опроса web-сервера, причём чем меньше этот интервал, тем сильнее будет нагрузка на сервер.

Третий вариант – разработка модуля, устанавливающего соединение с сервером на основе протокола WebSocket[2, 3]. У этого варианта отсутствуют проблемы двух прошлых

методов – нет оплаты за работу с уведомлениями, уведомления не уходят за пределы компании, и нет нагрузки на web-сервер. Отсутствие нагрузки объясняется принципом работы протокола WebSocket. Изначально клиент и сервер устанавливают соединение по протоколу HTTP, клиент спрашивает у сервера, включена ли на нём поддержка протокола WebSocket, при положительном ответе клиент и сервер обмениваются ключами доступа, завершают общение по HTTP, и в дальнейшем работают друг с другом только на основе протокола WebSocket до момента закрытия соединения. При этом для получения уведомлений клиент не будет отправлять запросов серверу – сервер сам разошлёт необходимые данные всем клиентам, подключенным к нему. Скорость рассылки данных близка к реальному времени – отправка производится без использования каких-либо временных интервалов, а сразу после того, как данные будут сформированы на сервере и готовы к отправке.

Программная реализация модуля выполнена в виде модуля к веб-приложению, работающему на фреймворке Java Spring.

Работу модуля можно продемонстрировать на примере: допустим, водитель собирает пассажиров на полёт, который состоится в 8 часов утра. Необходимо забрать пассажиров из второй зоны. Как только он прибывает в эту зону, он нажимает кнопку STANDING BY на своей странице. При нажатии кнопки на сервер по протоколу WebSocket отправляются данные о том, что готовится посадка пассажиров. Сервер получает данные и тут же отправляет их на экран диспетчера и водителя – в результате на обоих страницах таблица окрашивается в жёлтый цвет.

На рисунке ниже отображены 2 страницы – водителя и диспетчера, и на каждой из них - консоль получения данных от сервера.

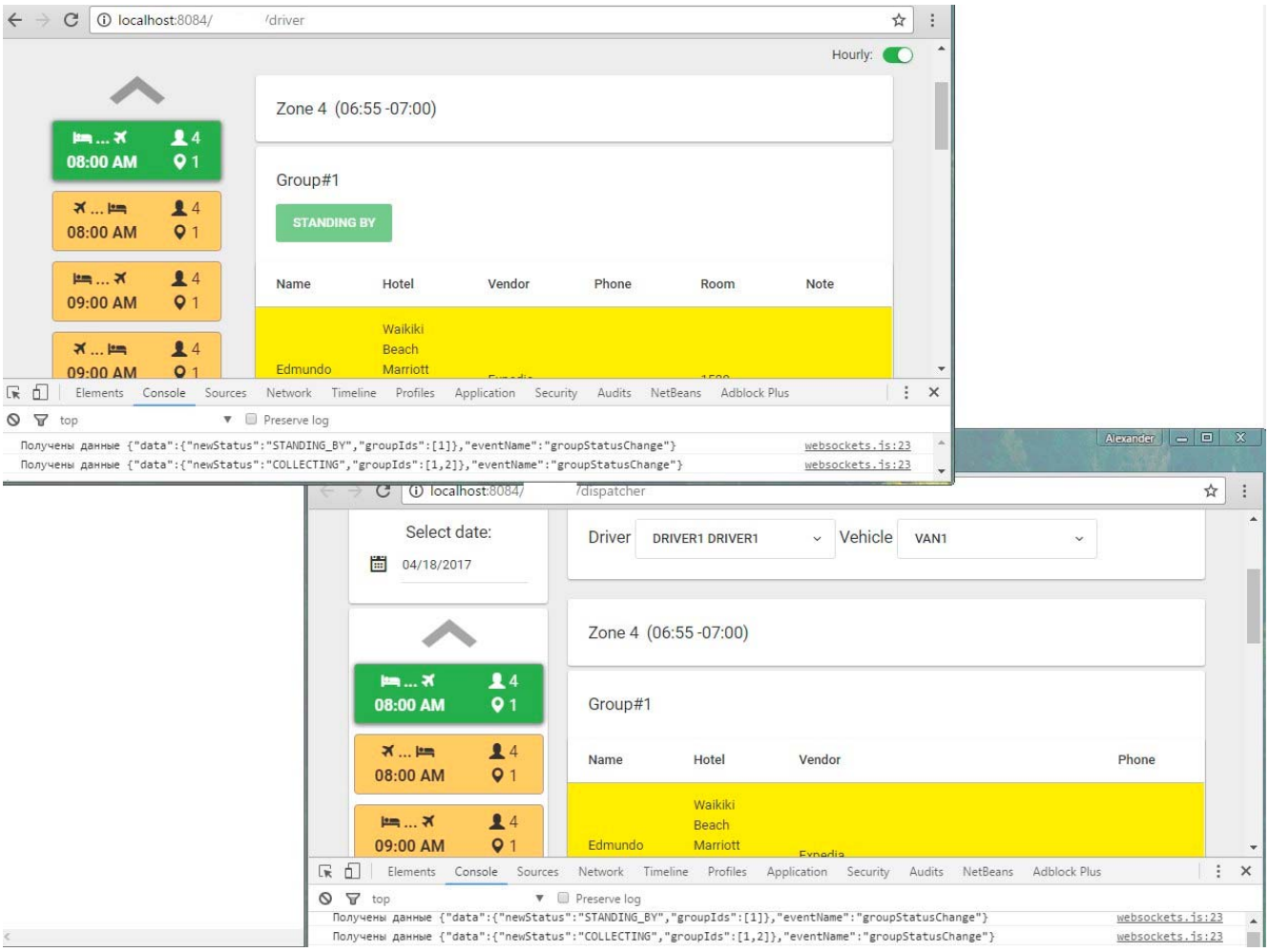


Рисунок 1 – Результат приёма данных по протоколу WebSocket

Список литературы

1. Данные о стоимости рассылки SMS-оповещений // CLX - Pricing Globally for SMS, MMS and Numbers [электронный ресурс]. – Электрон. текст. дан. – Режим доступа: https://www.clxcommunications.com/pricing/. – Загл. с экрана
2. Информация о протоколе WebSocket // WebSocket – JavaScript.ru [электронный ресурс]. – Электрон. текст. дан. – Режим доступа: https://learn.javascript.ru/websockets/. – Загл. с экрана
3. Информация об использовании протокола WebSocket // WebSockets – полноценный асинхронный веб – Хабрахабр [электронный ресурс]. – Электрон. текст. дан. – Режим доступа: https://habrahabr.ru/post/79038/. – Загл. с экрана

СИСТЕМА ВИЗУАЛИЗАЦИИ МОДУЛЬНОЙ СТРУКТУРЫ ПРОГРАММНЫХ ПРОДУКТОВ ПО ИХ ТЕКСТОВЫМ ОПИСАНИЯМ НА ПРИМЕРЕ СТУДЕНЧЕСКИХ РАБОТ

Ренёв Н.С. – студент, Крайванова В.А. – к.ф.-м.н., доцент
Алтайский государственный технический университет (г. Барнаул)

Основной универсальной формой передачи информации является текст. Но человек гораздо лучше воспринимает графическую информацию в виде изображений, графиков, схем. При работе с информацией с большой смысловой нагрузкой, в частности с различными документами, описывающими архитектуру программного продукта, требуется большое количество времени для того, чтобы понять суть текста. В связи с этим актуальным является разработка моделей генерации графического представления данного текста, что позволит в разы упростить восприятие текста.

Целью данной работы является разработка механизма извлечения структурированной информации о программном продукте по его описанию на естественном языке и визуализации этой информации.

Основную задачу можно разбить на три подзадачи: извлечение сущностей из текста, которые являются компонентами ПО, установление связей между компонентами и визуализация полученных результатов. В качестве компонентов могут выступать произвольные составляющие программного комплекса, например: модуль, функция, класс, файл. Компоненты могут быть связаны друг с другом различными способами: вложенность, передача данных, использование, реализация.

Для решения этих трех подзадач мы разработали алгоритм, состоящий из последовательного выполнения следующих этапов:

1. Токенизация (разбиение текста на токены — слова, знаки препинания).
2. Морфологический анализ текста (для каждого слова определить морфологические характеристики: род, число, падеж, части речи).
3. Извлечение компонентов из текста (нахождение участков текста, которые описывают компонент).
4. Отождествление компонентов (необходимо определить являются ли два различных участка текста описанием одного и того же компонента).
5. Установление связей между компонентами.
6. Удаление несвязанных компонентов.
7. Конвертация полученных компонентов и связей между ними в текстовой формат (например XML или формат проекта PlantUML[1]).
8. Построение графического представления на основе полученных результатов.

Для извлечения компонентов из текста был выбран способ на основе грамматик с использованием Томиты-парсера[2]. Томита-парсер - open-source инструмент для извлечения структурированных данных (фактов) из текста на естественном языке от компании Yandex, который специализируется на русском языке. Факты — таблицы с колонками, которые называются полями фактов и могут быть заполнены словами из текста или произвольными значениями. Извлечение фактов происходит при помощи контекстно-свободных грамматик и словарей ключевых слов. При возникновении неоднозначности в грамматике выполняется обход в ширину и выбирается наидлиннейшая цепочка. Парсер извлекает из текста все цепочки, которые выводятся из аксиомы грамматики. Части извлеченной цепочки терминалов могут быть интерпретированы, как поля для фактов[3].

Томита-парсер позволяет реализовать первые два этапа алгоритма. Благодаря морфологическому разбору можно использовать в качестве терминалов слова с определенными морфологическими признаками, например существительное в именительном падеже и единственном числе. Структура извлекаемых фактов имеет следующие поля: тип компонента, название компонента, описание компонента и флаг того, является ли описание компонента указателем (например “этот компонент” или “данный компонент”). Факты с данной структурой далее называются компонентами.

Для извлечения компонентов из текста нами разработаны следующие шаблоны:

- [компонент][описание компонента], например «модуль пользовательского интерфейса».
- [компонент][действие][описание действия], например «модуль, осуществляющий межпроцессорное взаимодействие»
- [характеристика компонента][компонент], например «семантический модуль».
- [название компонента на английском], например «OpenGL» или «GUI».
- [указатель][компонент], например «этот модуль» или «данный модуль».

Где [компонент] — это любое слово, которое хочет пользователь программы. На рисунке 1 приведены результаты работы грамматики для извлечения по предложенным шаблонам.

Рассмотрим модуль , принимающий данные пользователя и делающий первоначальную обработку . **EOS**

Синтаксический модуль строит синтаксическое дерево и сохраняет его и XML . **EOS**

Этот модуль также осуществляет первоначальный морфологический разбор . **EOS**

На первом этапе используется функция для разбиения предложения на части . **EOS**

Данные передаются модулю визуализации сцены , который создает графическое представление . **EOS**

Модуль для взаимодействия с пользователем . **EOS**

Component			
Type	**Name**	**Description**	**Pointer**
модуль		принимающий данные пользователя и делающий первоначальную обработку	false
модуль		синтаксический	false
default	XML		false
модуль		этот	true
функция		для разбиения предложения на части	false
модуль		визуализация сцены	false
модуль		для взаимодействия с	false

Рисунок 1 - Результат работы томита-парсера.

Как видно из результатов работы, программа хорошо справляется с извлечением перечисленных типов шаблонов, но плохо справляется с извлечением компонентов, в описании которых присутствуют предлоги. Кроме того, на практике нередко встречаются сложные типы шаблонов, которые не покрываются списком уже разработанных.

Сейчас ведется разработка паттернов извлечения связей. После решения задачи установления связей между компонентами будет разработана система визуализации, где в качестве инструмента визуализации модульной структуры будет использоваться PlantUML[1].

В дальнейшем данную систему можно улучшить, дополнив новыми шаблонами для извлечения компонентов или расширив словари, что делает её очень гибкой, но и поддерживать такую систему будет нелегко. В перспективе можно доработать систему, позволив пользователю самому определять вид извлекаемых компонентов и решить проблему со сложными конструкциями с помощью машинного обучения.

Список литературы

1. Сайт проекта PlantUML [Электронный ресурс]. - Режим доступа: http://plantuml.com/
2. М. А. Артемов, А. Н. Владимиров, К. Е. Селезнев, “Обзор систем анализа естественного текста на русском языке” // Вестник Воронежского государственного университета. Серия: Системный анализ и информационные технологии, 2013, № 2, 189–194.
3. Документация по Томита-парсеру [Электронный ресурс]. - Режим доступа: https://tech.yandex.ru/tomita/doc/

ОПЫТ АДАПТАЦИИ УЧЕБНЫХ МАТЕРИАЛОВ ДЛЯ ШКОЛЬНИКОВ ПОДПЕРСОНАЛЬНЫЕ МОБИЛЬНЫЕ ЦИФРОВЫЕ УСТРОЙСТВА

Леденева Е.А., Липатов В.А., Реутов А.С. - студенты
Сорокин А.В. – к.т.н., доцент, Рогозин К.И. – к.х.н., доцент
Алтайский государственный технический университет (г. Барнаул)

Наметившая устойчивая тенденция к потере интереса у учеников к бумажным носителям информации, а также электронным источникам в сети Интернет, требует поиска новых путей доставки и представления учебных материалов. Именно этому посвящена наша работа и представляемые тезисы. Бумажные учебники не соответствуют ожиданиям современных школьников, нацеленных на быстрое получение достаточной и небольшой по объему информации. Ограниченность средств, используемых в бумажных носителях, не позволяют эффективно сформировать в сознании обучаемых законченный образ изучаемых процессов и явлений. «Информационное цунами», которым характеризуется сеть Интернет, делает сложным, а иногда и принципиально невозможным достижение названной выше цели.

Исследования последних лет показывают, что основным способом улучшения результатов обучения является повышение мотивации участников учебного процесса. Одним из путей реализации данной цели является повышение интереса к учебным материалам, которые должны порождать у учащихся стимулы к освоению требуемых знаний и получению заданных компетенций.

Эффективными, по нашему мнению, могут стать такие учебные модальности, в которых используются гаджетоспецифические потенции, такие, как мультимедийность, интерактивность, многозадачность и нелинейность получения (и/или использования) знаний в ходе особым образом организованной учебной деятельности с применением мобильных

цифровых устройств, которые в настоящий момент имеются у всех обучающихся.

В качестве научной области, в которой создаются учебные материалы, нами был выбран предмет «ФИЗИКА».

Поставленная задача распадается на три последовательных этапа:

1. Создание качественного и исчерпывающего по объёму учебного контента, соответствующего требованиям, предъявляемым к учебному процессу;
2. Создание на основе этого контента программных средств его представления, адаптированных под возможности его представления на мобильных устройствах;
3. Размещение программных средств, обеспечение доступа к ним и техническая поддержка их функционирования и актулизации.

1. Контент

Отправной точкой работы является линейка учебников по физике для общеобразовательной школы, созданная творческим коллективом под руководством А.В. Грачева (физфак МГУ) и классно-урочное планирование к этим учебникам. Контент из данных учебников используется нами с согласия авторов. К каждому уроку создавался набор постеров, включающих «теорию» и «кейсы», а также дополнительные справочные материалы. Теория дается в максимально сжатом (концентрированном) виде, а кейсы представляют полный разбор простейших ситуаций с использованием стандартного набора невербальных средств представления учебного контента, позволяющих сформировать в сознании обучаемых полный, яркий, цветной и многоплановый образ изучаемых процессов и явлений. Такое представление позволяет создать осознанную мотивацию путем включения когнитивных возможностей индивидов в ходе их познавательной активности.

Материал был представлен в лаборатории «Дидактики естественных наук»Институт стратегии развития образования РАО, был обсужден с ведущими преподавателями кафедры «Общая физика» физического факультета МГУ, получил высокую оценку специалистов и право на распространение с использованием имени «физфак МГУ». Учителя физики и школьники, имевшие возможность работать с данным приложением, единодушны во мнении, что подобные инструменты обучения являются очень «нужными и полезными».

В настоящий момент создано приложение по разделу «Механика», (127 постеров), включающая 55 постеров теории, 51 кейсов, 19 справочных материалов. Планируется до конца 2018 года выставить весь курс школьной физики.

2. Программные средства

В создании программных средств участвуют студенты 1 курса АлтГТУ им. И.И. Ползунова. Эти студенты не имеют достаточных знаний и опыта создания мобильных предложений. Поэтому создание программных средств без специально созданных для этого инструментов, является принципиально невозможной. Поэтому было принято решение «писать» приложения под операционные системы Androidи iOS с помощью «конструкторов приложений» (КП). КП — это специальные сервисы, которые позволяют создавать приложения без использования программирования на стандартныу х языках программирования.

Существуют два основных вида конструкторов приложений: платные и бесплатные. Разница между ними в том, что в платных, как правило, намного больше функциональных возможностей представления контента (использование видео, аудио, мультимедиа) и интерактивности (навигация по приложению, обратная связь, включения тестовых заданий). Кроме того интерфейс таких конструкторов (а также созданных с помощью их приложений), более понятен и дружелюбен. Достоинством бесплатных КП простота создания приложений и минимальное использование технологических средств представления контента, позволяющие, с одной стороны, создать программных продуктов в короткий срок, с другой

стороны, минимальный срок освоение этих срок пользователями. У будущих пользователей также могут быть различные цели. Наиболее востребованной является быстрое информирование (справочник) о требуем контенте (до или во время урока) – конкретный, четкий и правильный ответ на конкретно стоящий запрос. Для достижения этой цели достаточно инструментария бесплатных КП. Для учителей и школьников, желающих изучить глубже учебный материал должны быть доступны большие возможности репрезентации контента, которые могут быть использованы только с использованием платных КП. Именно поэтому нами выбраны и те и другие конструкторы приложений.

3. Размещение и доступ к программным средствам

Создание программных средств не может быть конечной целью, которая не может быть иной, как привлечение наибольшего числа потенциальных пользователей. Поэтому учебные материалы создаются нами как мобильные приложения под операционные системы Android и iOS. Причем данные приложения должны быть скачаны с традиционных площадок (AppStore и GooglePlay) и могут быть использованы и использоваться в режиме offline. Не менее важным является информирование возможных пользователей о созданных программных средствах через социальные сети, и создание в них сообществ пользователей. В качестве первой (модельной) социальной сети нами выбран ресурс «В контакте», в которой создано несколько страниц, на которых будущие пользователи могут дать свои комментарии и оценку созданным программным средствам, а также внести свои предложения по их совершенствованию и актуализации.

ИЗВЛЕЧЕНИЕ ИНФОРМАЦИИ О СОБЫТИЯХ ИЗ СТАТЕЙ ВИКИПЕДИИ

Волков Е.А. – студент, Крайванова В.А. – к.ф.-м.н., доцент
Алтайский государственный технический университет (г. Барнаул)

Одним из важнейших направлений развития современных информационных технологий является Text Mining – это извлечение структурированной информации из неструктурированного текста[3]. Ключевыми технологиями в этой отрасли являются Named Entity Recognition (NER) – извлечение именованных сущностей/объектов и Information Extraction (IE) – извлечение фактов. Большая часть современных исследований сконцентрирована на извлечении информации из небольших отдельных текстов, например, новостей[1]. Разработка систем комплексного анализа больших коллекций связанных текстов выведет область Text Mining на новый уровень. Целью исследования является создание алгоритмов извлечения событий из статей Википедии на русском языке, с последующим построением исторических цепочек на основании извлеченных данных.

Для начала определим, что подразумевается под событием. Событие – это факт, имеющий конкретную привязку ко времени. Например, в таком предложении «Великая Отечественная Война началась 22 июня 1941 года.» фактом будет являться начало Великой Отечественной Войны, а датой - 22 июня 1941 года. В формальной модели текста событие представляет собой фрейм со следующим набором полей: дата начала, дата завершения, название, место события, действующие лица, дополнительная информация. Для некоторых полей этого фрейма в тексте может не содержаться соответствующей информации. Даты представляют собой достаточно строгие конструкции (рассматриваются абсолютные текстовые даты, содержащие в явном виде все необходимые для идентификации календарного интервала значения, например, 01.04.2001, 1 апреля 2000 года, весной 2001 года [2]), поэтому их извлечение может быть основано на применении регулярных выражений. Место события и действующие лица являются именованными сущностями (ИС).

Под именованной сущностью понимается слово или словосочетание, предназначенное для конкретного, вполне определённого предмета или явления, выделяющее этот предмет или явление из ряда однотипных предметов или явлений[1]. Бывают различные виды именованных сущностей: организация, название произведения и др. Анализ отношений между именованными сущностями и другими элементами предложений позволяет выделить из текста некоторый факт, а последующая проверка связи данного факта с датой определяет, является ли данная пара событием. Кроме того, сами именованные сущности, извлечённые из текста статьи, могут дать дополнительную информацию о событии: действующие лица, местоположение.

Рассмотрим существующие методы извлечения именованных сущностей и фактов из текста. Первый метод извлечения информации из текста основан на использовании онтологий[5]. Онтология или концептуальный словарь – это структура в которой описываются некоторые понятия и объекты, отношения между ними, а также их характеристики. Выделяют три типа онтологий: универсальные, отраслевые и узкоспециализированные. Также их можно разделить по содержимому на онтологии объектов (база данных) или онтологии концептов (база понятий). Например, к онтологиям относят Википедию, Dbpedia, Imdb, SUMO, DOLCE. При таком подходе получается высокая точность при извлечении сущностей из текста, а также высокая степень разрешения омонимия, но при этом извлекается только то, что есть в онтологии, а обновление онтологии приводит к тому, что необходимо проводить повторный анализ, либо разрабатывать какие-то методы для автоматического обновления. Поэтому такой подход хорошо использовать для закрытых классов, например, географических названий, или где производится оперативное пополнение источников, например, на ресурсах подобным сайту Кинопоиск.

Подход основанный на правилах использует полный или частичный синтаксический анализ и онтологию категорий[5], т.е. для сущностей указывается набор параметров, которыми она может обладать. Для данного подхода существует большое количество словарей по различным темам, которые используются в шаблонах отношений. Одним из инструментов использующим данный подход является Томита-парсер, который также применяется нами.

Другой подход к извлечению ИС и фактов текста может решаться при помощи машинного обучения. Для этого выбирается алгоритм, способный обучаться, и строится обучающая выборка для этого алгоритма. Затем алгоритм обучают и в зависимости от результатов его работы на тестовой выборке либо проводится дообучение алгоритма, либо начинается его использование[5]. При таком подходе не требуется большого количества ручного труда по написанию правил, нет необходимости в заранее подготовленной онтологии и в детальном описании каждого контекста. Положительными сторонами являются простота перенастройки системы под другие «стили» языка и возможность итеративного усложнения извлекаемой информации. Из минусов: недостаточно развиты инструменты автоматического обработки, потребность в большом обучающем корпусе, который верно и полностью размечен, а также проблема в отслеживании места возникновения ошибки.

В исследовании была произведена обработка информации из статей Википедии при помощи Томита-парсера, также Википедия была использована в качестве онтологии для определенных событий. В планах применить машинное обучение для улучшения результатов полученной модели.

Приложение состоит из двух модулей: модуля парсинга Википедии и извлечения событий и модуля REST-сервиса. Модуль парсинга содержит функции для первичной подготовки текста статьи и конфигурируемые алгоритмы извлечения событий с возможностью анализа качества их работы. Извлеченные события со всей сопутствующей информацией сохраняются в базе данных. Модуль REST-сервиса реализует возможность

доступа к хранилищу извлеченных событий с помощью REST-запросов. Сервисом обрабатываются три типа запросов:

1) получение информации о событии: {id, start_date, end_date, name}.
2) получение по идентификатору события связанных с ним событий: {reason: [{id, start_date, end_date, name}], result: [{id, start_date, end_date, name}]}, где reason – массив причин, result – массив следствий.
3) получение вложенных событий: {nested: [{id, start_date, end_date, name}]}, где nested – массив вложенных событий.

На данный момент ведётся реализация описанного функционала в виде программного комплекса, позволяющего строить цепочки исторических событий для большей части статей Википедии.

Список литературы

1. Сулейманова Е.А. Семантический анализ контекстных дат, Программные системы: теория и приложения, 2015, том 6, выпуск 4, 367–399.
2. Инструкция по определению именованных сущностей. // OpenCorpora Wiki URL: http://opencorpora.org/wiki/Nermanual/1 (дата обращения: 16.04.2017).
3. Основные технологии text mining. // DataReview.info URL: datareview.info/article/osnovnyie-tehnologii-text-mining/ (дата обращения: 16.04.2017).
4. Fact Extraction (ideograph) // Share and Discover Knowledge on LinkedIn SlideShare URL: https://www.slideshare.net/Tatiana.lando/fact-extraction-ideograph (дата обращения: 16.04.2017).
5. Извлечение объектов и фактов из текстов // Share and Discover Knowledge on LinkedIn SlideShare URL: https://www.slideshare.net/yandex/ss-28635572 (дата обращения: 16.04.2017).

ЗАДАЧА ПАРСИНГА «ПРОЗРАЧНОГО» СИНТАКСИСА НА ПРИМЕРЕ ВИКИ-РАЗМЕТКИ

Волков Е.А. – студент, Крайванова В.А. – к.ф.-м.н., доцент
Алтайский государственный технический университет (г. Барнаул)

Большинство формальных языков, с которыми сталкивается программист, имеют очень строгую структуру. Если они и позволяют вставлять в тексты произвольные конструкции, то они должны быть соответствующим образом обособлены (например, кавычками в языках программирования или специальной конструкцией CDATA в XML) и экранированы. При этом общая структура текстов остается строго формализованной. Некоторые языки разметки устроены по другому принципу. Большая часть текста – это просто набор символов, а формальные конструкции достаточно произвольно разбросаны по тексту. Примерами таких языков являются HTML и Markdown. Язык, на котором размечены страницы современной Википедии – это компиляция этих двух языков, которая постоянно дополняется новыми конструкциями.

Грамматика разметки, используемой тем или иным движком, зависит от версии движка и установленных плагинов. Кроме того, некоторые формальные конструкции, например, перенаправления, зависят от применяемой локали. Более-менее полный список формальных конструкции современной вики-разметки можно найти в статье [1]. Элементы страницы могут быть вложены друг в друга, и содержать произвольный текст (в том числе, с

элементами, совпадающими по синтаксису с формальными конструкциями, но ими не являющимися).

Парсер, встроенный в ядро вики-движка, разбирает и преобразует в HTML лишь самые базовые конструкции: теги nowiki, шаблоны, ссылки, заголовки и некоторые другие элементы – а также производит очистку от html[2]. Остальные элементы страницы разбираются специальными плагинами. Полное дерево разбора не строится, осуществляется только перевод конструкций[2]. Оригинальный парсер медиавики имеет сложную многопроходную архитектуру, и для задачи построения дерева объектов неприменим.

Существует множество парсеров, альтернативных тому, что встроен в движок Википедии. Длинный список проектов для различных языков вы можете найти в статье [3]. Большинство этих проектов трансформируют вики-разметку в HTML-разметку. Многие из этих проектов заброшены, не дойдя до стадии первого релиза.

Вместе с тем, Википедия – это популярный источник частично размеченных данных для задач искусственного интеллекта, и потому актуальной задачей является создание удобного и простого мультиязыкового модуля парсинга для вики-разметки. Статьи Википедии отличаются большим разнообразием синтаксиса, гораздо большим, чем обычные html-страницы. В Википедии часто содержатся вставки на языках программирования и различные формулы, которые частично пересекаются с синтаксисом самой вики-разметки.

Разработка парсера «с нуля» для данной задачи не представляется эффективным решением, так как грамматика вики-разметки постоянно развивается, и поддержание парсера в актуальном состоянии затруднительно. Поэтому нами рассмотрено три инструмента, которые позволяют генерировать парсеры на основе контекстно-свободных грамматик.

Так как Python – один из наиболее популярных на сегодняшний день языков программирования[4], активно применяющийся для прототипирования приложений, то в качестве претендента на платформу для разработки был выбран модуль pyparsing[5]. Основное применение этого модуля – разбор простых HTML-шаблонов для извлечения текста. Грамматика собирается из специализированных объектов, классы которых отвечают за различные синтаксические конструкции (ZeroOrMore, QuotedString и другие). Объекты могут быть вложены друг в друга. В процессе разбора осуществляется рекурсивный спуск по функциям этих классов, реализующих собственно парсинг. Библиотека pyparsing предоставляет гибкий инструмент описания КС-грамматик, с дополнительным «сахаром» в виде конструкций, заключенных в кавычки, вложенных скобочных конструкций для рекурсивного разбора и обработчиков событий разбора. Кроме того, парсер может работать в режиме не только полного разбора текста, но и поиска формальных конструкций внутри произвольного текста. Однако, парсинг произвольного текста внутри формальных конструкций (в частности, внутри шаблонов), требует создания очень сложного описания. Архитектура парсера (рекурсивный спуск) может вызвать серьезные проблемы с производительностью при разборе большого количества документов Википедии.

Вторым претендентом на платформу для парсинга стала разработка Яндекса Томита-парсер[6], основанная на GLR-разборе по недетерменированным и неоднозначным грамматикам. Данный инструмент предназначен для задач анализа текстов на естественном языке и извлечения из них фактов. Инструмент содержит возможности учета морфологии и сегментации на предложения, а также позволяет описывать поля извлекаемых фреймов прямо в грамматике, но для формирования терминалов с произвольным текстом требует больших дополнительных усилий. В частности, так как Томита рассчитана на естественный язык, разбор происходит по предложениям, поэтому пригодность данного инструмента для разбора формальных конструкций низкая.

Третьим претендентом на инструмент для задачи разбора вики-разметки выбран ANTLR [7] – генератор нисходящих парсеров для КС-свободных грамматик. Данный инструмент предназначен для генерации синтаксических анализаторов для различных языков программирования: python, java, c++ и другие. ANTLR требует подробного описания

лексического уровня грамматики. Поэтому для создания парсера нами был проведен анализ символов, встречающихся в Википедии. В статьях Википедии содержится 22634 различных Unicode-символа. Такое разнообразие вызвано использованием китайских иероглифов и японской слоговой азбуки в некоторых статьях. Большинство символов встречаются очень редко, но для создания качественного парсера необходимо учесть их все. На данный момент учтены кириллические и латинские символы, а также знаки, имеющиеся на стандартной клавиатуре.

В результате работы проанализированы три платформы для создания парсеров: pyparsing, Томита и ANTLR; создана ANTLR-грамматика, охватывающая большое количество синтаксических конструкций вики-разметки; получен полный список всех символов Википедии для реализации прозрачного пропуска конструкций. Исходный код проекта доступен по адресу[8]. Платформа ANTLR позволяет легко пополнять созданную грамматику, а также генерировать парсеры под различные языки программирования, что придает разработанной грамматике высокую практическую значимость.

Список литературы

1. Help:Wiki markup [Электронный ресурс]. // Wikipedia – Режим доступа: https://en.wikipedia.org/wiki/Help:Wiki_markup#Text_formatting, свободный.
2. Mediawiki/includes/parser/Parser.php [Электронный ресурс]. // Phabricator – Режим доступа:https://phabricator.wikimedia.org/diffusion/MW/browse/master/includes/parser/Parser.php, свободный.
3. Alternative parsers [Электронный ресурс]. // MediaWiki – Режим доступа: https://www.mediawiki.org/wiki/Alternative_parsers, свободный.
4. PYPL PopularitY of Programming Language [Электронный ресурс]. // Github – Режим доступа: http://pypl.github.io/PYPL.html, свободный.
5. Pyparsing Wiki Home [Электронный ресурс]. – Режим доступа: http://pyparsing.wikispaces.com/, свободный.
6. Томита-парсер [Электронный ресурс]. // Технологии Яндекса – Режим доступа: https://tech.yandex.ru/tomita/, свободный.
7. Официальная страница проекта ANTLR [Электронный ресурс]. – Режим доступа: http://www.antlr.org/, свободный.
8. Репозиторий WikiParser [Электронный ресурс]. // GitHub – Режим доступа: https://github.com/egoralvolk/WikiParser, свободный.

ПРОЕКТИРОВАНИЕ СИСТЕМЫ АНАЛИЗА ЗВУКОВОГО ТРАФИКА МОБИЛЬНЫХ РАЗГОВОРОВ

Аксёнов Р.З. – студент, Старолетов С.М. – к.ф.-м.н., доцент
Алтайский государственный технический университет (г. Барнаул)

Анализ звукового трафика – извлечение сведений посредством анализа звуковой информации на основе наблюдения за потоками трафика (наличие, отсутствие, объем, направление и частота). С развитием информационных технологий, искусственного интеллекта, распознавания речи становится возможным реализовывать системы анализа содержимого разговоров.

Такой анализ позволит осуществлять контроль сотрудников организаций, ведущих разговоры, нарушающие корпоративную тайну, разговоров на экстремистские темы, на темы кражи имущества и т.д.; проводить родительский контроль за детьми.

В работе рассматривается проектируемая функциональность своего приложения по анализу трафика.

Анализ звукового трафика мобильного телефона проходит по следующему принципу (рисунок 1) - сначала идет запись разговора, которая декодируется необходимым кодеком. Следующим этапом идет разбиение записанного разговора на части. Все записи сохраняются на устройстве, на которое записано приложение, и, чтобы учитывать адреса записей, существует база данных, которая включает в себя адрес записи и флаг выполнения сканирования. Далее идет сканирование фрагментов записи, которое осуществляется путем отправки записи на удаленный сервер, в данном случае, Google, после чего Google возвращает распознанную речь в виде текста. Если на момент записи и разбиения на фрагменты был отключен интернет, программа автоматически начнет синхронизировать и отправлять файлы на Google при следующем включении интернета. Отправка и получения результатов записи происходит асинхронно и не затрачивает ресурсов мобильного устройства. Следующим этапом является анализ файла на содержание слов, опасных для бизнеса или для жизнедеятельности людей, т.е. содержащих экстремистский уклон, призыв к насилию, религиозную ненависть и т.д. Помимо этого, пользователям будет представлен соответствующий интерфейс для добавления дополнительных слов в общую библиотеку.

Каждая библиотека слов состоит из записей, в которую включается «опасное слово» и его приоритет. Приоритет необходим для суммирования результатов анализа по различным библиотекам слов. Записи, в которых были найдены слова из запрещенных библиотек, имеют высокий приоритет и будут отображаться красным цветом, желтым будут отображаться записи, имеющие средний приоритет вхождения опасных слов, ну и зеленым - записи, в которых не было найдено ни одного запрещенного слова или выражения.

Приложение для мониторинга звукового трафика будет решать следующие задачи:

- мониторинг голосового трафика сотрудника компании или ребенка, использующего данного приложение (с его согласия);
- отслеживание длительности разговоров с конкретными номерами, суммарной длительности, средней длительности разговоров, максимальной длительности, за заданные интервалы времени;
- задание списка слов или словосочетаний, которые представляют интерес;
- определение нахождения слов или сочетаний из списка в разговорах, ведение списка разговоров, содержащих заданные слова, с указанием времени, длительности, номера звонящего и встреченных слов, по возможности – текстовую транслитерацию звонка;
- при вводе пароля открывает доступ к настройкам приложения, в которых можно просматривать статистику по звонкам, редактировать параметры анализа голосового трафика, списки слов и словосочетаний для анализа.

Рисунок 1 – Процесс анализа звукового мобильного трафика

Предполагается, что приложение будет работать в двух версиях (пользовательская и корпоративная).

Функционал пользовательской версии:

- запись разговоров;
- анализ записи разговоров и нахождение в нем слов из библиотек слов;
- возможность включения/отключения отображения номера телефона/даты звонка/вид вызова (Исходящий, Входящий).

Функционал корпоративной версии:

- запись разговоров;
- анализ записи разговоров и нахождение в нем слов из библиотек слов;
- добавление своих библиотек слов;
- отправка на сервер файлов записи и их анализа;
- возможность выбора названия файлов записи;
- возможность включения/отключения режима записи;
- возможность включения/отключения отображения номера телефона/даты звонка/вид вызова (Исходящий, Входящий).

Приложение реализуется под ОС Android на языке Java. Используется Google Cloud Speech API [1] для распознавания речи. В данном случае данный сервис выбран из-за простоты и необходимости получения быстрого работающего прототипа. В дальнейшем, данный сервис может быть легко изменен, благодаря использованию паттерна проектирования «мост»[2], который отделяет абстракцию распознания речи от ее конкретной реализации, также могут быть использованы различные API ключи с использованием паттерна «приспособленец»[2].

Список литературы

1. Google Cloud Speech API Documentation. Режим доступа: https://cloud.google.com/speech/docs/
2. Крючкова Е.Н., Старолетов С.М. Архитектурное проектирование и паттерны программирования [Электронный ресурс]: Учебно-методическое пособие.— Электрон. дан.— Барнаул: АлтГТУ, 2015.— Режим доступа: http://new.elib.altstu.ru/eum/download/pm/Krutkova-Patterns.pdf, авторизованный

ONLINE ПОРТАЛ ДЛЯ МОДЕЛИРОВАНИЯ И ВЕРИФИКАЦИИ РАСПРЕДЕЛЕННОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Ложкина Д.Д. – студент, Старолетов С.М. – к.ф.-м.н., доцент
Алтайский государственный технический университет (г. Барнаул)

Актуальность

Поскольку сложность разрабатываемого программного обеспечения постоянно растет, задача проверки корректности его работы усложняется, эта проверка особенно важно для программ, от которых зависят жизнь и здоровье людей. Наиболее очевидным способом проверки является запуск продукта на некоторых наборах тестов с целью выявления программных ошибок и неточности работы разработанного программного обеспечения. Тестирование программного обеспечения широко распространено, но, очевидно, что такой подход не может доказать правильность работы во всех возможных случаях.

Одним из альтернативных подходов улучшения качества продукта является формальная верификация [1], позволяющая фактически доказать (или опровергнуть) тот факт, что модель программной системы удовлетворяет формальным требованиям, заданным разработчиком. Этот процесс автоматизирован и обнаруживает те ошибки, которые зачастую трудно воспроизвести. Для выполнения процесса нужно построить адекватную модель системы и выразить подлежащие проверке свойства на языке логики, в частности, в виде формул темпоральной логики линейного времени (LTL - Linear-Time Logic) [2]. Поэтому, несмотря на полную автоматизацию процесса верификации, его выполнение требует работы специалиста достаточно высокой квалификации.

Проблема

Сегодня существует большое количество средств для верификации, но они не получили широкого распространения среди программистов: эти средства сложны, а оболочки для них неудобны. SPIN (Simple Promela Interpreter [3]) – одна из утилит для верификации корректности распределенных программных моделей. Системы, подлежащие верификации должны быть изложены на языке Promela (Process Meta Language), а свойства, которые требуются проверить, выражаются как LTL формулы.

Необходимость описать модель системы на специальном языке и сформулировать требования на языке темпоральной логики является препятствием для разработчиков. Зачастую перевод требований, выраженных на естественном языке, в LTL формулы представляет сложную задачу.

Например, система должна удовлетворять следующему свойству [4]: всегда, если p стало истинным, то когда-нибудь в будущем станет истинным q, а p будет оставаться истинным до тех пор, пока q не станет истинным. Такое выражение переводится в следующую

темпоральную формулу: *G (p -> (p U q))*, где *G* - темпоральный оператор «Globally» («Всегда») и *U* – темпоральный оператор «Until» («До тех пор, пока»).

Многие формулы могут быть очень длинными и сложными, даже специалисты в этой области не могут выразить их без затруднений, кроме того, понять, какое требование представляла формула ранее, непросто.

Также возникает проблема хранения всех заданных требований в одном месте, так чтобы всегда можно было вернуться к некоторому требованию, просматривая его формулу или описание.

Предлагаемое средство для решения задачи

Для решения данной проблемы нами был разработан web-портал для построения модели системы и требований к ней с использованием графических инструментов, для схем алгоритмов и диаграмм требований нами были разработаны специальные обозначения. Мы считаем, что такой подход поможет разработчикам легко задавать набор требований и строить модели системы таким же образом, как обычно строят всем известные блок-схемы алгоритмов. Использование online приложения избавит пользователей от необходимости установки дополнительного ПО на компьютер, при этом процесс верификации будет выполняться на сервере: пользователь строит схему модели системы и требований, они автоматически переводятся в исполняемый код на языке Promela и LTL формулы соответственно, выполняется процесс верификации, и пользователь получает результат.

Обозначение логических и темпоральных операторов

Для наилучшего понимания LTL формул мы используем графические обозначения, отличные от существующего подхода [5].

Мы представляем состояние системы, как сочетание определенных значений глобальных переменных системы. Каждое состояние системы обозначено в виде блока, имеющего свое имя и соединенного с другими состояниями с помощью темпоральных операторов.

Все операторы представлены в виде круга, содержащего специальный символ (см. Таблицу 1).

Таблица 1 – темпоральные операторы и их обозначения

Оператор	**LTL формула**	**Синтаксис Promela**	**Обозначение**
Состояние	State	State	State
И	State1 $\wedge$ State2	State1 && State2	State 1 A State 2
ИЛИ	State1 $\vee$ State2	State1 || State2	State 1 O State 2
Исключающее ИЛИ	State1 xor State2	State1 ^^ State2	State 1 X State 2

Импликация	State1 → State2	State1 -> State2	State 1 → State 2
Тождество	State1 ≡ State2	State1 <-> State2	State 1 ≡ State 2
Отрицание	¬ State1	!State1	¬ State
Globally	G State1	[]State1	∞ State
Finally	F State1	<>State1	✳ State
Until	State1 ∪ State2	State1 U State2	State 1 ▷ State 2
Скобки	(State1 ∧ State2)	(State1 && State2)	State 1 A State 2

Для стандартных логических операторов используется соответствующая заглавная буква (**A** для *And* («И»), **O** для *Or* («Или»), **X** для *Xor* («Исключающее ИЛИ»)) или специальный символ (для *Импликации, Отрицания* и *Тождества*). Поскольку наша цель - разработать обозначения темпоральных операторов как можно более интуитивными и понятными для пользователя, мы предлагаем следующие обозначения:

- *Globally* («Всегда») изображен в виде знака бесконечности, наиболее подходящая интерпретация состояния, которое должно выполняться на протяжении всего времени работы системы.
- *Finally* («Когда-нибудь») – состояние когда-нибудь в будущем будет истинно, обозначено в виде звездочки, как признак нового события.
- *Until* («Пока») представлено в виде треугольника, касающегося своей вершиной прямой линии, как факт того, что одно состояние должно выполниться когда-нибудь в будущем, а другое обязано выполняться во всех состояниях до обозначенного.
- *Скобк*и необходимы для обозначения порядка операторов в сложных формулах, поэтому обозначены как контур прямоугольника, который можно описать вокруг любой последовательности состояний. Таким образом мы можем переводить сложные формулы с большой глубиной вложенности.

Возможности разработанного портала

У пользователя есть возможность создавать аккаунт, добавлять, редактировать и хранить проекты. Подготовка к процессу верификации состоит из двух шагов.

Для начала пользователь должен описать модель своей системы: объявить системные переменные, создать процессы (имеется возможность работы с несколькими процессами) и каналы для их взаимодействия. В распоряжении пользователя несколько инструментов.

Верхняя часть страницы состоит из четырех панелей, которые могут быть скрыты по желанию пользователя для более удобной работы (см. Рисунок 1).

1) *Панель проекта* предназначена для изменения свойств проекта (название, описание и т.д.)

2) *Панель глобальных переменных* позволяет добавлять и редактировать переменные в следующем формате: тип переменной (выбирается из предложенного списка), имя, начальное значение и описание. Здесь же можно просмотреть список всех созданных переменных. Внешний вид этой панели используется как шаблон для всего приложения.
3) *Панель каналов* служит для создания каналов для взаимодействующих процессов и содержит следующие параметры: имя канала, размер буфера сообщений и набор возможных типов.
4) *Панель типов сообщений* инициализирует тип mtype, используемый в синтаксисе Promela (аналог перечисления в языках программирования высокого уровня).

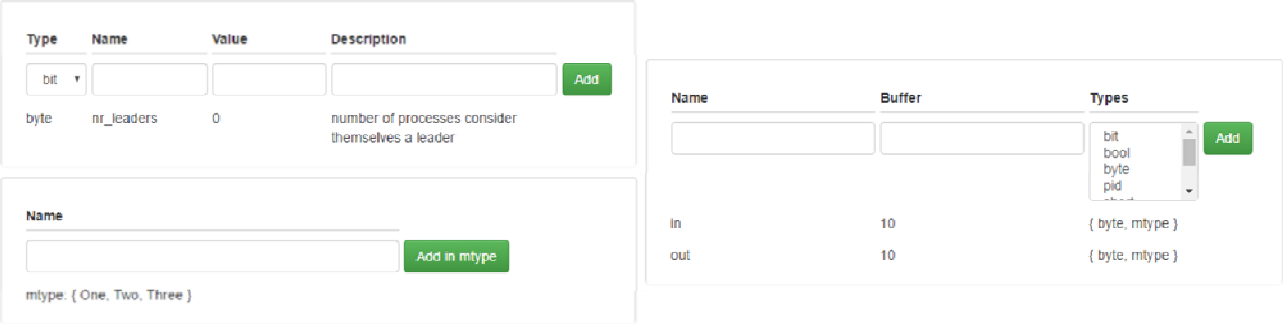


Рисунок 1 – Панели глобальных переменных, каналов и типов сообщений

Главная панель (*Панель процессов*) представляет собой поле со вкладками, где каждая отдельная вкладка содержит модель определенного процесса: поле для построения схемы, палитра графических объектов, текстовые поля для имени и описания, а также панель для локальных переменных. Пользователь перетаскивает объекты с палитры на графическое поле и устанавливает связи между ними с помощью мыши. Благодаря такому решению, пользователь может работать с несколькими процессами одновременно.

Набор графических объектов для построения схемы состоит из общепринятых фигур для построения блок-схем, а также специальных обозначений для отправки/получения сообщений, запуска и ожидания процесса (см. Рисунок 2).

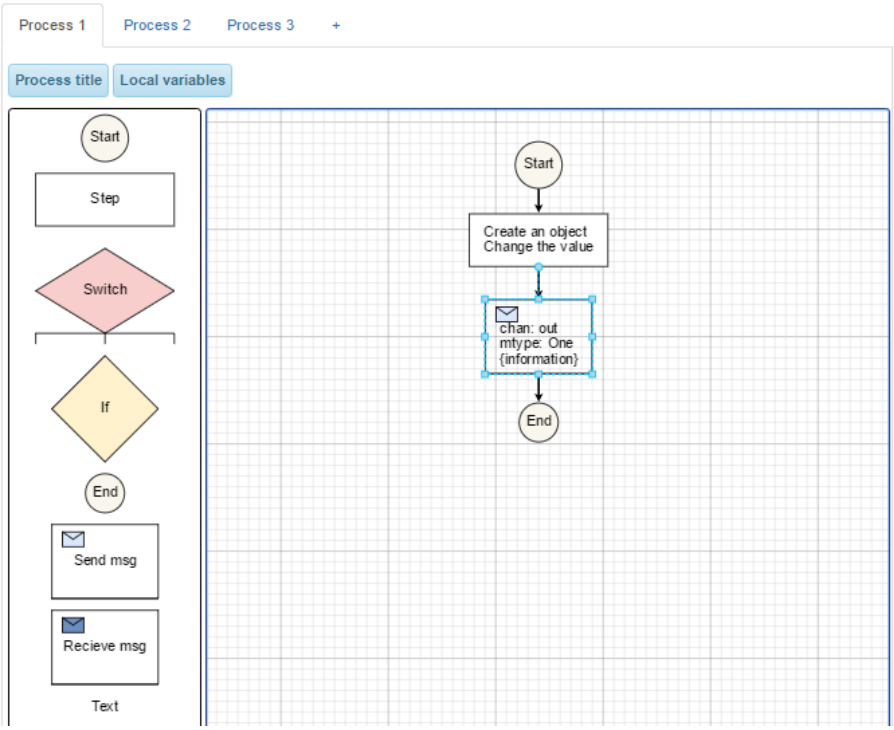


Рисунок 2 – Панель Процессов

Как только пользователь закончил редактирование модели и задал все нужные переменные, автоматически генерируется исполняемый Promela код для последующей верификации.

Второй шаг – задание требований к системе.

Верхняя панель содержит поле со вкладками, содержащими описание состояния системы: значение глобальной переменной и комментарий пользователя. Здесь пользователю доступны все созданные им на предыдущем шаге глобальные переменные и определенный набор операторов для задания значения.

Главная панель похожа на панель процессов, описанную выше: содержит такое же поле для построения схемы, поля для описания и имени, но две палитры объектов (см. Рисунок 3):

1) Графические обозначения операторов.
2) Созданные пользователем состояния системы.

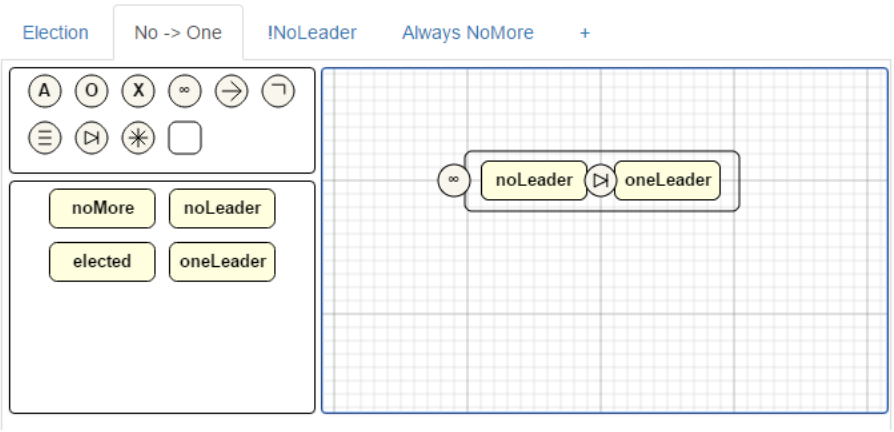


Рисунок 3 – Главная панель моделирования требований

Построенную схему требования мы легко можем преобразовать в LTL формулу в соответствии с введенными обозначениями.

На нижней панели находится список созданных системных требований, где пользователю доступны также поля с LTL формулой и изображением диаграммы (см. Рисунок 4).

System Requirements:

Name	Definition	Description	LTL Formula	Diagram
Election	At least once in the future	Election eventually has to hold	<> elected	elected
No -> One	NoLeader until OneLeader	NoLeader has to hold at least until OneLeader	[] (noLeader U oneLeader)	noLeader oneLeader
!NoLeader	NoLeader is always false	NoLeader doesn't have to hold	![] noLeader	noLeader
Always NoMore	NoMore is always true	NoMore has to hold	[] noMore	noMore

Рисунок 4 – Список созданных системных требований

После построения модели системы, генерации исполняемого кода, выражения требований и их перевода в язык логики LTL можно выполнять верификацию, результатом которой будет положительный вердикт (система удовлетворяет требованиям) или контрпример.

Практическое применение

В предыдущем исследовании мы разработали чат с использованием распределенного алгоритма выбора лидера для решения задачи надежной коммуникации [6]. Проблема выбора лидера состоит в следующем: пусть в системе асинхронно взаимодействуют несколько процессов, имеющих определенный вес (размер журнала, уникальный номер и т.п.); необходимо выбрать один «управляющий» процесс, координирующий работу всей системы. Мы рассмотрели множество известных алгоритмов, такие как Paxos [7], Raft [8], протокол выбора лидера в кольце [9] и др. Поскольку алгоритмы разработаны для распределенных систем, они могут обладать потенциальными ошибками во взаимодействии, и их работа должна быть верифицирована.

Мы можем представить пример построения модели и требований для *Алгоритма выбора лидера в однонаправленном кольце*. Для начала необходимо построить модель системы и сгенерировать код на языке Promela. На протяжении работы алгоритма, каждый участник выполняет один процесс, обновляющий текущее значение параметра. Мы объявили глобальные переменные, каналы и типы сообщений (см. Рисунок 1). Также мы изобразили схему алгоритма с помощью инструмента для построения диаграмм (см. Рисунок 5).

Следующий шаг – задание требований (см. Рисунок 4):

1. Лидер обязательно будет выбран.
2. В системе нет лидера только до того момента, пока не будет выбран один лидер.
3. В системе всегда не более одного лидера.
4. Невозможен выбор более одного лидера.

Теперь можно провести верификацию и убедиться в корректности алгоритма без установки дополнительного ПО и знания специальных языков.

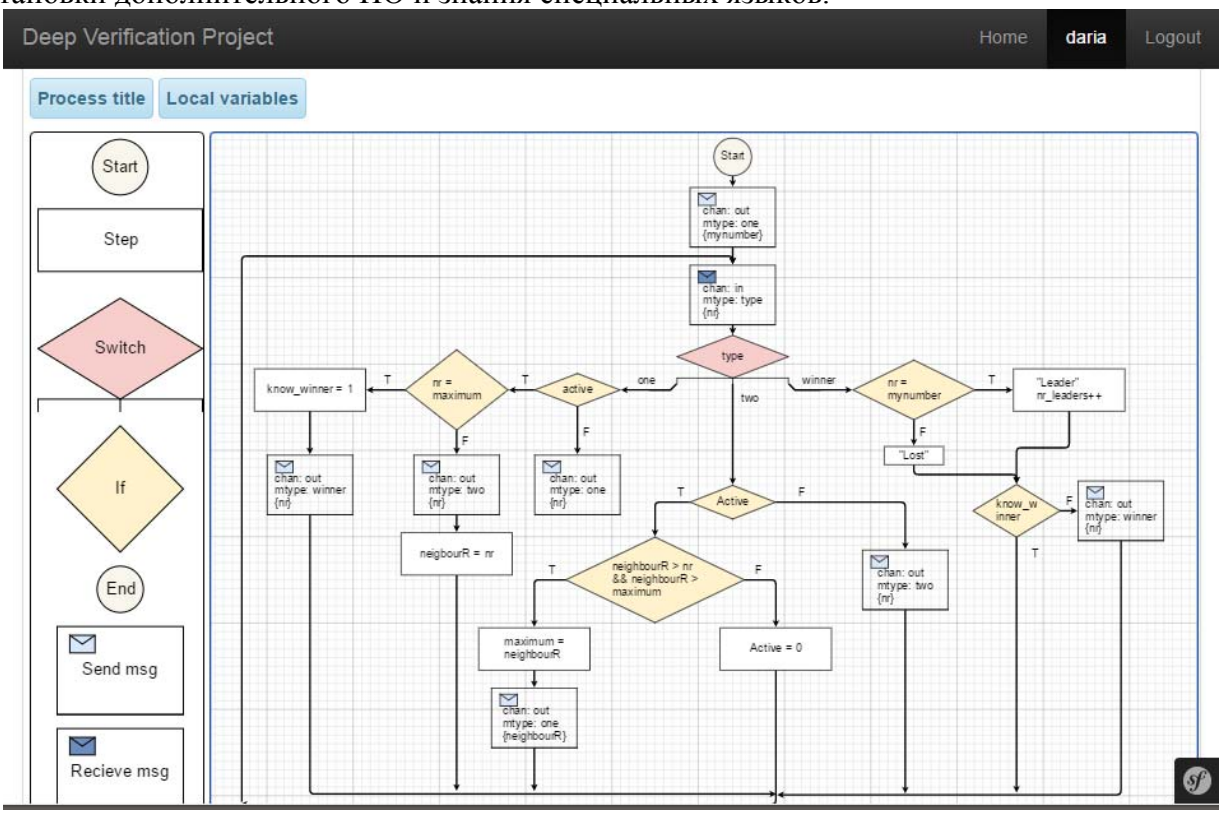


Рисунок 5 – Схема алгоритма выбора лидера

Особенности реализации и дальнейшая работа

Описанное приложение создано на языках PHP и JavaScript с использованием MVC фреймворка Symfony. Генератор кода для последующей верификации находится в разработке. В дальнейшем мы собираемся провести дополнительные исследования с целью улучшения нашего приложения, добавив:

- Шаблоны для LTL формул [10].
- Примеры проектов для известных алгоритмов.
- Возможность генерировать не только Promela код, но и код на языке Erlang [11].
- Представление нескольких процессов в одном окне.
- Извлечение требований, представленных в виде текста на естественном языке.

Заключение

Разработанное приложение позволяет широкому кругу пользователей, не являющимися специалистами в сфере верификации, создавать модели распределенных взаимодействующих систем и проверять их работу на корректность, таким образом повышая их надежность.

Список литературы

1. Карпов Ю.Г. Model checking. Верификация параллельных и распределенных программных систем. – BHV, 2010. – 560 с.
2. Linear-Time Temporal Logic [электронный ресурс]: Режим доступа: http://wwwstep.stanford.edu/tutorial/temporal-logic/temporal-logic.html
3. Spin - Formal Verification [электронный ресурс]: Режим доступа: http://spinroot.com
4. Шошмина И.В., Карпов Ю.Г. Введение в язык Promela и систему комплексной верификации Spin. – Санкт-Петербург, 2010. – 110 с.
5. Brambilla M., Deutsch A., Sui L., Vianu V. (2005) The Role of Visual Tools in a Web Application Design and Verification Framework: A Visual Notation for LTL Formulae, - Proc. 5th Int. Conf. on Web Engineering (ICWE 2005), Sydney, Australia, July 27-29, 2005. - C. 557.
6. Ложкина Д.Д., Старолетов С.М. Применение распределенного алгоритма выбора лидера для задачи надежной коммуникации. – Барнаул : АлтГТУ, 2016. – С. 19. Режим доступа: http://edu.secna.ru/media/f/pi2016.pdf
7. Lamport, Leslie. Paxos Made Simple. - ACM SIGACT News (Distributed Computing Column), 2001. – C. 51.
8. Ongaro D., Ousterhout J. In Search of an Understandable Consensus Algorithm. Stanford University. – 2014. – C. 18.
9. Attiya H., Welch J. Distributed Computing: Fundamentals, Simulations and Advance Topics. – 2004.
10. Dwyer M., Avrunin G., Corbett J. Property Specification Patterns for Finite-State Verification. - Florida, USA, 1998. - C. 7.
11. Старолетов С.М. Функциональные языки распределённых систем: Учебно-методическое пособие.- Барнаул : АлтГТУ, 2015 - 81 с.

ЦЕЛЕСООБРАЗНОСТЬ ИСПОЛЬЗОВАНИЯ МОДЕЛИ ДЕРЕВЬЕВ ПОВЕДЕНИЯ ПО СРАВНЕНИЮ С АВТОМАТНОЙ МОДЕЛЬЮ ПРИ ПРОГРАММИРОВАНИИ ДЕЙСТВИЙ ИГРОВЫХ ПЕРСОНАЖЕЙ

Пасюта А.А.– студент, Старолетов С.М. – к.ф.-м.н., доцент
Алтайский государственный технический университет (г. Барнаул)

В данной статье предлагается использования модели поведения роботов или игровых персонажей в виде дерева поведения (Behavior Tree). Дерево поведения – это ориентированный ациклический граф, узлами которого являются возможные варианты поведения робота или игрового персонажа. Ширина дерева указывает на возможные действия, которые может совершить объект. Высота его поддеревьев указывает на сложность алгоритма.

Актуальность

На сегодняшний день при разработке моделей поведения используются конечные автоматы. Со временем алгоритмы поведения стали более сложными и по мере их усложнения к конечному автомату стали добавляться все новые и новые элементы модели. При увеличении числа элементов у автомата, возникают проблемы:

- Усложняется отладка программы.
- При увеличении числа состояний, сложность конечного автомата будет резко возрастать. Таким образом, имея количество состояний равное N, у нас будет получаться максимальное количество переходов равное N*(N-1). Например, если в нашем конечном автомате 4 состояния, то в этой ситуации, в конечном итоге, получим 12 возможных переходов. В том случае, если нам необходимо добавить хотя бы одно состояние, то количество переходов изменится и в итоге станет равным 20. Таким образом, при увеличении числа состояний, количество возможных переходов растет в геометрической прогрессии и соответственно существенно усложняется программа.
- При необходимости добавления нового состояния необходимо создавать дополнительные связи и условия для уже существующих элементов модели.
- Усложняет модернизация программы.
- Добавления состояний во время работы программы сложно.

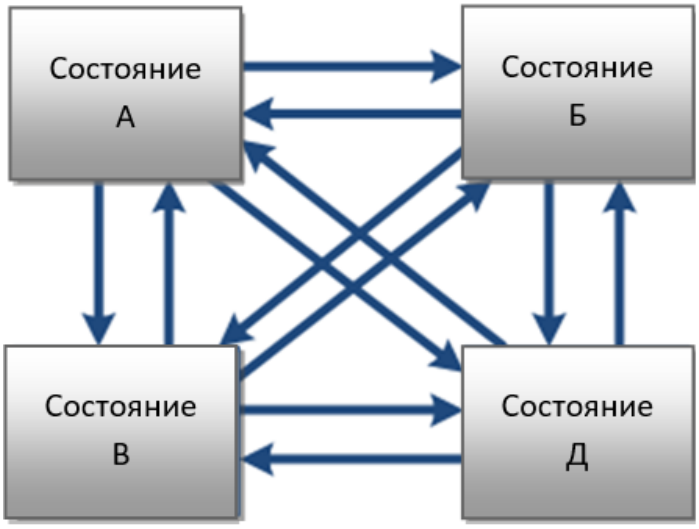


Рисунок 1 - Конечный автомат

В следствии всего этого:

- Увеличивается время разработки приложения.
- Увеличиваются затраты на разработки, поддержку и дальнейшую модернизацию продукта.
- Невозможно добавлять новые состояния в работающее приложение.

В связи с выше перечисленными проблемами следует, что необходима новая модель поведения, которая поможет разработчикам модулей поведения решить поставленные проблемы и упростить написания подобных модулей.

Предлагаемое решение задачи

Для устранения данных недостатков предлагается применять другую методику, так называемые деревья поведения, которые успешно применяются для создания моделей поведения в игровой индустрии. Одно из главных преимуществ деревьев поведения в том, что они имеют более формальную структуру, поэтому их использование упрощает описание поведения объекта. Для каждого состояния в конечном автомате программируется своя собственная логика принятия решений, в то время как в дереве она выведена за пределы состояний. С помощью такого подхода в деревья можно добавлять и удалять новые узлы даже во время выполнения программы. Так же, существует возможность разбить дерево с большим числом состояний на более мелкие поддеревья – это облегчает работу разработчика программы, упрощает работу тестировщика и ускоряет поиск багов в программе.

Вершины в дереве называются задачами или поведениями. Каждая задача имеет четыре состояния:

- «Успех» - задача успешно выполнена;
- «Отсутствие результата» - по какой-то причине задача невыполнима, или не выполнено ее условие;
- «Работа» - задача запущена в работу и ожидает завершения
- «Сбой» - в программе возникла неизвестная ошибка.

Дерево начинается просматриваться всегда с корня и дальше происходит поиск в глубину, начиная с левой ветви дерева. При встречи узла с несколькими подзадачами проверяется сначала левая. Среди узлов выделяют следующие типы: действие (action), селектор (selector), параллельный узел (parallel), узел исполнения последовательности (sequence), инвертор (inverter), условие (condition).

Действие представляет собой запись переменных или какое-либо движение. Узлы последовательностей поочередно исполняют поведения каждого дочернего узла до тех пор, пока один из них не выдаст значение «Отсутствие результата», «Работа» или «Сбой». Если этого не произошло, возвращает значение «Успех».

Узлы параллельных действий исполняют поведения дочерних узлов до тех пор, пока заданное количество из них не вернет статусы «Отсутствие результата» или «Успех».

Селекторы поочередно исполняют поведения каждого дочернего узла до тех пор, пока один из них не выдаст значение «Успех», «Работа» или «Сбой». Если этого не произошло, возвращает значение «Отсутствие результата».

Условия содержат критерий, по которому определяется исход, и переменную. Например, условие «Количества здоровье больше 80%?» и сравнивает значение переменной «Здоровье».

Узлы инверсии выполняют функцию оператора NOT.

Примеры использования

Основными примерами использования деревьев поведения служат создание модели поведения для беспилотных летательных аппаратов. В сфере компьютерных игр Behavior Tree применяются в таких играх как: Halo, Bioshoch, Spore и Sims.

Заключение

Рассмотрев все преимущества модели поведения Behavior Tree, было решено провести исследовательскую работу для наглядной демонстрации всех плюсов данного подхода и реализации расширенной библиотеки для манипулирования деревьями поведения. В качестве демонстрации данной библиотеки предполагается создать игру с персонажами, управляемыми деревьями поведения.

Рисунок 2 - Пример игры на деревьях поведения

Список литературы

1. Colledanchise Michele, and Ögren Petter 2016. How Behavior Trees Modularize Hybrid Control Systems and Generalize Sequential Behavior Compositions, the Subsumption Architecture, and Decision Trees. In IEEE Transactions on Robotics vol.PP2. Isla D. 2005. Handling complexity in the Halo 2 AI. In Game Developers Conference (Vol. 12)
2. Isla D. 2008. Halo 3-building a better battle. In Game Developers Conference. 2008.
3. Lim, C. U., Baumgarten, R., & Colton, S. 2010. Evolving behaviour trees for the commercial game DEFCON. In Applications of Evolutionary Computation, pp. 100-110. Springer Berlin Heidelberg, 2010.
4. Ögren, Petter. "Increasing Modularity of UAV Control Systems using Computer Game BTs." In AIAA Guidance, Navigation and Control Conference, Minneapolis, Minnesota, pp. 13-16. 2012.
5. Colledanchise Michele, Marzinotto Alejandro, and Ögren Petter."Performance Analysis of Stochastic BTs." In Robotics and Automation (ICRA), 2014 IEEE International Conference on. 2014.
6. Marzinotto, Alejandro, Colledanchise Michele, Smith Christian, and Ögren Petter. "Towards a Unified BTs Framework for Robot Control." In Robotics and Automation (ICRA), 2014 IEEE International Conference on. 2014.
7. Klöckner, Andreas. "Interfacing BTs with the World Using Description Logic." In AIAA Guidance, Navigation and Control Conference, Boston, MA. 2013.

ЦЕЛЕСООБРАЗНОСТЬ ПРИМЕНЕНИЯ МЕТОДОЛОГИИ РАЗРАБОТКИ MDD В КОМПАНИИ ПО ПРОИЗВОДСТВУ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Шевелева А.Г. – студент, Старолетов С.М. – к.ф.-м.н., доцент
Алтайский государственный технический университет (г. Барнаул)

В данной статье предлагается методология разработки программного обеспечения MDD – Model Driven Development [1] (разработка, управляемая моделями).

Из опыта работы в компании, занимающейся разработкой стабильных программных продуктов, в которые со временем добавляется новый функционал, со стороны должности QA-инженера, были замечены несовершенства методологий разработки, применяемых на данный момент:

- Разрабатываемый продукт не имеет схемы взаимодействия процессов;
- Часть функционала проекта не описана в документации;
- Не все критические функции покрыты модульными тестами;
- В процесс разработки не включены инженеры по качеству (они работают с готовым продуктом);
- Заказчик не имеет возможности следить за ходом проекта.

Перечисленные выше недочеты приводят к таким проблемам как:

- Проект становится трудно поддерживаемым;
- Отсутствие документации;
- В процесс разработки тяжело ввести нового члена команды;
- Сложности при проведении рефакторинга [2] кода;
- В коде присутствуют ошибки, которые могли быть закрыты при участии в разработке инженера по качеству;
- Удваивается время на проверку валидности выполненной задачи;
- Недовольство клиента из-за несоответствия реализованного продукта его ожиданиям.

Сейчас существует большое количество различных методологий разработки ПО, рассмотрим те, которые покрывают часть затронутых проблем:

- TDD (Test Driven Development) [3] – разработка через тестирование. Данный метод позволяет решить проблемы, связанные с рефакторингом кода, перекрытием ошибок и увеличением времени на проверку валидности.

  TDD представляет из себя следующий подход к реализации (рисунок 1):

  1. Планирование тестов
  2. Написание модульных тестов, пока один из них не будет провален
  3. Производится написание методов, проверяемых написанными ранее тестами
  4. Проверка написанного кода тестами
     - 4.1. Если есть проваленные тесты, то переходим к шагу 3
     - 4.2. Если ошибок в тестах нет, то возвращаемся к шагу 1
  5. Завершение процесса разработки

Благодаря такой методологии разработки весь кода проекта покрыты тестами, что в дальнейшем упрощает его поддержку.

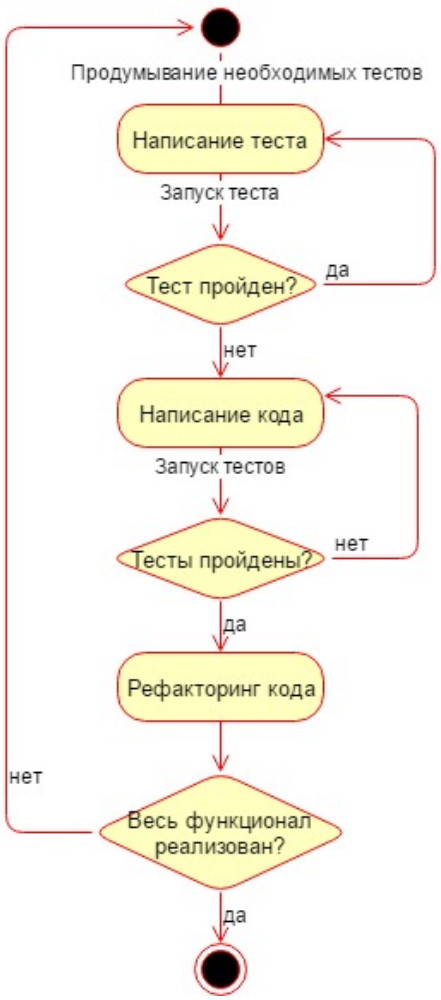


Рисунок 1 – Методология TDD

- BDD (Behavior Driven Development) [4] – разработка через поведенческие спецификации. Используя методологию BDD мы сможем дополнительно, относительно TDD, решить проблему с документированием проекта, а также этот метод позволяет заказчику высказывать свои пожелания до окончательной реализации.

  Методология разработки по BDD придерживается следующего алгоритма (рисунок 2):

  1. Разрабатываются спецификации
  2. Производится генерация кода по спецификации
  3. Написание тестов по спецификации
  4. Написание кода программы, необходимого для прохождения тестов
     - 4.1. Если тесты пройдены, то производится доработка спецификации (переходим на шаг 1)
     - 4.2. Если тесты провалились, то производится рефакторинг кода (переходим на шаг 4)
  5. Завершение процесса разработки.

Рисунок 2 – Методология BDD

- MDD (Model Driven Development) – разработка, основанная на моделях. MDD позволяет создать проект на основе разработанной модели, упрощает разработку и сопровождение реализуемого продукта. Но у данной методологии есть минусы:
  - MDD невозможно внедрить в уже начатый проект, в отличии от методологий TDD и BDD;
  - Методология MDD подходит для больших проектов, которые будут развиваться на протяжении длительного промежутка времени.

При изучении текущего рынка аналогичных продуктов был найден RSA (Rational Software Architect) [5] - инструмент для проектирования и разработки, который увеличивает эффективность MDD за счет использования UML для создания приложений и служб с тщательно проработанной архитектурой.

MDD начинается с анализа задачи и созданием модели будущего продукта, в котором будут участвовать как разработчики, так и инженеры по качеству, что позволит разработать надежную модель. По созданной модели будут сгенерированы классы, тесты и документация. Тесты будут сгенерированы на основе сценариев, отображенных на схеме, что будет позволять представлять заказчику реализованный и протестированный функционал. После генерации шаблона кода разработчики начинают реализовывать сгенерированные методы.

При добавлении в проект нового функционала проблем не возникнет, так как в модель будет добавлен новый элемент, который таким же успешным образом будет реализован, что повышает гибкость разрабатываемой системы. Рефакторинг кода тоже не составит особого труда, так как функционал покрыт тестами и если изменения привнесли ошибки, то система сообщит об этом, что позволит разработчику исправить баг сразу же, без повторного возвращения к этой задаче.

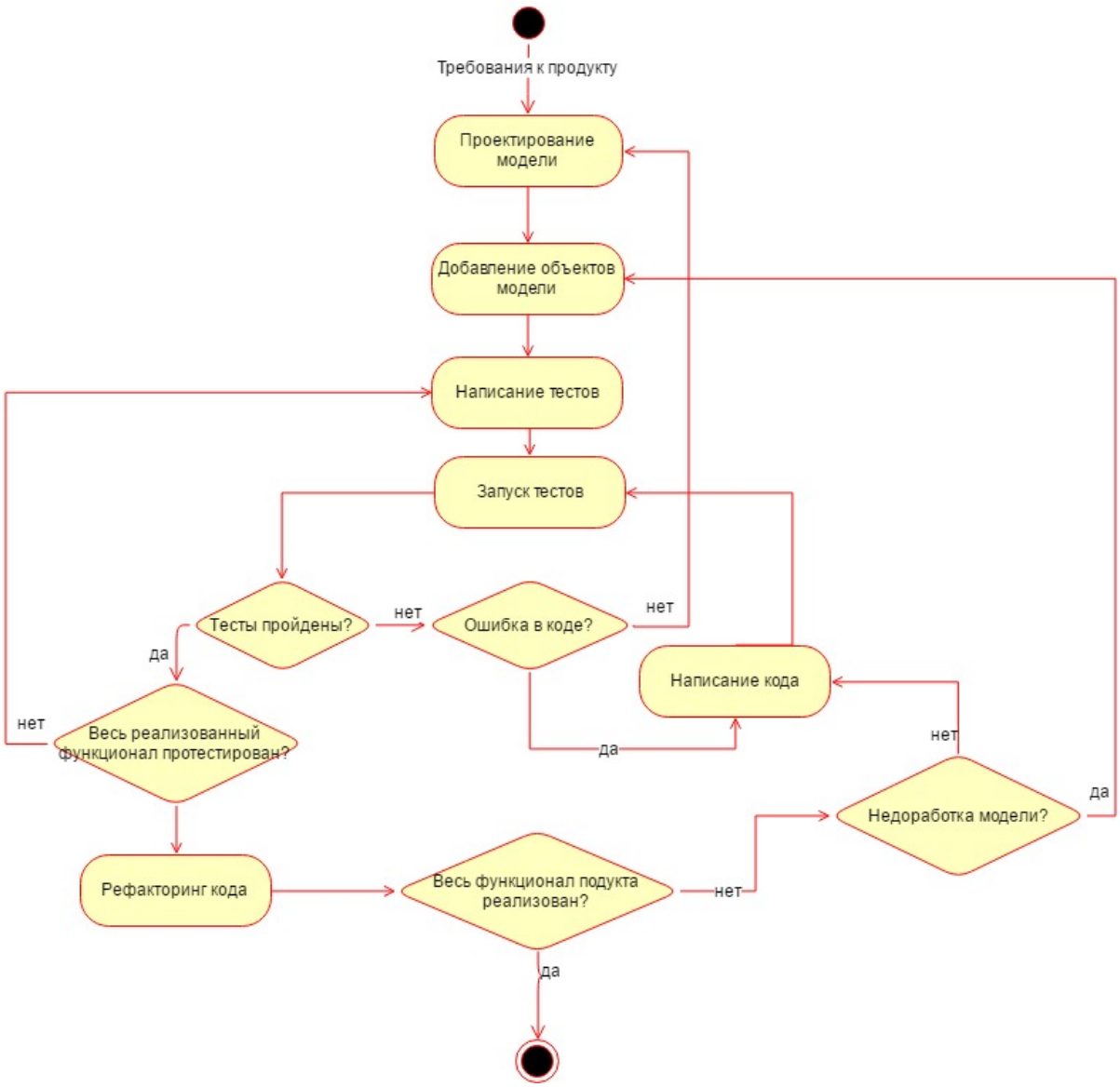


Рисунок 3 – Методология MDD

Основной проблемой этого подхода будет являться реализация функционала, позволяющего производить редактирование уже сгенерированного при изменении, относящейся к нему, модели. Предлагаемый процесс MDD по производству программных продуктов в компании представлен на рисунке 4.

Методология разработки ПО на основе моделей позволит побороть проблемы, указанные выше. Но перейти к разработке по такой модели сложно, поэтому предлагается реализовать приложение, помогающее создавать продукт с использованием данной методологии, что позволит без больших неудобств перейти на разработку по MDD. Данное программное обеспечение будет внедрено в процесс разработки нового проекта компании. Данный проект состоит в разработке нового приложения, которое будет использовать большая часть клиентов. Помимо этого, он будет рассчитан на долговременное использование и должен быть легко редактируемым, что прекрасно позволяет реализовать методология разработки по MDD.

При использовании подхода MDD мы получаем качественный продукт, необходимый бизнесу, в разработке которого будут участвовать менеджеры, разработчики, инженеры по качеству и, непосредственно, сам заказчик. Помимо этого, в разработку данного проекта можно легко ввести нового члена команды, так как присутствует модель разрабатываемого проекта и написана документация.

Рисунок 4 – Предлагаемый процесс MDD

Список литературы

1. Völter M., Stahl T., Bettin J., Haase A., Helsen S. Model-Driven Software Development: Technology, Engineering, Management. Wiley, 2006 - 446 p. Режим доступа - http://www.voelter.de/data/books/mdsd-en.pdf
2. Фаулер М., Бек К., Брант Д., Апдайк У., Робертс Д., Гамма Э. Рефакторинг. Улучшение существующего кода. Символ-Плюс, 2008 – 432 с.
3. Бек К.. Экстремальное программирование: разработка через тестирование. Питер, 2003 – 224с.
4. Wynne M., Hellesoy A., Tooke S. The Cucumber Book, Second Edition. Behaviour-Driven Development for Testers and Developers. The Pragmatic Bookshelf, 2017, 336p. ISBN: 978-1-68050-238-1
5. IBM Rational Software Architect. Режим доступа - https://www.ibm.com/developerworks/downloads/r/architect/

АРХИТЕКТУРА ИНТЕЛЛЕКТУАЛЬНОЙ СИСТЕМЫ ПО СОЗДАНИЮ МУЗЫКАЛЬНЫХ ПРОИЗВЕДЕНИЙ

Козлов Н.С. – студент, Крючкова Е.Н. – к.ф.-м.н., профессор
Алтайский государственный технический университет (г. Барнаул)

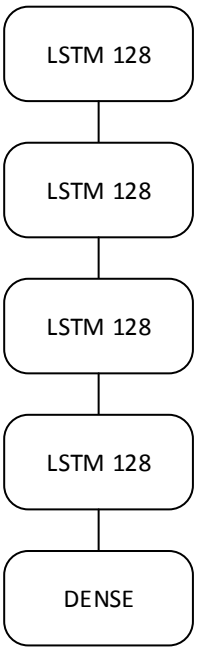
В современном мире высока потребность в «оригинальной» озвучке того или иного продукта, за озвучку фильмов или игр берутся мировые композиторы, а иногда даже и коллективы, происходит сложная и подчас не понятная обычному человеку работа по написанию множества музыкальных произведений, которые подчеркивают аспекты, мысли и идей того, к чему они написаны. Зачастую профессиональные музыканты широко используют музыкальную теорию (которая достаточно строго описывает правила построения и анализа произведений) при написании таких работ, а на полагаются, как это принято в современном непрофессиональном сообществе, на слух. Стоит такая процедура достаточно много, а оценить результаты работы можно только субъективно.

Идея разработки

Идея состоит в том, чтобы повысить доступность качественной музыки для производителя, и как следствие конечного потребителя с помощью группы рекуррентных нейронных сетей. Эта группа должна быть способна генерировать «партитуры» для музыкальных произведений заданного жанра.

Архитектура сетей

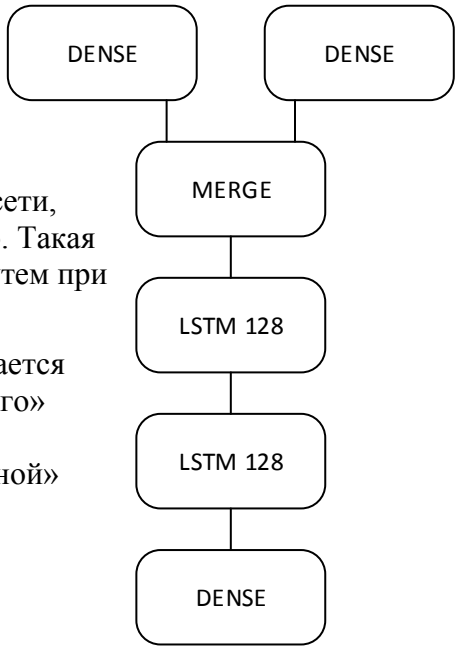
Выбор архитектуры нейронной сети был основан на сетях, генерирующих тексты – рекуррентных нейронных сетях типа LSTM (Long-Short-term-memory). Для построения такой сети были использованы TensorFlow – библиотека от Google и Keras – фреймворк для упрощения работы с TensorFlow. Во входном слое стоят нейроны, принимающие по одному символу, количество таких входов должно быть равно длине X (X - набор данных подающийся на вход нейронной сети при обучении), в выходном слое (y - слое) стоит столько нейронов, сколько различных «идентификаторов» есть в базе.



Архитектура «основной» нейронной сети, которая генерирует первую партию в произведении выглядит вот так. Такая архитектура выбрана экспериментальным путем при генерации текста.

Архитектура «дополнительной» нейронной сети, которая генерирует дополнительную партию. Такая архитектура выбрана экспериментальным путем при генерации текста.

При обучении дополнительных слоев выбирается «основная» партия и партия «дополнительного» инструмента, дальнейший процесс обучения проходит аналогично обучению сети «основной» партии.



Процесс обучения

Генерации музыкальных произведений с помощью рекуррентных нейронных сетей - задача сложная. Музыкальное произведение в лучшем случае двумерное (высота ноты и ее длительность). Для обучения сети на таких данных требуется специальная их подготовка. В качестве входных данных для обучения выбран формат midi, поскольку он является стандартом для точной записи музыкальных произведений и взаимно-однозначно отображается в ноты, но он недостаточно удобен для машинного обучения. Данные о музыкальном произведении в этом формате хранятся как события (в том числе начала и конца звучания ноты) для каждого инструмента. Приведение midi-файла к тексту решает вопрос удобности обучения, а генерация текста на нейронных сетях - задача известная.

Опишем алгоритм приведения:

1) Разобьём музыкальное произведение на части. Это легко сделать, нам известны параметры файла: *"ticks per quarter note"* – количество тиков на четверть и смещение каждой следующей ноты от предыдущей, а также размеры произведения (сколько нот приходится на один такт, например - 4/4). Пусть такая часть будет равна по длительности 1/16.
2) Для каждой 1/16 нам известно, какие ноты воспроизводятся за время звучания этой 1/16 конкретным инструментом.
3) Сопоставим каждой такой части некоторый идентификатор так, чтобы одинаковые наборы, звучащие одинаково имели одинаковый идентификатор.

В результате таких операций мы получили отображение в текст любого музыкального произведения с потенциальным расширением базы «частей» в формате midi. Теперь расскажем об алгоритме обучения и архитектуре сети:

Процесс обучения проходит итеративно, количество итераций определяется экспериментально. На каждой итерации на обучение нейросети дается два набора чисел (для работы с TensorFlow текст приводится к набору чисел-идентификаторов, будем называть их символы). Для их создания собирается два набора слов, первый – данные, поступающие на вход, второе, результаты, которые должна выдавать сеть на соответствующие входные данные. Входной массив формируется из оригинального текста последовательным смещением на «шаг» начала выборки сегмента заданной длины из текста и добавления этого сегмента в массив. В выходной массив добавляется следующее слово после текущего

выбранного сегмента. Первый массив приводится к трехмерному представлению: длина последовательностей символов, длина последовательности символов, количество всех уникальных символов. Массив содержит в себе только 0 и 1. Второй массив приводится к двумерному представлению: длина последовательностей символов, количество всех уникальных символов. Он так же состоит только из 0 и 1.

На такой информации нейросеть обучается на каждой итерации. После обучения в рамках одной итерации производится генерация текста на основе выбранного случайно сегмента текста, равного по длине входному набору.

Статистические данные по генерации

При использовании данного алгоритма с параметрами:

- Шаг обучения: 2,4,6,8 слов.
- Размер окна обучения: 10 слов.
- Итераций по тексту: 100.
- Разброс результирующей выборки: 0.5, 1.0 (0.2 не берем, поскольку в результате на всех итерациях появляются только точки и очень редко однобуквенные слова).

Возьмем первые десять слов для оригинального текста и посмотрим данные для сгенерированных текстов. Расставим эти слова в порядке убывания количества для оригинального текста: Первый столбец – размер шага, второй число слов, третий частота встречаемости слова, четвертый – абсолютная разница в частоте встречаемости между генерируемым текстом и оригиналом.

2	9900	каждые N слов	Разница	4	8000	каждые N слов	Разница
.	620	15,96774194	2,735371993	.	597	13,40033501	0,167965066
и	268	36,94029851	6,192145385	и	270	29,62962963	1,118523493
я	235	42,12765957	1,682429892	я	215	37,20930233	3,235927356
не	195	50,76923077	0,272017837	не	140	57,14285714	6,101608536
в	180	55	1,136470588	в	127	62,99212598	9,128596572
что	138	71,73913043	5,095753433	что	137	58,39416058	8,249216418
на	145	68,27586207	0,67594516	на	116	68,96551724	0,013710012
?	122	81,14754098	4,590286732	?	77	103,8961039	18,15827618
он	102	97,05882353	0,147121269	он	78	102,5641026	5,358157766
…	95	104,2105263	1,785470387	…	86	93,02325581	9,401800114
		Среднее	2,431301268			Среднее	6,093378152

6	9900	каждые N слов	Разница	8	9900	каждые N слов	Разница
.	829	11,94209891	1,290271028	.	788	12,56345178	0,668918166
и	298	33,22147651	2,473323387	и	352	28,125	2,623153123
я	214	46,26168224	5,816452561	я	257	38,52140078	1,923828904
не	192	51,5625	0,521251394	не	177	55,93220339	4,890954783
в	198	50	3,863529412	в	142	69,71830986	15,85478045
что	142	69,71830986	3,074932858	что	145	68,27586207	1,632485068
на	167	59,28143713	9,670370103	на	144	68,75	0,201807229
?	98	101,0204082	15,28258045	?	99	100	14,26217228
он	106	93,39622642	3,809718383	он	102	97,05882353	0,147121269
…	90	110	7,574944072	…	112	88,39285714	14,03219879
		Среднее	5,337737364			Среднее	5,623742006

Заметим, что для разных размеров шага, порядок первых десяти слов меняются, а некоторые из них даже не входят в десятку. Для большей наглядности построим графики частот встречаемости слов и разницы частот встречаемости между оригиналом и генерируемым текстом (теперь уже не абсолютной).

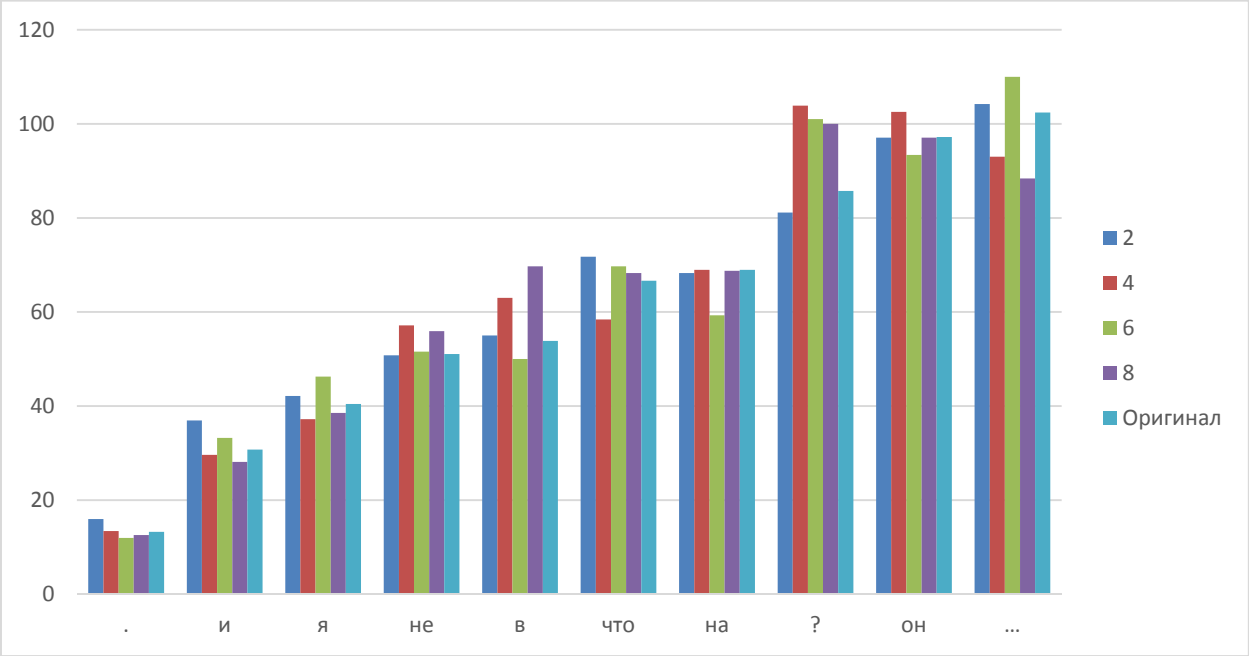


Рисунок 1 - Частотные характеристики появления слов

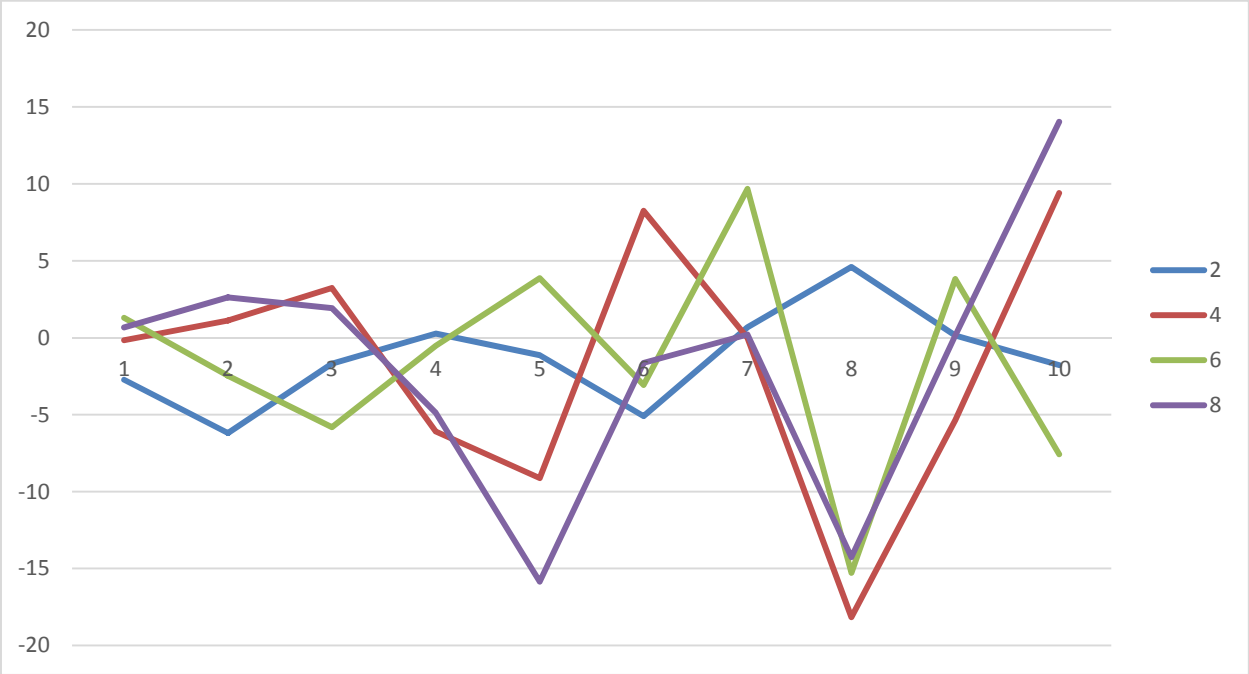


Рисунок 2 – Разница частот обучения от оригинала

Рассмотрим статистику:

Пусть T – исходный текст, T_1 - сгенерированный текст. Количество слов в них соответственно $M=|T|$ и $M_1=|T_1|$, а количество различных слов соответственно N и N_1.

Пусть n_i - количество вхождений слова I в T, n_{1i} - количество вхождений слова I в T_1. Найдем максимальное, минимальное и среднее количество вхождений слов:

$$n_{max} = \max\{n_i\},\ n_{min} = \min\{n_i\},\ n_{cp} = M/N\ n_{1max} = \max\{n_{1i}\},\ n_{1min} = \min\{n_{1i}\},\ n_{cp} = M_1/N_1$$

Определим частотность слова I в T и T_1

$$P_i = n_i/M,\quad p_{1i} = n_{1i}/M_1$$

Тогда можно найти максимальную, минимальную и среднюю частотность слов в тексте в целом

$$P_{max}=\max\{p_i\}\quad p_{1max} = \max\{p_{1i}\}$$

$$P_{max}=\max\{p_i\}\quad p_{1max} = \max\{p_{1i}\}$$

$$P_{cp} = \sum p_i/M\quad P_{1cp} = \sum p_{1i}/M_1$$

Выберем диапазон частотностей Δ и вычислим соответствующие частотности на отрезках

$$[p_{max}-\Delta, p_{max}],\ [p_{min}, p_{min}+\Delta],\ [p_{cp} -\Delta/2, p_{cp} +\Delta/2]$$

для текста T. Аналогичные вычисления произведем для текста T_1.

$$\tilde{p} = \frac{\sum_{p_i \in C} p_i}{K}$$

Тогда проведем вычисления для Δ = 0.7

Тогда частота для оригинального текста = 0.327150290786654

Частоты для текстов 2, 4, 6, 8:

Gen 2

MAX: 0.33740740740740743

MIN: 0.0003194888178913738

AVG: 0.00019972039145196724

Gen 4

MAX: 0.5224583333333334

MIN: 0.00042826552462526765

AVG: 0.5224583333333334

Gen 6

MAX: 0.553063973063973

MIN: 0.000434593654932638

AVG: 0.553063973063973

Gen 8

MAX: 0.522962962962963

MIN: 0.0

AVG: 0.522962962962963

Тогда проведем вычисления для Δ = 0.2

Тогда частота для оригинального текста = 0.3271502907866544

Частоты для текстов 2, 4, 6, 8:

Gen 2

MAX: 0.00019972039145196724

MIN: 0.00021175857864895898

AVG: 0.33740740740740743

Gen 4

MAX: 0.00019972039145196724

MIN: 0.00020460225649814338

AVG: 0.5224583333333334

Gen 6

MAX: 0.00019972039145196724

MIN: 0.00019432001171131605

AVG: 0.553063973063973

Gen 8

MAX: 0.00019972039145196724

MIN: 0.00017461200766709947

AVG: 0.522962962962963

Выводы по статистике: глубоких связей между изначальным текстом, и текстом, полученным при генерации не обнаружено, но близкая по частотности употребляемость популярных слов нас вполне удовлетворяет. Нас интересует обучение с минимальным шагом смещения окна, потому что частота встречаемости символов при таком обучении максимально близка к частоте оригинала.

Выводы

Метод генерации музыкальных произведений с использованием текста в качестве промежуточной ступени для генерации позволяет решить проблему многомерности входных данных, хотя и ценой достаточно сильной потери связности. В качестве метода генерации основы музыкального произведения этот метод подходит очень хорошо.

Список литературы

1. Библиотеки для глубокого обучения. [Электронный ресурс]. Режим доступа: http://www.pvsm.ru/cat/theano , свободный..
2. Методология обучения рекуррентной искусственной нейронной сети с динамической стековой памятью [Электронный ресурс]. Режим доступа: http://i-intellect.ru/articles/438/ , свободный.
3. Алгоритм и программная реализация гибридного метода обучения искусственных нейронных сетей [Электронный ресурс]. Режим доступа: http://i-intellect.ru/articles/283/ , свободный

КОМПЛЕКС КЛИЕНТ-СЕВРЕНЫХ ПРИЛОЖЕНИЙ PRICE-CHECKER

Мальков С.А. – студент, Крючкова Е.Н. – к.ф.-м.н., профессор
Алтайский государственный технический университет (г. Барнаул)

В современном обществе широкое распространение получают многофункциональные мобильные устройства, иначе – смартфоны. В связи со значительным увеличением их производительности, функциональности и доступности, стало возможно использовать их в тех сферах деятельности, в которых ранее они были не применимы. Одной из таких сфер является ритейлерская деятельность. Речь пойдет о применении мобильных технологий в сфере розничной торговли. Задачей ритейлеров является предоставление того, что ищет современный покупатель, путем грамотной комбинации мобильных технологий с отличными розничными предложениями. Многие уж делают шаги в этом направлении, но с точки зрения крупной торговли - мы все еще используем лишь крупицы возможного.

Одной из самых значимых разработок последних лет в сфере интеграции мобильных технологий в современный ритейл являются QR-коды - двумерные штрих-коды, первоначально изобретенные для автомобильной промышленности. С помощью мобильного телефона, клиенты могут сканировать QR-коды для просмотра информации о продукте, получения скидок, взаимодействия с брендом или покупки товара.

Еще одной разработкой является вариант использования дополненной реальности: в режиме реального времени дополняет мир цифровыми элементами, такими как звук или графика. Так компания Starbucks соединила технологии QR-кодов с дополненной реальностью в ее проекте «Волшебная чашка. Потребители любят «Вау-фактор», и дополненная реальность дает продавцам возможность представить покупателям инновационный опыт, который отвечает индивидуальным потребностям и повышает уровень продаж.

Так же стоит упомянуть и о мобильных платежах. Они навсегда изменят привычные способы оплаты покупок. Мобильные платежи предлагают покупателю максимальное удобство - возможность расплачиваться за товары и услуги без наличных, при помощи того, что является наиболее постоянным спутником потребителя. С точки зрения технологии, многие рассматривают Near Field Communication (NFC).

Объединяя вышесказанное и применяя все это к предприятиям розничной торговли, можно представить мобильное приложение, которое бы облегчило жизнь покупателям торговых сетей. Например, поход в магазин за продуктами может значительно ускориться, если при выборе покупок пользоваться смартфоном с установленным приложением, которое бы помогало бы при выборе товаров, а так формировало и рассчитывало стоимость корзины и помогало оплачивать покупки с помощью виртуального чека, многократно уменьшая время обслуживания покупателя на кассе.

Также, владельцы магазинов, предоставляющие пользователям возможность использования мобильного приложения, получают доступ к большому количеству первичной статистической информации о клиентах, чем они интересуются, почему не приобретают тот или иной товар и что выбирают взамен отложенного. Полученные данные могут быть использованы, например, для задач планирования в магазине. Так же появляются обширные возможности с точки зрения предоставления пользователям уникальных предложений, показа акций, рекламы и т.д.

Self-checkout как новая форма обслуживания покупателей

Self-checkout (касса самообслуживания) — новая форма обслуживания покупателей, которая в их глазах имеет много преимуществ. Современное общество все более склонно к общению с гаджетами, а не с людьми. Касса самообслуживания вызывает ощущение

защищенности у покупателя, потому что он сам контролирует процесс сканирования, процесс оплаты, и уверен в том, что его не обсчитают или не обманут, давая сдачу. Для некоторых покупателей может быть актуальна конфиденциальность покупки. Самое главное — self-checkout будет привлекать покупателей.

Описание предлагаемого решения

Стоит заметить, что и в обычных кассах, и в кассах самообслуживания клиент (или кассир, в случае со стандартными кассами) должен отсканировать все товары на кассе, узнав итоговую стоимость только в момент оплаты и практически не имея возможности заменить неподходящий товар.

На текущий момент ведется разработка мобильного приложения, которое бы позволяло оптимизировать скорость прохода клиентов через кассу, улучшить их осведомленность о приобретаемых товарах, а также расширить общий список возможностей взаимодействия клиента и магазина.

Оптимизация скорости прохода покупателей будет достигаться за счет того, что покупатель вместо того, чтобы сканировать каждый товар из собранной корзинки на кассе, будет предоставлять заранее подготовленную информацию о собранной корзине товаров, как единое целое, в удобном для сканирования на кассе виде, с помощью ее стандартных средств, например, ридером штрих кодов.

Важно уточнить, что данный сценарий еще более актуален на обычных кассах, где товар сканируется кассиром, т.к. как правило на обычных кассах очереди больше, относительно автоматизированных касс.

Так же мобильное приложение решает следующий ряд задач:

- По возможности идентифицировать клиента
- Осуществлять мониторинг действий пользователя в стенах торговой точки
- Отслеживать товары, которыми интересуется клиент
- Предоставлять клиентам актуальную информацию о товарах
- Показывать связанные товары
- Предоставлять информацию об акциях, проводимых в магазине
- Иметь возможность подписаться на товары, с последующими уведомлениями о скидках
- Показывать информацию о бонусном балансе

Важно иметь ввиду, что предлагаемые функции являются значительным конкурентным преимуществом и позволяют увеличить скорость обслуживания, а также повысить качество предоставляемых услуг.

Разрабатываемая система предназначена для внедрения в уже имеющуюся систему программного обеспечения торговых предприятий компании «ООО Ритейл Сервис».

Перспективы

Разработанная система имеет большие возможности к дальнейшему расширению. В дальнейшем может быть реализовано:

- Отображение рекламных предложений;
- Пользовательская статистика;
- Индивидуальные предложения;
- Расширение предоставляемой информации о товарах;
- Различного рода реклама.

Список литературы

1. Статья «Мобильные приложения – модный тренд или необходимость?» [Электронный ресурс]. // RETAIL. – Режим доступа: http://www.retail.ru/interviews/79960, свободный.
2. Статья «Мобильные технологии на вооружении торговых продовольственных сетей» [Электронный ресурс]. // ADVERTOLOGY. – Режим доступа: http://www.advertology.ru/article124100.htm, свободный.
3. Голощапов, А. Google Android. Создание приложений для смартфонов и планшетных ПК [Текст] / А. Голощапов. — БХВ-Петербург. — 928 с.
4. Статья «Сканеры штрих кода» [Электронный ресурс]. //SCANCODE. – Режим доступа: http://www.scancode.ru/catalog/2, свободный.

РАЗРАБОТКА САМООБУЧАЮЩИХСЯ АЛГОРИТМОВ ДЛЯ ТРЕКИНГА ОБЪЕКТОВ В ВИДЕОПОТОКЕ

Некрасов Д.В. - студент, Старолетов С.М. - к.ф.-м.н., доцент
Алтайский государственный технический университет (г. Барнаул)

Трекингом называется определение местоположения движущегося объекта (или нескольких объектов) во времени с помощью камеры. Алгоритм анализирует кадры видео и выдает положение движущихся целевых объектов относительно кадра. Основная проблема в трекинге состоит в сопоставлении положений целевого объекта на последовательности кадров, особенно если объект движется быстро относительно частоты кадров [1, 2].

Более формально, задачу трекинга объектов можно сформулировать следующим образом:

- Дана последовательность кадров { I(t, x, y) }.
- Задан объект слежения (может быть несколько).
- Требуется:
  - обнаружить объект на каждом кадре;
  - определить траекторию объекта.

В настоящее время существует множество алгоритмов трекинга объектов. Глобально их можно разделить на несколько больших категорий: с использованием алгоритмов машинного обучения и без него. Основные алгоритмы трекинга объектов приведены на рисунке 1.

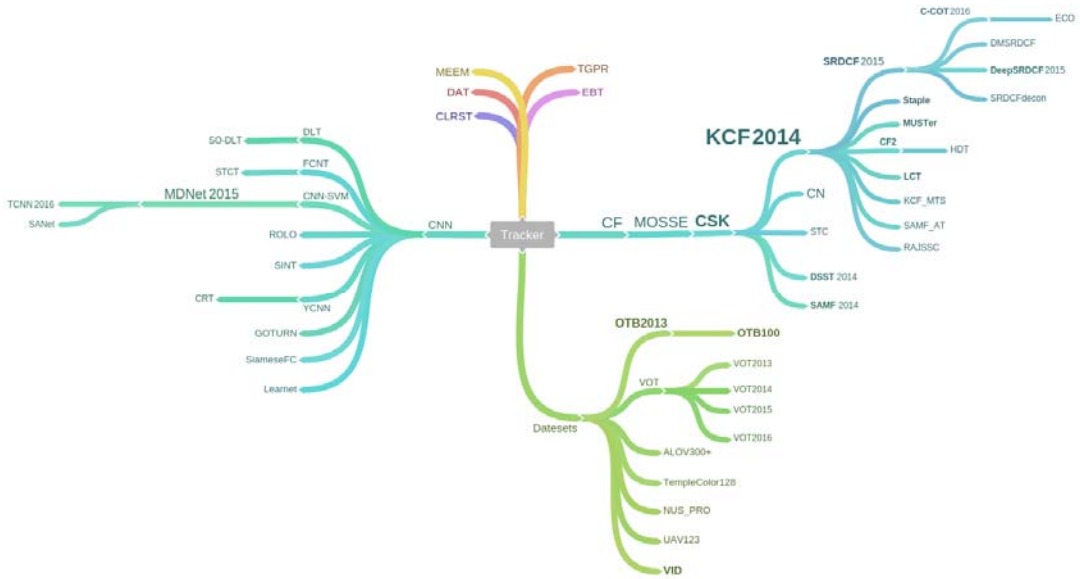


Рисунок 1 - Основные алгоритмы трекинга объектов

Алгоритмы трекинга объектов без использования машинного обучения состоят из двух основных частей:

- Представление и локализация целевого объекта.
- Фильтрация и объединение данных.

Представление и локализация целевого объекта представляет собой по большей части восходящий процесс, т.е. последовательный и его последующие шаги не затрагивают предыдущие. Обычно вычислительная сложность этих алгоритмов достаточно мала. Ниже приведены стандартные алгоритмы представления и локализации целевого объекта:

- Blob tracking: Сегментация интерьера объекта (например blob detection, block-based correlation или оптический поток (optical flow)).
- Kernel-based tracking (Mean-shift tracking): Итеративная процедура локализации, основанная на максимизации критерия подобия (Bhattacharyya coefficient).
- Contour tracking (трекинг контуров): Поиск границы объекта (например активные контуры или Condensation algorithm).
- Визуальное согласование особенностей (feature matching): Регистрация (Image registration).
- Point feature tracking (Слежение за точечными особенностями сцены): Задача формулируется так - дана последовательность изображений некоторой сцены, полученная с движущейся или неподвижной камеры. Необходимо получить набор как можно более точных последовательностей координат проекции некоторых точек сцены в каждом кадре.
- Фильтрация и объединение данных представляет собой по большей части нисходящий процесс, который включает в себя объединение априорной информации о сцене или объекте, соотносящейся с динамикой объекта и вычислением различных гипотез. Вычислительная сложность этих алгоритмов обычно намного выше. Вот некоторые стандартные алгоритмы фильтрации:
- Фильтр Калмана: оптимальный рекурсивный (Bayesian filter) для линейных функций, подверженных шуму по Гауссу.
- Фильтр частиц (Particle filter): полезно для семплинга базового пространства состояний распределения нелинейных и негауссовых процессов.

Алгоритмы с использованием машинного обучения можно разделить на два вида: с использованием классических алгоритмов машинного обучения (метод опорных векторов, линейная регрессия, логистическая регрессия, случайный лес, градиентный бустинг и т. д.) и с использованием алгоритмов глубокого обучения (свёрточные нейронные сети с количеством скрытых слоёв не меньше трёх).

К алгоритмам первого вида относится, например, алгоритм P-N Learning, предложенный чешским учёным Зденеком Калалом [3]. Этот алгоритм для обучения использует информацию, получаемую во время съёмки (т. е. самообучается). На этапе детектирования в качестве классификатора используется алгоритм случайного леса. Предложенная учёным схема приведена на рисунке 2.

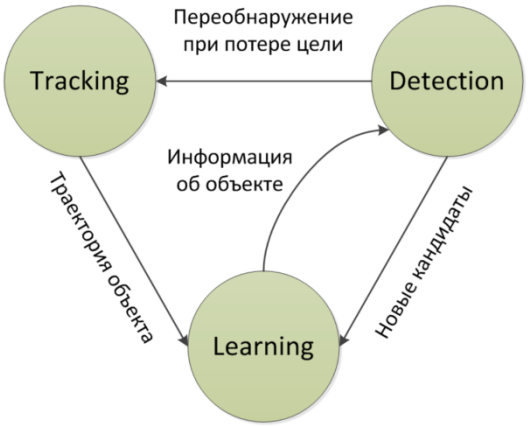


Рисунок 2 - Общая схема алгоритма P-N Learning.

Данный алгоритм обладает рядом достоинств:

- Работа в реальном времени.
- Неограниченная длительность слежения.
- Отсутствие стадии off-line обучения (не требуется априорная информация об объекте).
- Стабильность к перекрытиям.
- Алгоритм также содержит некоторые недостатки:
  - Слежение только за одним объектом,
  - Требуется ручная инициализация цели.

Примером алгоритма трекинга объектов с использованием глубокого машинного обучения является алгоритм Online Tracking by Learning Discriminative Saliency Map with Convolutional Neural Network [4]. Этот алгоритм основан на предварительно обученной свёрточной нейронной сети с дополнительным слоем, который отвечает за классификацию объектов методом опорных векторов. Схема работы этого алгоритма приведена на рисунке 3.

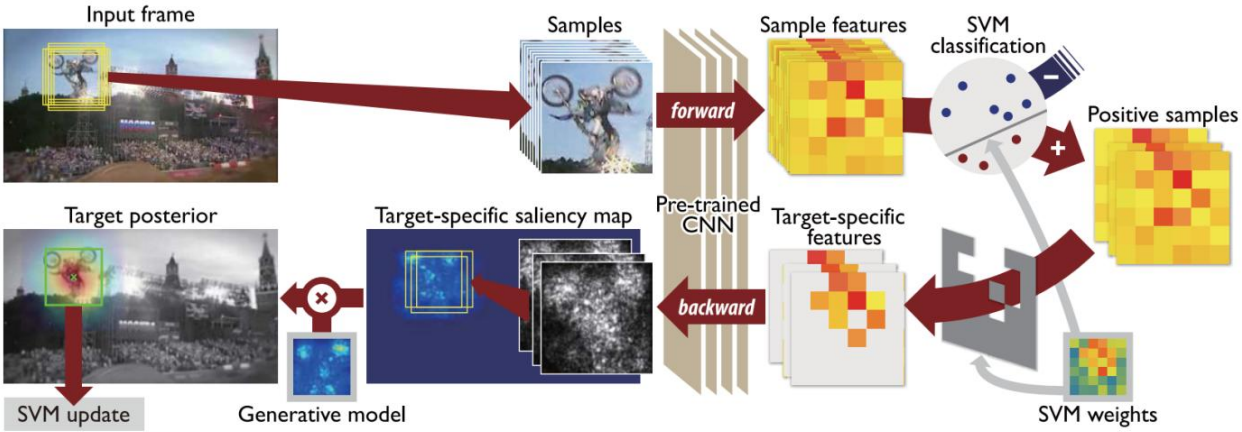


Рисунок 3 – Схема алгоритма Online Tracking by Learning Discriminative Saliency Map with Convolutional Neural Network

Данный алгоритм обладает более высокой точностью, по сравнению с алгоритмом P-N Learning, однако он имеет ряд недостатков. Для работы алгоритма необходимо предварительно обучить нейросеть для каждого объекта, который мы хотим отслеживать, в то время как в алгоритме P-N Learning нам нужно только указать сам объект на видео. Кроме того, данный алгоритм способен отслеживать только один объект, как и в случае алгоритма P-N Learning.

После проведённого анализа алгоритмов трекинга объектов было решено разработать алгоритм для трекинга нескольких объектов на видео. За основу было решено взять алгоритм P-N Learning, поскольку он не требует предварительного обучения, алгоритм учится во время работы и с течением времени увеличивает точность отслеживания. Для реализации множественного трекинга предлагается использовать идеи многопоточного программирования для распараллеливания работы алгоритма на несколько объектов [5].

В качестве демонстрации работы алгоритма будет разработано десктопное приложение с возможностью выделения необходимых для трекинга объектов, а также с настройкой некоторых параметров алгоритма.

Список литературы

1. Кустикова В.Д., Разработка мультимедийных приложений с использованием библиотек OpenCV и IPP. Отслеживание движения и алгоритмы сопровождения ключевых точек. Н. Новгород, Нижегородский государственный университет им. Н.И. Лобачевского, 2013. – 94 с.
2. Форсайт Д.А., Понс Ж. Компьютерное зрение. Современный подход. М., Вильямс, 2004. – 928 с.
3. Zdenek Kalal, Jiri Matas, Krystian Mikolajczyk. P-N Learning: Bootstrapping Binary Classifiers by Structural Constraints, CVPR 2010
4. Seunghoon Hong, Tackgeun You, Suha Kwak, Bohyung Han, Online Tracking by Learning Discriminative Saliency Map with Convolutional Neural Network, arXiv:1502.06796
5. Maurice Herlihy, Nir Shavit, The Art of Multiprocessor Programming, Morgan Kaufmann, 2012. - 552 с.

ПРОЕКТРОВАНИЕ ВЕРИФИЦИРУЮЩЕГО РАСШИРЕНИЯ КОМПИЛЯТОРА ДЛЯ ФОРМАЛЬНОЙ ПРОВЕРКИ РАСПРЕДЕЛЕННЫХ ПРИЛОЖЕНИЙ

Дубко А.В. – магистрант, Старолетов С.М. – к.ф.-м.н., доцент
Алтайский государственный технический университет (г. Барнаул)

Актуальность

С каждым новым уровнем сложности программных систем возрастает вероятность совершения ошибок, при этом затрудняется их быстрое обнаружение. Особенно подвержены ошибкам распределенные и многопоточные программы. Такие ошибки обычно невозможно или крайне тяжело обнаружить тестированием.

Проблема верификации и создания верифицирующего компилятора является одной из фундаментальных проблем современного программирования, находящаяся в стадии решения.

Предметная область

Компилятор – это системная компьютерная программа, которая транслирует написанные человеком программы с языка высокого уровня в эквивалентные исполняемые программы.

Верифицирующий компилятор – компилятор, который доказывает специфицированные математические утверждения о свойствах транслируемых программ.

Главная задача компилятора состоит в генерации кода для целевой машины и в проверке правильности программы (отсутствии синтаксических и семантических ошибок).

Ошибка – отклонение, из-за которого программа выдает неожиданное поведение и, как следствие, результат.

Верификация [1] (рисунок 1) – подтверждение на основе представления объективных свидетельств того, что установленные требования к программному продукту были выполнены.

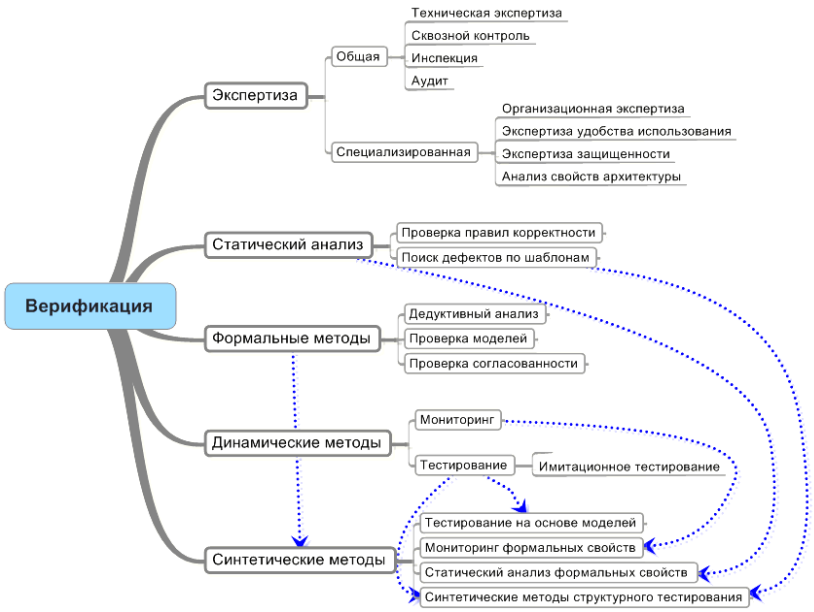


Рисунок 1 – Классификация методов верификации

Динамическая верификация – приемы и методы, которые используют результаты реальной работы программной системы для проверки этих результатов установленным требованиям.

Формальная верификация – приемы и методы формального доказательства (или опровержения) того, что модель программной системы удовлетворяет заданной формальной спецификации [2].

В данной работе основной упор сделан на метод формальной верификации – проверка модели (model checking). Это метод проверки того, что на данной формальной модели системы заданная логическая формула выполняется. Исторически эта техника была разработана для моделей систем переходов и темпоральных логик.

Для выполнения верификации спецификация программы должна состоять из набора формальных утверждений о желаемых свойствах поведения системы. Для формальной спецификации используется язык логики, каждое утверждение которого может быть либо истинно, либо ложно для проверяемой программной системы. Верификатор проверяет выполнение заданных формальных утверждений на заданной формальной модели.

Наиболее широко распространенным методом проверки правильности систем является сегодня тестирование. Тестирование – динамический метод верификации, заключающийся в проверке работы построенной системы в различных реальных ситуациях, при различных исходных данных.

У тестирования есть недостатки:

- тестирование очень трудоемкий процесс;
- все реакции системы при выполнении тестов должны быть заранее зафиксированы;
- тестированием можно проверить лишь очень немногие траектории вычисления системы;
- тестирование не может гарантировать правильность системы;
- тестированием невозможно выявить ошибки в параллельных системах, которые могут зависеть от соотношения скоростей процессов, редко встречающихся ситуаций и поэтому невоспроизводимы.

В пользу создания верифицирующих компиляторов или расширений, помимо недостатков тестирования, говорят также:

- реальная стоимость ошибок;

- повышение качества программ, еще на этапе разработки;
- исследования при создании, затрагивающие как теорию программирования, так и программную инженерию.

Поэтому лучшим решением будет являться разработка верифицирующего расширения на основе средств платформы Roslyn для компилятора и среды разработки на языке C#.

Распределенные системы

Распределенной системой называют такую систему, в которой отношения положений элементов непосредственно влияют на функционирование системы, в том числе на анализ и синтез системы.

Распределённые вычисления – это метод решения трудоёмких вычислительных задач, используя несколько компьютеров, обычно объединённых в параллельную вычислительную систему. Распределённые вычисления применяются также и в распределенных системах управления.

Проблемы распределенных вычислений

- гонка данных (race condition);
- бесконечная отсрочка (indefinite postponement);
- взаимоблокировка (deadlock).

Обзор существующих технологий

Платформа .NET и язык C# предоставляют возможность для создания распределенных компонент или приложений:

- Web API;
- WCF (Windows Communication Foundation);
- .NET Sockets.

Платформа .NET предоставляет средства для статического анализа кода в качестве расширяемых инспекций платформы компилятора Roslyn языка C#. Также платформа предоставляет средства для динамической верификации: Microsoft Test.

Однако средств для формальной верификации приложений (в частности распределенных) как таковых платформа .NET не предоставляет.

Верификатор Spec#

Spec# является языком программирования с поддержкой особенностей языка спецификаций, расширяющих возможности языка программирования C# и содержит инструмент статической проверки, основанный на доказательстве теорем. Можно говорить о верифицирующем компиляторе на уровне спецификаций.

Spec# расширяет C# следующими возможностями [3]:

- использование ненулевых типов;
- структуры предусловий и постусловий;
- проверяемые исключения;
- инварианты.

К сожалению, на уровне спецификаций методов крайне затруднен подход верификации распределенных приложений.

Roslyn

Roslyn является платформой, которая предоставляет разработчику различные мощные средства для анализа и разбора кода [4]. Исходный код данной платформы имеется в открытом доступе.

С помощью средств анализа, предоставляемыми платформой Roslyn, можно производить полный разбор кода, анализируя все поддерживаемые конструкции языка.

Можно выделить несколько пунктов (рисунок 2), которые нужно выполнить для получения необходимых для анализа данных:

- создание рабочего пространства (workspace);
- получение решений (solution – не обязательно);
- получение проектов;
- разбор проекта: получение компиляции, списка файлов;
- разбор файла: получение синтаксического дерева и семантической модели.

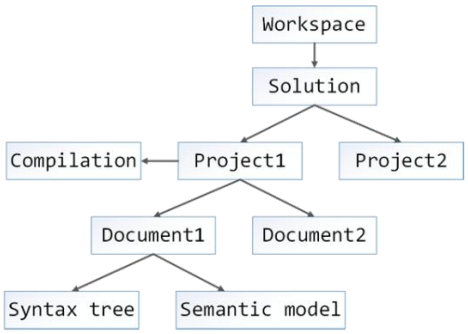


Рисунок 2 –Получение данных при анализе в Roslyn

Синтаксическое дерево строится на основании исходного кода программы и используется для анализа различных конструкций языка. Семантическая модель предоставляет информацию об объектах и их типах.

Для каждой конструкции языка определены узлы своего типа, а для каждого типа узла определён метод, выполняющий обход узлов подобного типа.

Синтаксическое дерево является базовым элементом, необходимым для анализа кода. Именно по нему происходит перемещение в ходе анализа. Дерево строится на основе кода, приведённого в файле, из чего следует вывод, что каждый файл имеет своё синтаксическое дерево.

Существует три основных элемента дерева:

- syntax nodes – синтаксические конструкции языка;
- syntax tokens – лексемы, терминальные символы грамматики языка (идентификаторы, ключевые слова, спец. символы);
- syntax trivia – элементы дерева, которые не будут скомпилированы в IL-код, дополнительная синтаксическая информация (комментарии, директивы препроцессора, пробелы).

Акторная модель

Акторная модель представляет собой математическую модель параллельных вычислений, которая трактует понятие «актор» как универсальный примитив параллельного вычисления.

Актор является вычислительной сущностью, которая в ответ на полученное сообщение может одновременно:

- Отправить конечное число сообщений другим акторам;
- Создать конечное число новых акторов;

- Выбрать поведение, которое будет использоваться при обработке следующего полученного сообщения.

Не предполагается существования определенной последовательности вышеописанных действий и все они могут выполняться параллельно.

Получатели сообщений идентифицируются по адресу (Pid), который иногда называют «почтовым адресом».

Модель акторов характеризуется внутренне присущим параллелизмом вычислений внутри одного актора и между акторами, динамическим созданием акторов, включением адресов акторов в сообщения, а также взаимодействием только через прямой асинхронный обмен сообщениями без каких-либо ограничений на порядок прибытия сообщений.

Верификатор SPIN

SPIN – утилита для верификации корректности распределенных программных моделей. Аббревиатура расшифровывается как Simple Promela Interpreter – простой интерпретатор программ на языке Promela. Данная утилита служит для автоматизированной проверки моделей, а также может работать в качестве симулятора, исполняя один из возможных путей работы системы, предоставляя результаты этого исполнения.

Формализация модели на языке Promela

С помощью системы SPIN выполняется проверка не самих программ, а их моделей. Для построения модели по исходной параллельной программе или алгоритму пользователь (обычно вручную) строит представление этой программы на входном языке пакета SPIN – Promela (Protocol Meta Language). Требования же к модели описываются на языке темпоральной логики линейного времени LTL.

Для доказательства свойств модели используется верификация. Верификатор пытается найти контрпример – неправильную, ошибочную траекторию поведения, опровергающую заданное пользователем свойство, анализируя все возможные поведения модели.

Программы на Promela не имеют классов, они представляют плоскую структуру взаимодействующих параллельных процессов (акторную модель), имеют минимум управляющих конструкций, все переменные имеют конечные области определения. Поэтому такую программу можно считать моделью анализируемой системы они представляют абстракцию системы, в которой пользователь должен отразить те аспекты и характеристики реальной проверяемой системы, которые значимы для заданных для верификации свойств.

Формализация требований к модели на языке LTL

Темпоральная логика – это любая логическая система, которая позволяет формализовать утверждения, истинность которых изменяется со временем, не вводя явно понятие времени.

Язык LTL состоит из множества атомарных высказываний, логических операций и темпоральных операторов. Пусть φ – правильно построенная формула. Тогда темпоральные операторы:

- Xφ (в следующем состоянии φ верно – neXt);
- Gφ (φ верно всегда – Globally);
- Fφ (φ когда-нибудь будет верно – Finally);
- ψUφ (ψ будет верно до тех пор, пока не станет верно φ – Until)

– тоже правильно построенные формулы [6].

SPIN не поддерживает оператор X (neXt).

Постановка задачи

1. По произвольному программному коду на языке C#, используя средства платформы компилятора Roslyn, перевести его в промежуточную акторную модель.
2. По промежуточной акторной модели построить модель на языке Promela для верификатора SPIN.
3. Проверить ее корректность исходя из предположений безопасности, живучести и достаточности, а также предположений о дизайне системы, задаваемых вручную.
4. Верификатор, проведя автоматическую верификацию модели сообщит о ее корректности или некорректности. Эту информацию преобразовать в уровень кода и показать в списке ошибок, привязанных к линиям кода.

В настоящий момент производится реализация данного проекта в виде магистерской диссертации.

Список литературы

1. Кулямин В.В., Методы верификации программного обеспечения, Институт системного программирования РАН, 2009.
2. Верификация автоматных программ с использованием верификатора Spin. М.А. Лукин, А.А. Шалыто. СПбГУ ИТМО. 2008. № 53. Автоматное программирование, с. 145-162.
3. Spec# – Microsoft Research. – Режим доступа: https://www.microsoft.com/en-us/research/project/spec/
4. Roslyn Overview. – Режим доступа: https://github.com/dotnet/roslyn/wiki/Roslyn%20Overview
5. Введение в язык Promela и систему комплексной верификации Spin. И.В. Шошмина, Ю.Г. Карпов. Санкт-Петербургский государственный политехнический университет, 2012.

СИСТЕМА РАСПОЗНАВАНИЯ МАРКИ АВТОМОБИЛЕЙ ПО ФОТОГРАФИЯМ С КАМЕР ФОТОФИКСАЦИИ КАК ЗАДАЧА КЛАССИФИКАЦИИ ИЗОБРАЖЕНИЙ

Лаптев М.А. – магистрант, Старолетов С.М. – к.ф.-м.н., доцент
Алтайский государственный технический университет (г. Барнаул)

Анализ транспортных средств широко используется в различных программных системах, таких как помощник водителя, умная парковка, системы автопилотирования автомобиля, системы контроля дорожного траффика и скоростного режима. Также имеются автоматические системы сбора платы за проезд на основе идентификации автомобиля.

Задача анализа транспортных средств включает в себя распознавание номера регистрационного знака, определение типа транспортного средства [7] и определение марки и модели автомобиля [8]. Распознавание номера регистрационного знака позволяет однозначно идентифицировать транспортное средство, но процесс распознавания не является надёжным и имеет ряд специфических проблем, таких как плохое разрешение изображений, недостаточное освещение и подложные номера. Таким образом, определение типа, марки и модели транспортного средства помогает подтвердить идентификацию по номеру, а также предоставляет ценную информацию в том случае, когда идентификация по номеру невозможна. По этим причинам ГИБДД нуждается в системе распознавания марки автомобилей по фотографиям с камер фотофиксации.

Однако, различать транспортные средства одного производителя и одной модели на текущий момент актуальная, но не решённая задача. Существует подход, основанный на выделении SURF особенностей для решения сразу двух задач: детектирование и распознавание автомобилей [9]. Для увеличения эффективности вычислений на изображениях с фронтальным и задним видами широко используется выделение региона интереса. Первыми эту идею выдвинули Петровиц и Кутц [10], заложив при этом основу для дальнейших исследований [11,12]. Другим обширным подходом является использование 3D особенностей геометрии автомобилей [13,14]. В этих работах предполагается сопоставлять 3D линии с 2D изображениями. Существуют также подходы, основанные на SIFT дескрипторах [15], углах Хариса [16] и дескрипторах LBP [17]. Что касается классификации, то чаще всего используются: метод опорных векторов, метод ближайших соседей и нейронные сети. В данной статье будет приведён обзор свёрточных нейронных сетей, как самого перспективного на сегодняшний день метода.

Свёрточная нейронная сеть (convolutional neural network, CNN) была представлена в 1998 году французским исследователем Яном Ле Куном (Yann LeCun) [6], как развитие модели неокогнитрон (neocognitron).

Типичная модель свёрточной сети, состоит из трёх типов слоёв: свёрточные (convolutional) слои, субдискретизирующие (subsampling, подвыборка) слои и слои "обычной" нейронной сети – перцептрона.

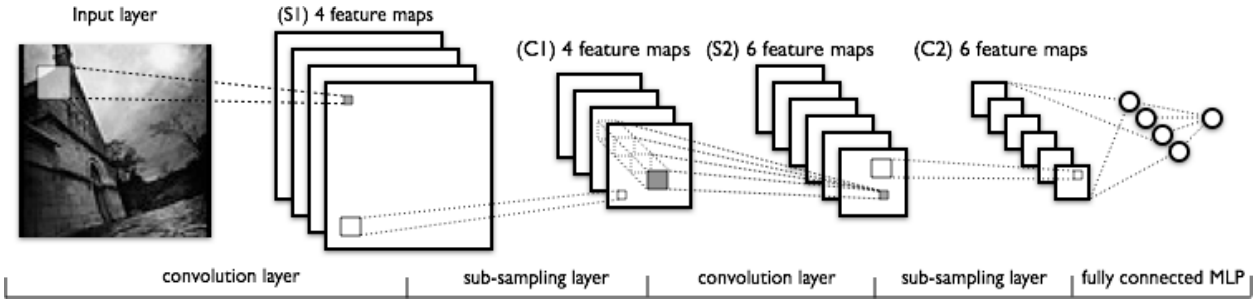


Рисунок 1 − Схема свёрточной сети

Первые два типа слоёв (convolutional,subsampling), чередуясь между собой, формируют входной вектор признаков для многослойного перцептрона. Сеть можно обучать с помощью градиентных методов.

Свёрточный слой реализует идею локальных рецептивных полей, т.е. каждый выходной нейрон соединен только с определённой (небольшой) областью входной матрицы и таким образом моделирует некоторые особенности человеческого зрения.

В упрощённом виде этот слой можно описать следующей формулой (1):

$$x^i = f\,(x^{i-1} * k^i + b^i) \qquad (1)$$

Здесь x^i - выход слоя i, f() - функция активации, b –коэффициент сдвига, символом «*» обозначена операция свёртки входа x с ядром k.

Субдискретизирующие слои выполняют уменьшение размера входной карты признаков (обычно в 2 раза). Это можно делать разными способами, например, методом выбора максимального элемента (max-pooling) - вся карта признаков разделяется на ячейки 2x2 элемента, из которых выбираются максимальные по значению. Формально слой может быть описан следующим образом (2):

$$x^i = f\,(a^i * \mathrm{subsample}(x^{i-1}) + b^i) \qquad (2)$$

Здесь x^i - выход слоя l, f() - функция активации, a,b - коэффициенты, subsample() - операция выборки локальных максимальных значений.

Использование этого слоя позволяет улучшить распознавание образцов с изменённым масштабом (уменьшенных или увеличенных).

Слой MLP - это слой “обычного” многослойного перцептрона (MLP), его можно описать следующим соотношением (3):

$$x_j^l = f\left(\sum_i x_i^{l-1} \cdot w_{ij}^{l-1} + b_j^{l-1}\right) \quad , \tag{3}$$

Здесь x^l - выход слоя l, f() - функция активации, b - коэффициент сдвига, w - матрица весовых коэффициентов.

В зависимости от задачи, свёрточные сети могут иметь разную топологию.

Рассмотрим свёрточную сеть из 7 слоёв, порядок их следования описан ниже.

1. входной слой - матрица картинки;
2. свёрточный слой - набор однотипных матриц (карт признаков);
3. субдискретизирующий слой - уменьшенный в 2 раза предыдущий набор матриц;
4. свёрточный слой - предыдущий набор матриц объединяется в одну, в соответствии со схемой соединения слоёв, и генерируется новый набор;
5. субдискретизирующий слой - уменьшенный в 2 раза предыдущий набор матриц;
6. слой MLP - предыдущий набор матриц разворачивается в вектор и обрабатывается как MLP;
7. слой MLP (выходной).

При этом, нейроны (карты признаков) второго субдискретизирующего слоя и третьего свёрточного слоя соединяются выборочно т.е. в соответствии с матрицей смежности, которая задаётся как параметр сети. Таким образом, каждая выходная карта формируется частичной суммой результатов свёрток входных карт, для каждой такой частичной суммы свой набор ядер свёртки.

Разработка топологий – сложный и недетерминированный процесс. На текущий момент есть две наиболее эффективные топологии: Inception от Google [3] и ResNet от Microsoft.

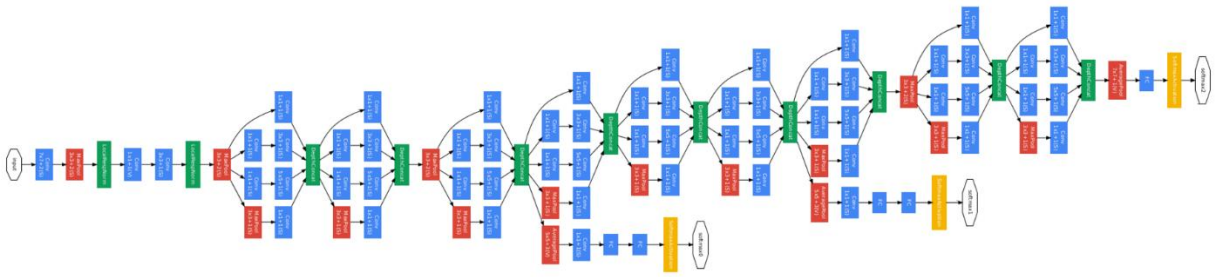


Рисунок 2 − Схема топологии Inception

Рассмотренные два вида сетей, а также их гибриды выигрывают соревнование ImageNet с 2014 года. Эти сети не стоят на месте и каждый год появляется новая инкарнация Inception и новая версия ResNet. Текущим трендом является замена крупных свёрток более мелкими [4] и добавление новых укороченных (shortcut) связей между уровнями [5], что позволяет пробрасывать найденное решение через несколько слоёв и избегать переобучения.

Для увеличения скорости обучения и повышения итоговой точности был разработан алгоритм предобработки изображения с камеры ГИБДД, включающий выделение региона интереса и приведение его к единому бинаризованному формату (Рисунок 3).

Рисунок 3 – Результат бинаризации региона интереса

Весь алгоритм представлен на рисунке 4 и представляет собой последовательное применение графических фильтров и методов компьютерного зрения для выделения наиболее ценной информации о фронтальной части автомобиля.

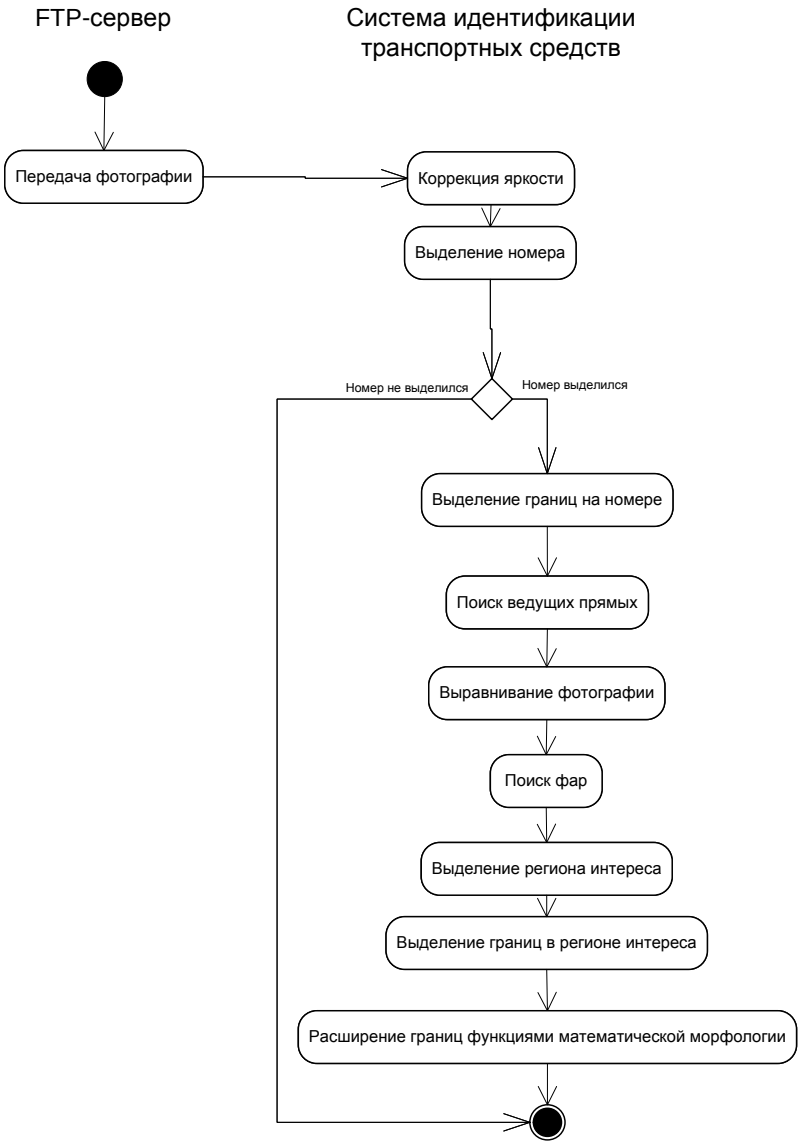


Рисунок 4 – Схема алгоритма выделения региона интереса и предварительного анализа

Заключение

На сегодняшний день наилучшим образом задачу классификации решают алгоритмы, основанные на нейронных сетях. Глубокие свёрточные сети справляются с классификацией изображений лучше, чем человек. Наиболее эффективными сетями являются Inception и ResNet, они показывают лучшие результаты в соревнованиях ImageNet.

Задача распознавания марки и модели автомобиля – частный случай классификации изображений, и поэтому может быть решена описанными методами. Наиболее перспективным представляется разрабатываемый нами гибридный подход, позволивший бы объединить несколько нейронных сетей с различной архитектурой и классические алгоритмы компьютерного зрения [2]. Такой подход позволит добиться более высокой точности распознавания, а также получить возможность оценивать вероятность верного распознавания для новых изображений, сравнивая результаты, полученные разными методами. Алгоритм предобработки уже реализован и следующим этапом планируется непосредственное обучение сетей.

Список литературы

1. Лаптев М.А., Старолетов С.М. Система распознавания марки автомобиля по фотографиям с камер фотофиксации. - Научно-образовательный журнал АлтГТУ "Горизонты образования". - 2016, вып. 18. Режим доступа: http://edu.secna.ru/media/f/pi2016.pdf
2. Лаптев М.А., Старолетов С.М. Математическая модель интегрального алгоритма идентификации транспортных средств / XVII Всероссийская конференция молодых учёных по математическому моделированию и информационным технологиям – 2016. Режим доступа: http://conf.nsc.ru/files/conferences/ym2016/pdf/341007/ru/tex/abstracts_341007_ru.pdf
3. Going Deeper with Convolutions. Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich - 2014
4. Rethinking the Inception Architecture for Computer Vision Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna -2015
5. Deep Residual Learning for Image Recognition Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun Microsoft Research - 2015
6. LeCun, Yann. "LeNet-5, convolutional neural networks". – http://yann.lecun.com/exdb/lenet/
7. Chen, Z.; Ellis, T.; Velastin, S.A. Vehicle type categorization: A comparison of classification schemes. In Proceedings of the 2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC), Washington, DC, USA, 5–7 October 2011; pp. 74–79.
8. Ma, X.; Eric, W.; Grimson, L. Edge-based rich representation for vehicle classification. In Proceedings of the 2005 Tenth IEEE International Conference on Computer Vision (ICCV), Beijing, China, 17–21 October 2005; pp. 1185–1192.
9. Hsieh, J.W.; Chen, L.C.; Chen, D.Y. Symmetrical SURF and Its Applications to Vehicle Detection and Vehicle Make and Model Recognition. IEEE Trans. Intell. Transp. Syst. 2014, 15, pp. 6–20.
10. Petrovic, V.; Cootes, T. Analysis of features for rigid structure vehicle type recognition. In Proceedings of the British Machine Vision Conference, London, UK, 7–9 September 2004; pp. 587–596.
11. Zafar, I.; Edirisinghe, E.A.; Avehicle, B.S. Localized contourlet features in vehicle make and model recognition. In Proceedings of the SPIE Image Processing: Machine Vision Applications II, San Jose, CA, USA, 18–22 January 2009; p. 725105.

12. Pearce, G.; Pears, N. Automatic make and model recognition from frontal images of vehicles. In Proceedings of the IEEE International Conference on Advanced Video and Signal-Based Surveillance, Klagenfurt, Austria, 30 August–2 September 2011; pp. 373–378.
13. Prokaj, J.; Medioni, G. 3-D model based vehicle recognition. In Proceedings of the 2009 Workshop on Applications of Computer Vision, Snowbird, UT, USA, 7–8 December 2009; pp. 1–7.
14. Ramnath, K.; Sinha, S.N.; Szeliski, R.; Hsiao, E. Car make and model recognition using 3D curve alignment. In Proceedings of the IEEE Winter Conference on Applications of Computer Vision, Steamboat Springs, CO, USA, 24–26 March 2014.
15. Lowe, D.G. Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. 2004, 60, pp. 91–110.
16. Mikolajczyk, K.; Schmid, C. Scale and Affine Invariant Interest Point Detectors. Int'lJ.Computer Vision 2004,1, pp. 63–86.
17. Varjas, V.; Tanacs, A. Vehicle recognition from frontal images in mobile environment. In Proceedings of the 8th International Symposium on Image and Signal Processing and Analysis, Trieste, Italy, 4–6 September 2013; pp. 819–823.
18. Ahonen, T.; Hadid, A.; Pietikainen, M. Face Description with Local Binary Patterns: Application to Face Recognition. IEEE Trans. Pattern Anal. Mach. Intell. 2006, 28, pp. 2037–2041.

РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ НА ПЛАТФОРМЕ UWP ДЛЯ СОСТАВЛЕНИЯ ГРАФИКА РАБОТ

Данильченко Д.Б. – студент
Алтайский государственный технический университет (г. Барнаул)

В настоящее время работодателям разрешено предоставлять рабочее место со сменным графиком работ. В суровых реалиях эти графики создаются в табличных и текстовых редакторах, а также, не редки случаи составления карандашом от руки. После утверждения графика, необходимо уведомить работников, для этого обычно применяется способ вывешивания листа с тем самым планом, где-нибудь на стенде в проходной. Но человек – не робот, и он не может помнить все, а тем более график на целый месяц вперед. Также, могут возникнуть ситуации, что человек попросил подменить себя на смене, а его товарищ забыл об этой договоренности и вспомнил только в следующий раз, когда его «поблагодарил» обратившийся к нему человек. В целях предотвращения такой плачевной картины с составлением графика и отсутствием уверенности в актуальности информации о дате выхода на следующую смену, было предложено разработать приложение для составления графика смен для персонального компьютера и мобильное приложение для обновления и просмотра составленного графика смен.

Разработка приложения велась для линейки Windows устройств – мобильные и настольные. Использовался новый SDK – UPW (Universal Windows Platform) [2], набор контролов и компонентов от компании Syncfusion – Accordion, Calendar, Chart и RadialMenu.

В мобильном приложении доступны следующие функции:

1. Обновить график (только в одном направлении от «начальника» к «рабочему»).
2. Редактировать график локально (автономно).
3. Устанавливать будильники на смену.
4. Устанавливать отметки о фактическом прибытии и убытии с/на места работы.

5. Выполнить денежный расчет, в соответствии с настройками. Расчет покажет текущую статистику по отработанным часам и заработанным деньгам.
6. Вносить неявки – больничные, отгулы, отпуски.

В настройках возможен выбор метода оплаты труда – оклад или часовая ставка, надбавок – ночные смены, выходные и праздничные дни, работа сверхурочно, вредные условия труда, выслуга лет, региональный коэффициент.

В мобильном приложении будет 7 страниц с которых происходит работа с приложением:

1. AbsenceAddPage_MobileFamily – Страница, с которой пользователь добавляет неявку.
2. AbsenceObserverPage_MobileFamily – Страница, с которой пользователь управляет своими неявками.
3. About_MobileFamily – Страница с информацией о приложении и возможность обратной связи.
4. Calendar_MobileFamily – Страница, где осуществляется основная работа со сменами – добавление, редактирование, удаления.
5. CalculateSalary_MobileFalimy – Страница, где производится денежный расчет на «текущее положение дел».
6. SetCheckedAppointment – Страница, где пользователь устанавливает отметку фактического прибытия/убытия на работу.
7. Settings_MobileFamily – Страница с настройками приложения, в которую входит 2 части. Основная – настройка ночной смены, если вдруг вам нужно учитывать ночную смену по другому часовому поясу, установка будильников и закрепление дополнительного тайла. Зарплатная часть, назовем ее так – установка метода оплаты, количество денег за период времени и установка надбавок, которые применяются.

Все эти страницы используются в главной MainPage_MobileFamily. Она обеспечавает переключение между этими страницами.

Так как основным средством синхронизации были выбраны события в календаре, то возик вопрос – как хранить дополнительные поля у смен? Ответ оказался прост – хранить их в виде JSON текста в стандартном поле события Details. Для реализации этого механизма используются структуры AbsenceCustomData, CustomData и CustomDataAllParameters. А класс AppointmentHelper выполняет операции по сериализации и десериализации данных.

Класс BindigClass обеспечивает доступность общих параметров для всех страниц. А StringValidator обеспечивает проверку корректности ввода пользовательских данных в элементы.

Классы ScheduleAppointmentSmena и AbsenceScheduleAppoinment наследуются от класса ScheduleAppointment, который используется для работы с календарём.

Класс IconConverter используется для управления картинкой-идентификатором у смены. Смена имеет 3 состояния Unchecked, HalfChecked и Checked.

Класс MyMoney обеспечивает функцию расчета денег. Он учитывает параметры, которые были установлены в настройках, учитывает состояние всех смен за текущий месяц, если найдены смены в состоянии не Checked, то будет выдано предупреждение и буден произведен расчет без тех смен, учитывает и события всех типов неявок. Для вычисления является ли день, на который запланирована смена, праздничным используются классы XmlToCalendar, calendar, day, holiday, calendarDay, calendarHoliday. Список праздников получаю из XML файла производственного календаря на текущий год.

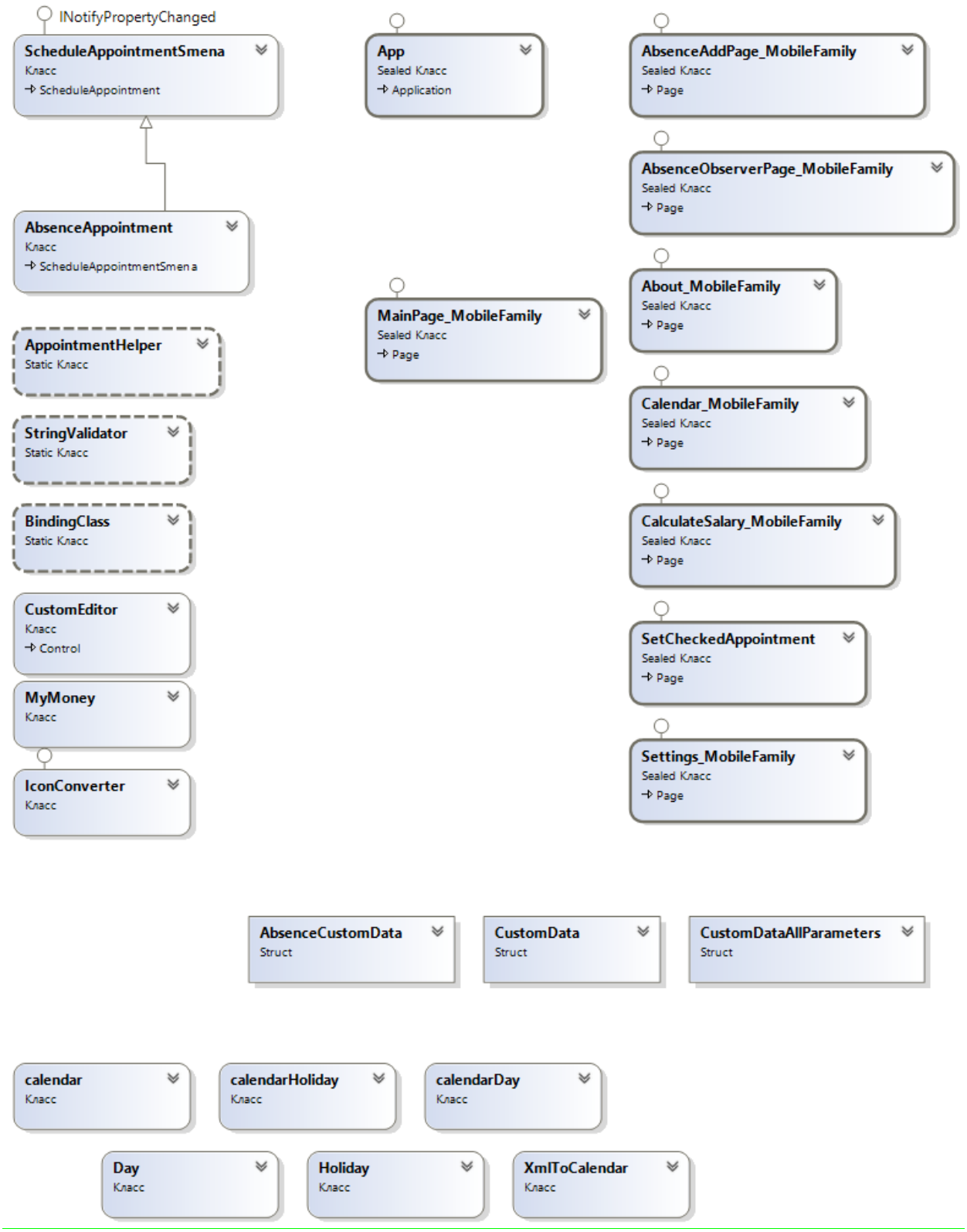
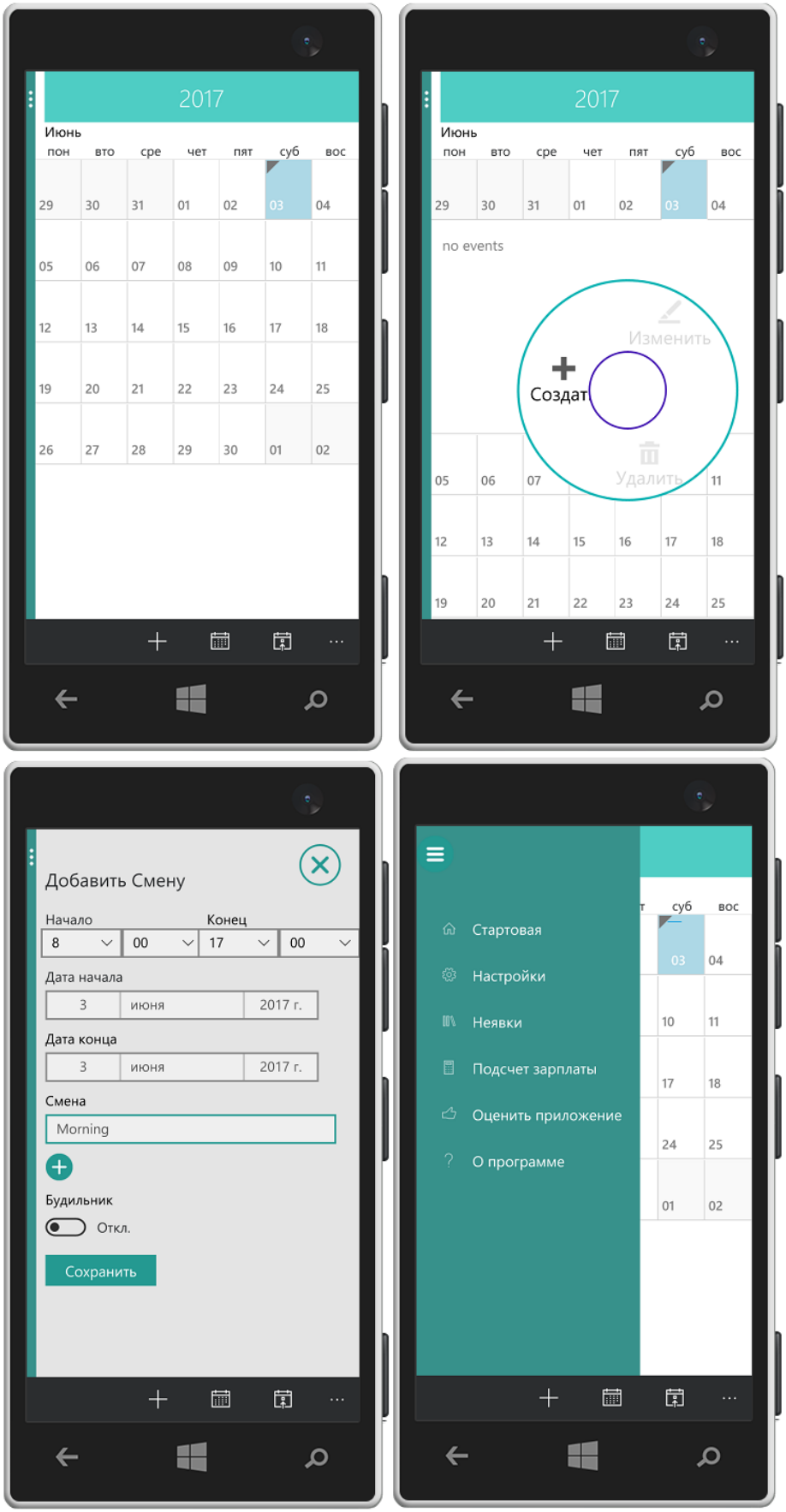
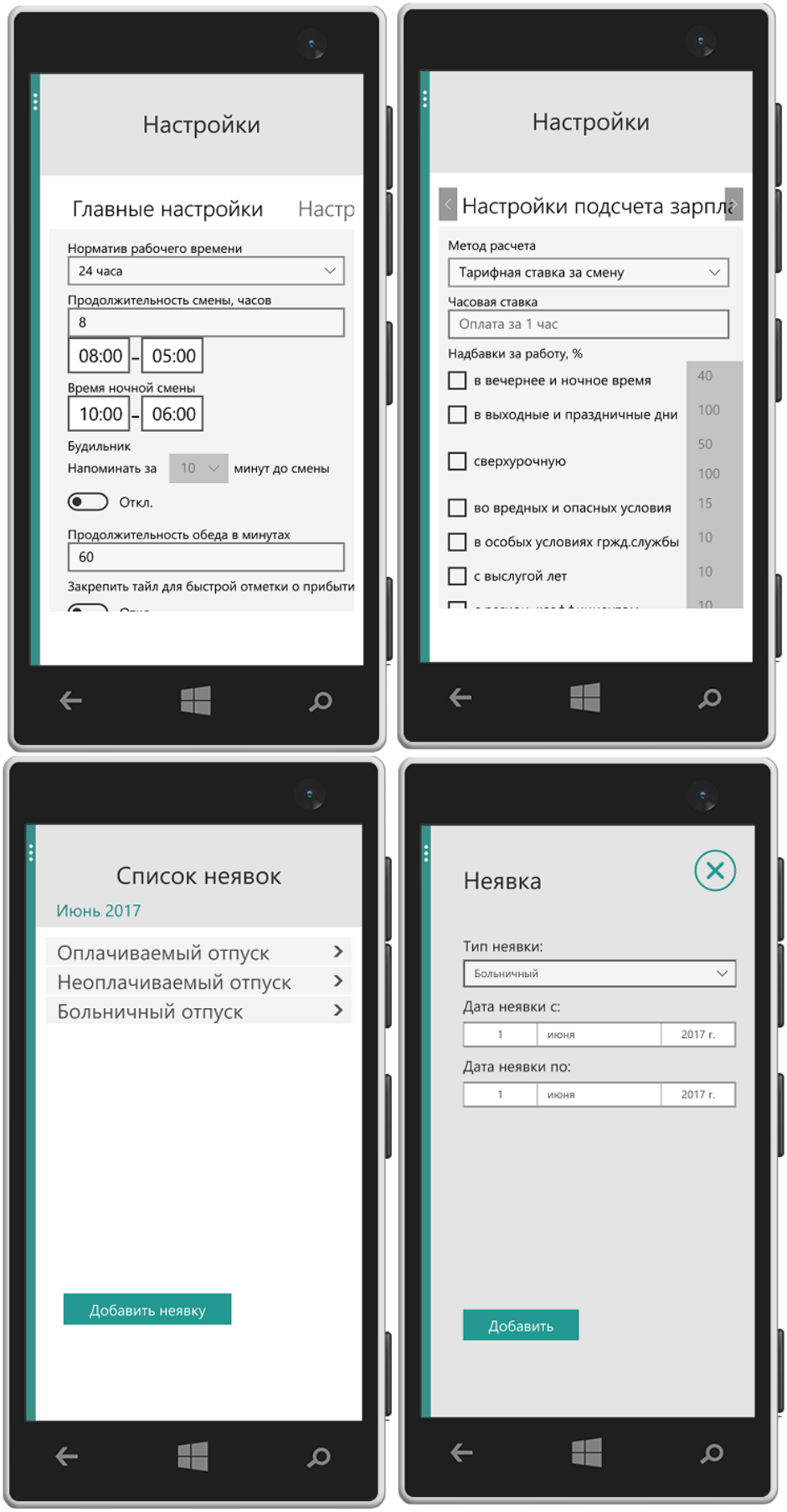
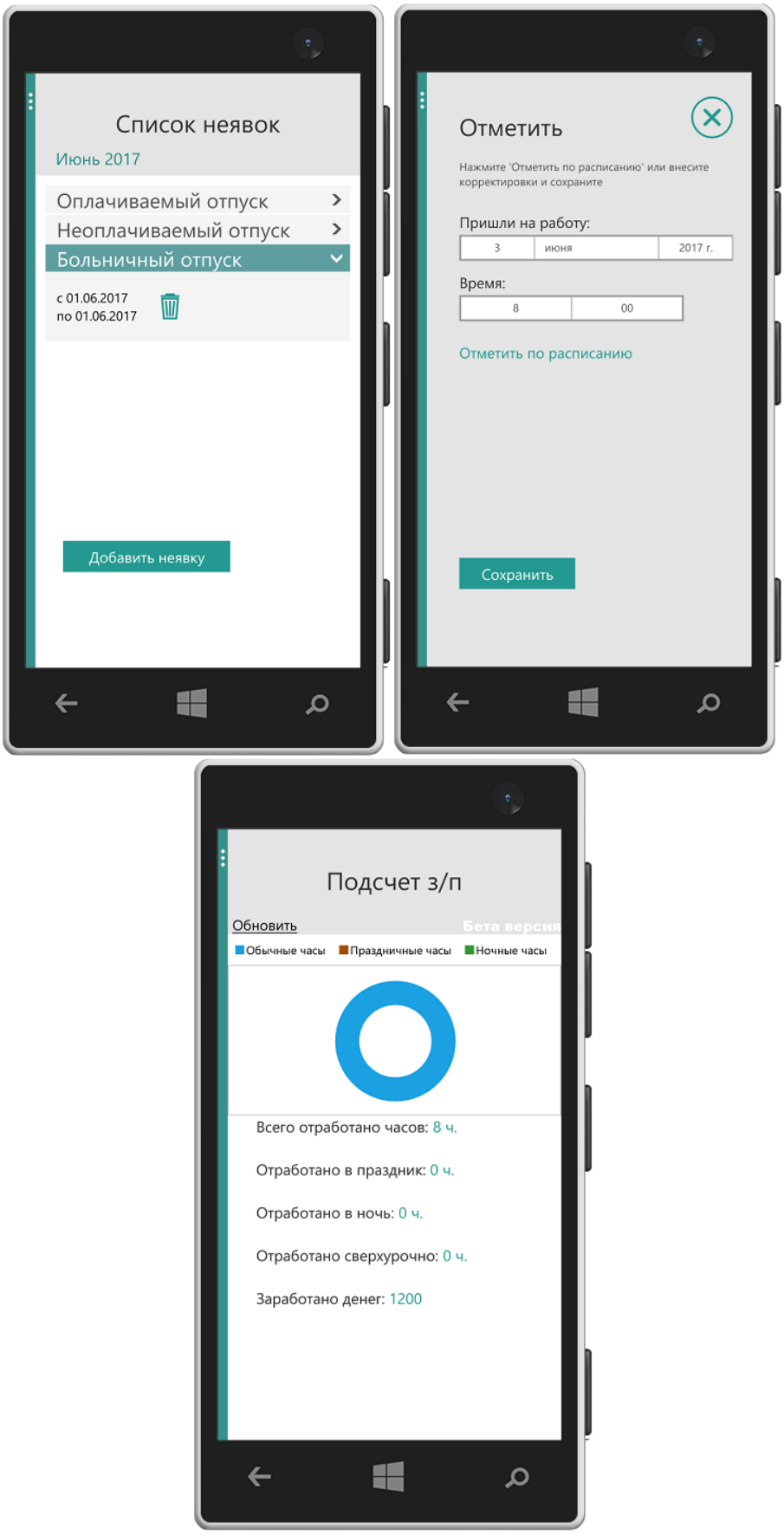


Рисунок 1 – Схема представления классов

Скриншоты приложения с эмулятора:




Настройки
Главные настройки
Настр
Норматив рабочего времени
24 часа
Продолжительность смены, часов
8
08:00
05:00
Время ночной смены
10:00
06:00
Будильник
Напоминать за
10
минут до смены
Откл.
Продолжительность обеда в минутах
60
Закрепить тайл для быстрой отметки о прибыти
Настройки
Настройки подсчета зарпла
Метод расчета
Тарифная ставка за смену
Часовая ставка
Оплата за 1 час
Надбавки за работу, %
в вечернее и ночное время
40
в выходные и праздничные дни
100
сверхурочную
50
100
во вредных и опасных условия
15
в особых условиях гржд.службы
10
с выслугой лет
10
Список неявок
Июнь 2017
Оплачиваемый отпуск
Неоплачиваемый отпуск
Больничный отпуск
Добавить неявку
Неявка
Тип неявки:
Больничный
Дата неявки с:
1
июня
2017 г.
Дата неявки по:
1
июня
2017 г.
Добавить


Список неявок
Июнь 2017
Оплачиваемый отпуск
Неоплачиваемый отпуск
Больничный отпуск
с 01.06.2017
по 01.06.2017
Добавить неявку
Отметить
Нажмите 'Отметить по расписанию' или внесите корректировки и сохраните
Пришли на работу:
3
июня
2017 г.
Время:
8
00
Отметить по расписанию
Сохранить
Подсчет з/п
Обновить
Бета версия
Обычные часы
Праздничные часы
Ночные часы
Всего отработано часов: 8 ч.
Отработано в праздник: 0 ч.
Отработано в ночь: 0 ч.
Отработано сверхурочно: 0 ч.
Заработано денег: 1200

На данный момент реализована часть проекта с мобильным приложением.

Список литературы

1. Develop apps for the Universal Windows Platform (UWP) [Электронный ресурс]. – Режим доступа : https://msdn.microsoft.com/en-us/library/dn975273.aspx, свободный. – Загл. с экрана.
2. Manage appointments - UWP app developer | Microsoft Docs [Электронный ресурс]. – Режим доступа : https://docs.microsoft.com/en-us/windows/uwp/contacts-and-calendar/managing-appointments, свободный. – Загл. с экрана.
3. Триггеры и фоновые задания в приложениях Windows Store / Хабрахабр [Электронный ресурс]. – Режим доступа : https://habrahabr.ru/post/264199/, свободный. – Загл. с экрана.

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ СТАТИЧЕСКОЙ ОБРАБОТКИ ДАННЫХ ОБ ИЗМЕНЕНИЯХ ФАЙЛОВ ПРОЕКТА В РАСПРЕДЕЛЁННОЙ СИСТЕМЕ КОНТРОЛЯ ВЕРСИЙ GIT

Амосов М.С. – студент, Старолетов С.М. – к.ф.-м.н., доцент
Алтайский государственный технический университет (г. Барнаул)

Актуальность

На сегодняшний день в мире пишется большое количество разнообразного программного обеспечения. От небольших скриптов, до гигантских проектов с сотнями тысяч строк кода. Для разработки таких проектов разработчики чаще всего используют системы контроля версий. Git [1] является наиболее популярной из них. Поскольку изменения в проект вносит каждый разработчик, руководителю необходимо знать, кто внёс какой вклад в разработку проекта. Наличие функционала позволяющего решить данную проблему быстро и в удобной форме может существенно улучшить процесс разработки в целом.

Постановка задачи

В процессе разработки программного обеспечения разработчик активно использует систему контроля версий. На протяжении всего рабочего дня он делает изменения в коде, добавляет новый функционал, устраняет ошибки, подтягивает изменения других разработчиков и т.п. Соответствующая система контроля версий (в нашем случае Git) сохраняет всю информацию о проделанной работе разработчика в виде отдельных коммитов, хранящих в себе количество добавленных строк, удалённых строк, данные об авторе и т.п.

Команда по разработке ПО может состоять из нескольких разработчиков (в зависимости от размеров компании 3-5,5-7 и т.д.). К каждой такой команде прикреплены соответствующие ей разрабатываемые проекты. Одному проекту соответствует одно или несколько (если проект большой) репозиториев, с которыми соответственно и работают эти разработчики, ежедневно добавляя/изменяя функционал и синхронизируя изменения друг с другом.

Менеджер проекта по разработке ПО может использовать данные из системы контроля версий для анализа работы конкретного разработчика или группы за день/месяц/год и таким образом делать выводы о том, кто качественно выполняет свою работу, а кто “халтурит”.

Формализация функциональности

Данное программное обеспечение предназначено для команд разработки с целью отслеживания прогресса разработчиков. Может быть использована менеджерами проектов или тим-лидерами.

Приложение должно предоставлять возможность анализа файлов выбранного репозитория. Выдавать пользователю основную информацию о репозитории, авторах и их коммитах.

Должна быть возможность интеграции базы данных со следующими сущностями:

- Команда разработчиков
- Проект (распределяется командам для последующей разработки)
- Репозиторий (место, где хранятся все файлы проекта)
- Сам разработчик

Программа должна предоставлять возможность редактирования, добавления и удаления информации основных сущностей базы данных описанных выше.

Задается диапазон дат (по умолчанию, за вчерашний день). Выбирается проект (или все проекты). Необходимо выяснить прогресс разработчика (вклад) по указанному проекту (проектам).

Примеры результатов анализа:

- Количество коммитов (с указанием проектов)
- Среднее количество затронутых изменений линий кода в коммите
- Количество pull request, которые применил (merge) данный разработчик
- Промежутки времени, в которые производились коммиты с перекрытием (с 9:01 до 9-15 и с 20:00 до 20:07) (чтобы проверять, что разработчик не работает по ночам и т.д.)
- Количество pull-request, которые приняли (merge) другие у данного разработчика
- Количество pull-request, которые сделал разработчик, но другие их не применили (merge)
- Количество строк, добавленных разработчиком
- Количество строк, удаленных разработчиком
- Количество строк, измененных разработчиком
- Количество коммитов, которые являются фиксами багов (в описании коммита указано «fix bug» или другие словосочетания из списка)
- Количество багов, которые допустил данный разработчик (есть коммиты любых за это время с сообщениями вида «fix bug», меняющие линии, которые были добавлены в результате коммита данного разработчика)

По результатам анализа должен считаться взвешенный рейтинг (по всем показателям, у каждого показателя свой коэффициент, который задается) по разработчикам, лучший и худший разработчик команды за период.

Предлагаемое решение поставленной задачи

Для решения поставленной задачи планируется написать desktop-приложение на кроссплатформенном языке Java. Подсчёт статистики в приложении будет осуществляться с помощью двух внешних библиотек: JGit и EGit.

JGit [2] - относительно полнофункциональная реализация Git, написанная изначально на Java, и широко используется в Java-сообществе. С помощью неё будет организован основной подсчёт статистики. (Анализ коммитов, авторов, репозиториев и т.п.).

EGit [3] - плагин Eclipse для работы с репозиториями Git. Некоторые его модули основаны на библиотеке JGit. Из EGit нам понадобится лишь основной его модуль под названием “org.eclipse.egit.core”. С помощью него мы сможем работать непосредственно с api сервиса github.com и брать необходимые нам данные.

На рисунке 1 представлены сценарии использования, (англ. Use Case), изображающий основные функциональные требования данной системы.

Из данных сценариев использования видно, что программа будет предоставлять пользователю возможность как редактирования основных сущностей базы данных, так и возможность просмотра различного рода статистики этих сущностей. Также предусмотрена возможность персональной настройки приложения “под себя”.

Рисунок 1 – Сценарии использования приложения для анализа репозиториев

На рисунке 2 представлена физическая схема базы данных. На ней отображены основные сущности при командной разработке ПО (”Разработчик”, “Команда”, “Роль”, “Репозиторий”, “Проект”) и связи между ними.

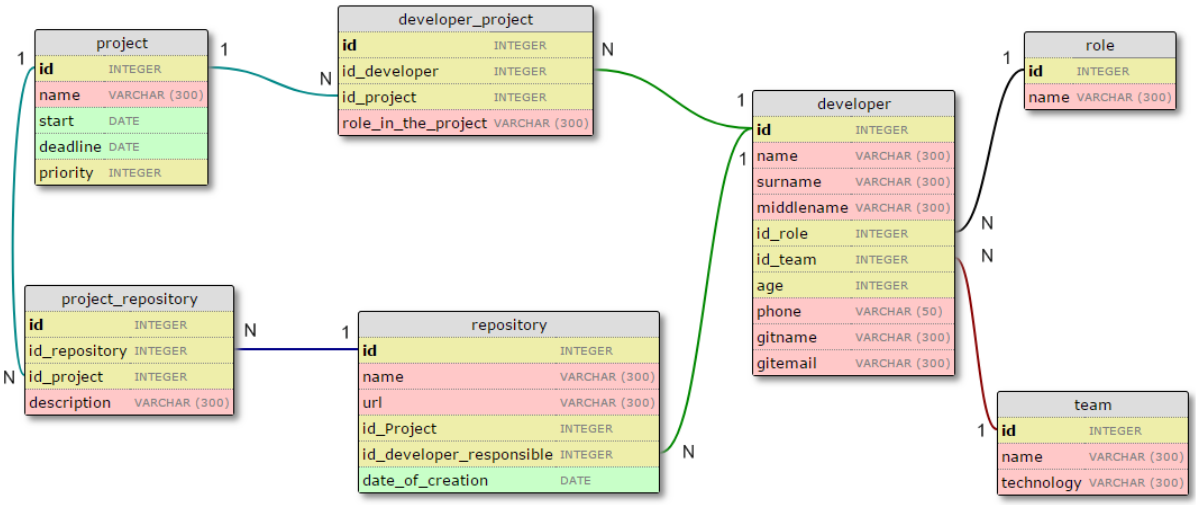


Рисунок 2 – Физическая схема базы данных [5]

На рисунках 3 и 4 представлены скриншоты пользовательского интерфейсов, отображающих некоторые примеры статистических данных в виде графиков.

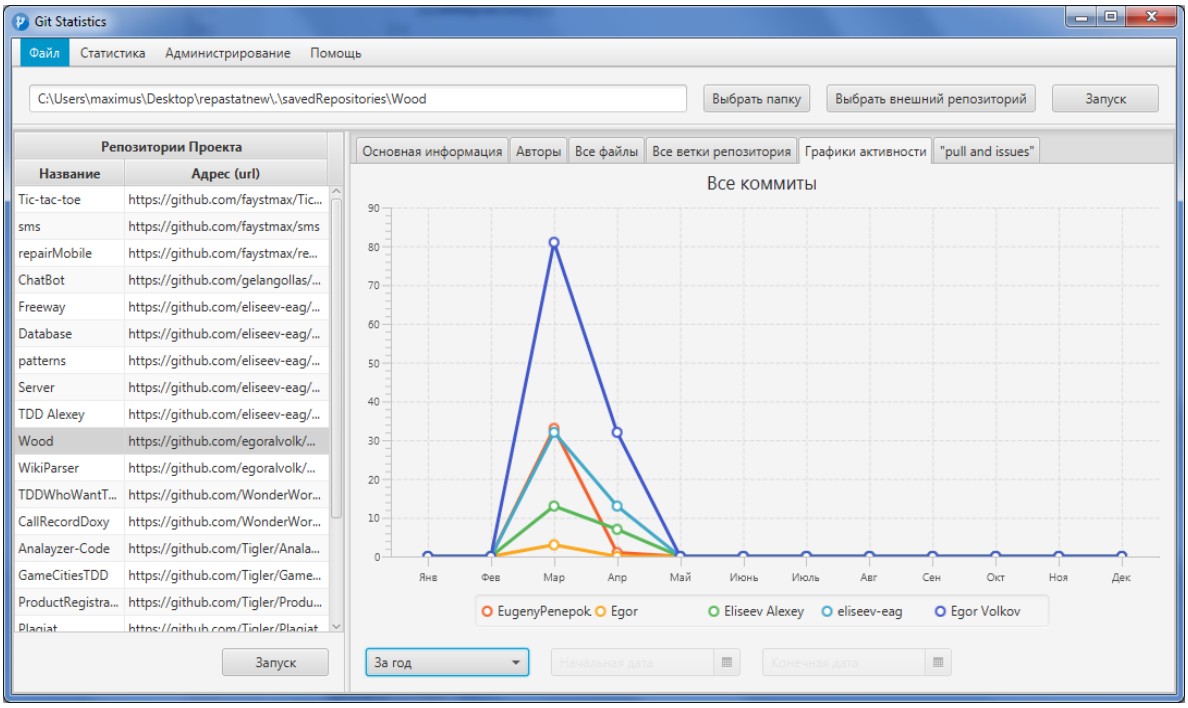


Рисунок 3 – График активности разработчиков в команде

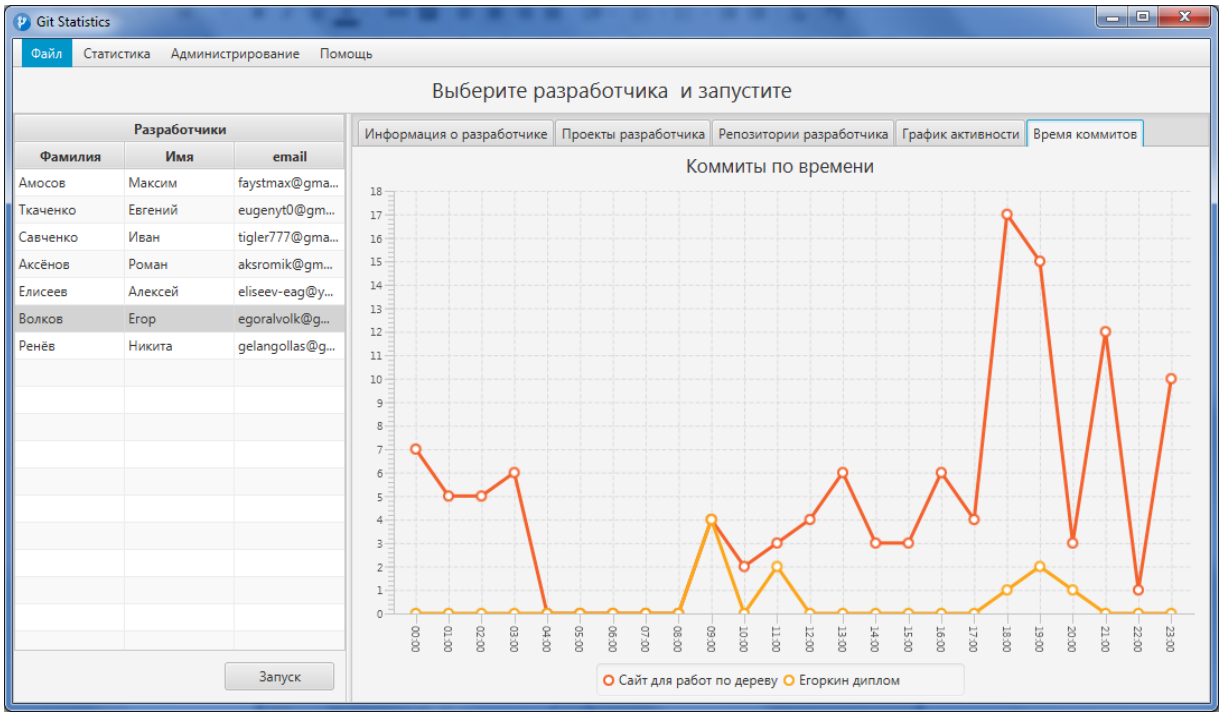


Рисунок 4 – График времени коммитов разработчика

Заключение

Данный программный продукт, реализует статическую обработку данных об изменениях файлов проекта в системе контроля версий Git. Программа имеет современный мощный пользовательский интерфейс и предоставляет пользователю возможность по управлению базой данных разработчиков/команд и их проектов, а также предоставляет большой набор различного рода статических данных.

Стоит отметить, что при написании программы были использованы современные паттерны проектирования [4]: “Мост”, ”MVC”, ”Информационный эксперт”, ”Простая фабрика”, ”Фасад”. В связи с этим расширение функционала либо изменение существующего не составит существенных трудностей для разработчика в будущем.

Список литературы

1. Официальная страница Git [Электронный ресурс]. – Режим доступа: https://git-scm.com/, свободный.
2. Документация JGit [Электронный ресурс]. – Режим доступа: https://eclipse.org/jgit/documentation/ , свободный.
3. Документация EGit [Электронный ресурс]. – Режим доступа: https://www.eclipse.org/egit/documentation/ , свободный.
4. Крючкова Е.Н., Старолетов С.М. Архитектурное проектирование и паттерны программирования [Электронный ресурс]: Учебно-методическое пособие.— Электрон. дан.— Барнаул: АлтГТУ, 2015.— Режим доступа: http://new.elib.altstu.ru/eum/download/pm/Krutkova-Patterns.pdf, авторизованный.
5. WWW SQL Designer - дизайнер моделей баз данных [Электронный ресурс]. – Режим доступа:http://ondras.zarovi.cz/sql/demo/, свободный.

ПРОБЛЕМЫ ПРИ РАЗРАБОТКЕ ДИАЛОГОВЫХ СИСТЕМ НА ЕСТЕСТВЕННОМ ЯЗЫКЕ

Головин М.А. – магистрант, Крючкова Е.Н. – к.ф.-м.н., профессор
Алтайский государственный технический университет (г. Барнаул)

С распространением электронных устройств сильнее проявилась проблема их использования: далеко не каждый человек может за разумное время научиться пользоваться стандартным интерфейсом. Особенно это относится к пожилым людям. Одним из самых разумных подходов к этой проблеме – научить компьютер распознавать команды на естественном для человека языке. При хорошей реализации этого подхода нагрузка на пользователя и требования к нему существенно снизились бы.

Понимание компьютером простых команд на естественном языке – задача, на данный момент успешно реализуемая. Но такой подход всё равно налагает на пользователя определённые ограничения, ему приходится знать хотя бы примерный формат команд, а также становится трудно уточнить задачу.

Диалог – функциональная разновидность языка, реализующаяся в процессе непосредственного общения между собеседниками и состоящая из последовательного чередования реплик [1]. К основным языковым особенностям диалога относятся: обилие вопросительных и побудительных предложений, наличие повторов и переспросов, синтаксическая неполнота реплик, компенсируемая за счёт предыдущих высказываний.

Диалоговые системы – важная составляющая современных информационных систем, большинство взаимодействий с пользователями осуществляются с применением диалогов. Можно выделить основные способности такой системы:

- понимать обращённые к ней реплики
- формировать ответные реплики в соответствии с логикой и динамикой диалога
- позволять пользователю достигать своих прагматических целей через диалог

Результатами диалога могут быть какие-либо выполняемые системой действия, побуждением к которым для системы были высказывания (команды) пользователя, передача

и уточнение параметров действий, выдача пользователю необходимой информации в процессе диалога [2].

Это всё верно в большей степени для классических диалоговых систем, в которых диалог идёт по определённому сценарию, варианты взаимодействия очень сильно ограничены. В таких системах возможные вопросы пользователя и варианты ответа на них обычно хранятся во внутренней базе данных, или ведётся поиск по внешней базе, в интернете (из недавних примеров – система Watson от компании IBM).

Недостатки подобных систем очевидны: это малая вариативность, невозможность отхода от контекста конкретной предметной области базы данных, зачастую – невозможность поддержания контекста диалога, также система не может отвечать на общие вопросы (без конкретных действующих лиц). В таких случаях функционал системы обычно ограничивается величиной базы, поддержание таких систем трудоёмко, а расширение круга задач зачастую невозможно.

Более интересны так называемые интеллектуальные диалоговые системы (ИДС), часто именуемые понятиями «чат-бот» и «виртуальный собеседник». Основная задача таких систем – ведение диалога с человеком на естественном языке в форме, максимально приближенной к общению между людьми.

Сфера применения таких систем очень широка:

1. Построение интерфейсов для систем поиска и анализа информации.
2. Системы поддержки и консультирования клиентов (интеллектуальные экспертные системы).
3. Автоматизированное тестирование знаний.
4. Создание виртуальных собеседников общего пользования.

Одним из наиболее важных преимуществ диалоговых систем, при условии использования распознавания и синтеза речи, является отсутствие необходимости взаимодействовать с компьютером напрямую. Они часто используются через телефон, а также могут не занимать глаза и руки, что критично для некоторых областей (управление автомобилем, медицинские операции).

К подобным системам выдвигаются более жёсткие требования:

1. ИДС должна иметь хотя бы примитивный механизм самообучения.
2. ИДС должна уметь конструировать ответ на естественном языке, не просто выдавать готовый из базы.
3. ИДС должна уметь поддерживать беседу на заданную тему. Например, задавать вопросы, спорить, иметь и отстаивать точку зрения.
4. ИДС должна обладать некоторой индивидуальностью и эмоциями.

Нужно сразу отметить, что создание ИДС полностью соответствующей заданным требованиям – AI-полная проблема, т.е. требует создания искусственного интеллекта. Даже задача по корректному семантическому разбору естественной речи уже является AI-полной. Поскольку до создания подобных систем современной компьютерной науке далеко, мы не будем ставить целью удовлетворение всех вышеописанных условий.

Тем не менее, в современной науке и культуре именно разработку полнофункциональной ИДС чаще всего связывают с разработкой искусственного интеллекта. Широко известен «тест Тьюринга», в котором человеку, общающемуся с людьми и прототипами ИДС в текстовом виде, необходимо определить, кто из собеседников действительно является человеком. Существует ежегодная премия Лёбнера, которая вручается разработчику программы, наиболее успешно проходящей тест Тьюринга. У данного подхода есть существенные недостатки, один из них – имитация диалога с реальным человеком за счёт моделирования излишне эмоционального поведения, что, по мнению многих исследователей, нисколько не приближает нас к созданию искусственного интеллекта.

Бьорн Брингерт [3] выделяет три основные проблемы современных диалоговых систем и речевых приложений:

a) они недостаточно естественны

b) они недостаточно удобны, ограниченны в использовании

c) они дороги в разработке

Все эти причины взаимосвязаны и взаимозависимы: современные системы недостаточно удобны т.к. недостаточно естественны, а разработка более естественных систем дорога. Норберт Флегер [4] отмечает, что многие исследовательские прототипы так и не получили развития в промышленных системах: «Современные коммерческие диалоговые системы обладают очень ограниченными возможностями при работе с естественным языком. Многие из этих систем требуют от пользователя сокращённый и упрощенный набор команд».

На данный момент отсутствует модель диалога и соответствующий ей диалоговый движок, которые возможно использовать в разных проектах, чтобы не начинать каждый раз с нуля. Так же не существует должной среды разработки для подобных систем [5]. В связи с возникающими сложностями, разработчики делают упор на простых диалогах, где пользователю задаются вопросы на которые необходимо дать короткие однозначные ответы.

Одной из первых диалоговых систем на естественном языке принято считать программу ELIZA (Элиза) [6]. Наиболее известен сценарий «Доктор» для этой системы, при котором пародируется общение пользователя с психотерапевтом. Основным принципом действия программы является сопоставление с образцом. Перед началом работы в систему добавляется скрипт – сценарий, база заготовленных правил разбора и преобразования реплик. Входная реплика системы определённым образом преобразуется, выделяются ключевые слова, по которым из скрипта выбирается наиболее подходящее правило. На сегодняшний день, большинство систем, создатели которых получали премию Лёбнера, работают по этому принципу. Но до прохождения теста Тьюринга им далеко – большинство среднестатистических пользователей сможет определить природу своего собеседника даже в коротком диалоге.

Промышленным стандартом для построения голосовых диалоговых систем является VoiceXML. Как понятно из названия, для описания скрипта диалога используется язык XML, скрипты генерируются web-сервером в виде HTML. Поддерживается синтез голоса на основе текста, а также проигрывание аудиофайлов. Но используемые в системе скрипты не используют даже правил преобразования, команды имеют достаточно жёсткий формат. Поэтому обычно VoiceXML используется для построения меню для систем телефонии, при любом взаимодействии, выходящем за рамки сценария, идёт переадресация на оператора.

Наиболее интересным из известных проектов является NADIA (Natural DIAlog system) – концепт диалоговой системы, разработанный Маркусом М. Бергом в 2014 году [7]. Основной задачей разработки стояло построение платформы более функциональной чем VoiceXML-подобные, но при этом пригодной для промышленного использования, в отличие от ELIZA-подобных. Для достижения этого предлагается разработка диалогового движка, способного работать с абстрактным представлением вопросов и ответов.

Какие же основные отличия естественных языков от формальных, классически используемых для программирования вычислительных машин и создания интерфейсов? Перечислим основные:

- многие слова имеют не одно, а несколько значений
- значения отдельных слов и выражений зависят от их окружения (контекста)
- распространены синонимы и омонимы
- многие правила употребления слов явно не сформулированы, для каждого правила есть исключения

Как уже было выше упомянуто, проблема разбора и формализации фраз на естественном языке является AI-полной, поэтому распространены различные эвристические подходы к

ней. Задачу синтаксического разбора можно считать практически решённой для большинства распространённых языков. С анализом смысла (семантики) – совсем другая ситуация. Для интеллектуальной диалоговой системы необходимо моделирование максимального количества аспектов языка, даже с современными вычислительными мощностями это слишком тяжёлая задача, многозначность естественных языков делает работу подобных алгоритмов слишком медленной. Также очень сильно возрастает сложность построения баз знаний, которые обычно строят вручную специалисты по лингвистике.

Проблема управления ведением диалога является не менее сложной. Несмотря на наличие лингвистических диалоговых теорий и классификаций, построить формализованную модель не представляется возможным. Тем не менее, стоит выделить теорию диалоговых актов, систему классификации функций в диалогах DIT++, теорию вежливости и теорию связных пар [8-11]. Разработки в этой области потенциально могут служить основой для построения эвристических алгоритмов управления диалогом.

Список литературы

1. Всеволодова А.В. Компьютерная обработка лингвистических данных : учеб.пособие / А.В. Всеволодова. – 2-е изд., испр. – М.: Наука : Флинта, 2007. – 96 с.
2. Филиппович Ю.Н. Интеллектуальные технологии и системы. Сборник учебно-методических работ и статей аспирантов и студентов. Выпуск 8 / Сост. и ред. Ю.Н. Филипповича. – М.: НОК «CLAIM», 2006. – 326 с
3. Bjorn Bringert – Programming Language Techniques for Natural Language Applications. PhD thesis, University of Gothenburg, 2008.
4. Norbert Pfleger. Context-based multimodal interpretation: an integrated approach to multimodal fusion and discourse processing. PhD thesis, Universität des Saarlandes, 2008.
5. James R. Glass and Eugene Weinstein. Speechbuilder: Facilitating Spoken Dialogue System Development. In Proc. of the 7th European Conference on Speech Communications and Technology, INTERSPEECH, pages 1335–1338, 2001.
6. Joseph Weizenbaum. ELIZA - A computer program for the study of natural language communication between man and machine. Commun. ACM, 9 (1) : 36–45, Jan 1966. ISSN 00010782.
7. Markus Berg. Natürlichsprachlichkeit in Dialogsystemen. Informatik-Spektrum, 36 (4) : 371–381, 2013. ISSN 0170-6012
8. Searle J. A taxonomy of illocutionary acts. In K. Gunderson, editor, Language, Mind, and Knowledge, pages 344–369. University of Minnesota Press, 1975.
9. Harry Bunt et al. Towards an ISO Standard for Dialogue Act Annotation. In Nicoletta Calzolari et al., editors, 7th International Conference on Language Resources and Evaluation, Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC 2010), Valletta, Malta, 2010. European Language Resources Association (ELRA).
10. Penelope Brown and Stephen C. Levinson. Politeness: Some Universals in Language Usage (Studies in Interactional Sociolinguistics). Cambridge University Press, 1987.
11. Schegloff E.A. Sequence Organization in Interaction: A Primer in Conversation Analysis, volume 1 of Sequence Organization in Interaction: A Primer in Conversation Analysis. Cambridge University Press, 2007. ISBN 9780521532792.

МОДЕЛЬ ДЛЯ МОДЕЛИРОВАНИЯ ПОВЕДЕНИЯ ЖИВЫХ ОБЪЕКТОВ НА ОСНОВЕ ВЗАИМОДЕЙСТВУЮЩИХ РАСШИРЕННЫХ ВЕРОЯТНОСТНЫХ АВТОМАТОВ

Шабанов П.А.– студент, Старолетов С.М. – к.ф.-м.н., доцент
Алтайский государственный технический университет (г. Барнаул)

Актуальность работы

В условиях современного рынка в частности игровой индустрии ключевым значением выступает время разработки продуктов. И иногда перед ставится задача по моделированию одного или нескольких живых объектов и их взаимодействия. Надо сказать, что моделирование поведения объектов, в частности живых, очень трудоемкий и сложный процесс. И зачастую тратится много человеко-часов, чтобы смоделировать простейший объект, что приводит к большим финансовым затратам.

Нами разрабатывается программный фреймворк, который позволит быстро и легко создавать модели поведения животных, что позволит сэкономить время программиста, а также позволит компании ускорить темпы развития своего приложения, использующего взаимодействующие объекты.

Для моделирования живых объектов предлагается использовать расширенные конечные вероятностные конечные автоматы.

Математическая постановка задачи

В рамках данной работы живой объект (животное) представляется как некоторая абстрактная система, разбитая на множество подсистем, решающих свои конкретные задачи. Помимо этого, каждая подсистема способна влиять на другие подсистемы данной системы по средствам механизма общения характеристик животного между собой.

В связи с чем предлагается самостоятельная автоматная теоретико-множественная модель на разных уровнях аналогично способу построения модели в [1] для описания поведений живых организмов, состоящая из трех уровней абстракции:

- мир (World);
- животное (Animal);
- подсистема животного (SubSystem).

Рассмотрим первый уровень. Мир в предлагаемой вероятностной модели задается как кортеж:

$$World = (Animal^*, SubScribes),$$

где Animal – животное;

Animal* = Animal x Animal…x Animal – итерация (далее, набор) животных;

SubScribes – множество подписчиков животных, выступающих наблюдателями за другими животными, определяется как отображение:

$$SubScribes: Animal \rightarrow Animal^*.$$

Животное (второй уровень) в предлагаемой вероятностной модели является следующим уровнем абстракции и задается как кортеж:

$$Animal = (S, s_0, S_{fin}, \gamma, SubSystem^*, Communicator),$$

где S – множество допустимых состояний животного, например:

$$S \in \{Start, Stand, Run, Go, Sit, Lies, Sleep, Die\},$$

Start – состояние, идентифицирующее, что животное начало жизнедеятельность;

Stand – состояние, идентифицирующее, что животное стоит;

Run – состояние, идентифицирующее, что животное бежит;
Go – состояние, идентифицирующее, что животное идет;
Sit – состояние, идентифицирующее, что животное сидит;
Lies – состояние, идентифицирующее, что животное лежит;
Sleep – состояние, идентифицирующее, что животное спит;
Die – состояние, идентифицирующее, что животное умерло;
s_0 – начальное состояние автомата;
S_{fin} – множество заключительных состояний;
γ – недетерминированная функция переходов, определяемая как отображение:

$$\gamma: S\times Speed\times Weight\times HeartBeat\times Pressure\times P \rightarrow 2^{S\times T\times Speed\times Weight\times HeartBeat\times Pressure}.$$

Находясь в состоянии из S с заданными скоростью движения Speed $\in$ R, весом Weight $\in$ R, сердцебиением HeartBeat $\in$ R, давлением Pressure $\in$ R и вероятностью P $\in$ R, $0 \leq P \leq 1$, модель животного недетерминированно переходит после временной задержки T $\in$ R в состояние с измененными скоростью движения Speed, весом Weight, сердцебиением HeartBeat, давлением Pressure.

SubSystem* - подсистемы животного, являющиеся низшим уровнем абстракции, решают конкретные задачи функционирования животного.

Подсистема (третий уровень) задается как кортеж:

$$SubSystem= (Type, Scale^*),$$

где Type – тип подсистемы:

$$Type \in \{Muscular, Motor, \ldots\},$$

(Muscular - мускульная, Motor - двигательная и т.д.).

Scale* - набор допустимых шкал системы, которые являются способом инкапсуляции характеристик животного, каждая из которых задается как кортеж:

$$Scale = (Name, Ch_{max}, Ch_{min}, Ch_{current}, Percent, T_{ожид.}),$$

где Name - название шкалы;
Ch_{max} – число, идентифицирующее максимальное значение шкалы;
Ch_{min} – число, идентифицирующее минимальное значение шкалы;
$Ch_{currunt}$ – число, идентифицирующее текущее значение шкалы, причем: $Ch_{current} \in R$, $Ch_{min} \leq Ch_{current} \leq Ch_{max}$;

Percent – число, идентифицирующее процентную составляющую текущего значения шкалы относительно отрезка, в котором она может принимать свое значение;
$T_{ожид}$ – число, идентифицирующее время отклика данной шкалы.

Также на этом уровне абстракции действует коммуникатор Communicator – посредник, обеспечивающий взаимодействие множеств Scale, задается как итерация из кортежей:

$$Communicator = (F, Scale)^*,$$

где F – функция изменения шкал, определяемая как отображение текущей шкалы в измененную шкалу:

$$F: Scale \rightarrow Scale.$$

Дальнейшая работа

После создания данной автоматной модели она будет реализована в виде объектно-ориентированного фреймворка с использованием паттернов проектирования [2].

Список литературы

1. Старолетов С. М. Моделирование распределенных многокомпонентных программных систем и их тестирование на основе автоматных вероятностных моделей / Монография. С.М. Старолетов, Е. Н. Крючкова. – Барнаул : Изд-во АлтГТУ, 2011. – 107 с. – ISBN 978-5-7568-0859
2. Крючкова Е. Н., Старолетов С.М. Архитектурное проектирование и паттерны программирования [Электронный ресурс]: Учебно-методическое пособие. − Электрон. дан.− Барнаул: АлтГТУ, 2015.− Режим доступа: http://new.elib.altstu.ru/eum/download/pm/Krutkova-Patterns.pdf, авторизованный

РАЗРАБОТКА КРОССПЛАТФОРМЕННОГО ТЕКСТОВОГО РЕДАКТОРА С КОМБИНИРОВАННЫМ ТЕКСТО-РУКОПИСНЫМ ВВОДОМ

Галкин Р.Е. − студент, Андреева А.Ю. − к.ф.-м.н., доцент
Алтайский государственный технический университет (г. Барнаул)

У многих пользователей мобильных платформ так или иначе появляется проблема выбора наилучшего ПО для различных целей, таких как просмотр документов, ведение заметок и т.д. Данная проблема касается и выбора текстового редактора, в котором бы удобно сочетались различные функции. Нередко происходит ситуация, когда кроме текста необходимо быстро записать формулу, зарисовать схему. Популярные текстовые редакторы не всегда предоставляют ввод формул, а в редакторах, которые оснащены данной функцией ввод формул - это целая задача, так как необходимо выбирать служебные символы, индексы и т.п. Зачастую ввод формул представлен своего рода конструктором выражений. Данный метод обеспечивает достаточно медленный ввод в текстовом редакторе и не может быть использован для ведения, например, конспекта лекций. Гораздо быстрее использовать рукописный ввод, но тут пользователя ожидает другие проблемы. Популярные текстовые редакторы предоставляют несколько режимов рукописного ввода:

1. Свободное "рисование" на листе. Этот режим плох тем, что пользователю, используя тачскрин устройства, очень сложно вводить текст, размером с печатные символы из-за толщины пальцев. Рукописно-введенная информация в таком режиме либо будет очень крупная, либо, если пользователь пытается вводить её размером с обычный напечатанный символ, нечитаемая.

2. Предоставление блоков со свободным рисованием. Данный режим хоть и наследует проблемы первого, но всё же немного удобнее вследствие с возможностью организации таких блоков по листу. В них всё же можно писать укрупненно. Но к примеру лекцию с обилием формул данный режим превратит в огромный документ, читать который будет не очень удобно.

Исходя из проблем перечисленных решений, можно сделать вывод, что наиболее удобный вариант для режима рукописного ввода - это ввод информации в "крупном" формате с последующем её масштабированием в размеры строчки простого печатного текста. Найти приложение, которое бы обеспечивало данную функцию в магазинах

приложений таких как AppStore и GooglePlay не удалось, отсюда и появляется задача работы, а именно - реализация приложения с описанной возможностью.

Наиболее близкая целевая аудитория этого приложения - студенты, которые могут использовать данное ПО на своих планшетных ПК для ведения лекций. Сейчас почти у каждого есть планшетный ПК, который люди так или иначе используют для оптимизации повседневной жизни. Следовательно ведение лекций студентами так же можно оптимизировать благодаря удобному текстовому редактору с файловой системой для хранения лекций.

Так как речь идет о мобильном приложении, необходимо выбрать целевую платформу. По данным StatCounter имеются следующие статистические данные по платформам: WindowsPhone 1.75% устройств, iOS - 20.07% и Android - 68.54% устройств. Интерес вызывают, очевидно две платформы, но две платформы - это два приложения с разным исходным кодом. Поэтому было принято решение разрабатывать кроссплатформенное приложение, что позволит не только по сути разрабатывать одно приложение, но и охватить 1.75% пользователей с WindowsPhone.

В качестве основного инструмента для создания одного приложения под различные ОС, принято решение использовать PhoneGap. PhoneGap (называемый также Apache Callback, основанный на Apache Cordova) — бесплатный open-source фреймворк для создания мобильных приложений, созданный Nitobi Software. Позволяет создать приложения для мобильных устройств используя JavaScript, HTML5 и CSS3, без необходимости знания «родных» языков программирования (например, Objective-C), под все мобильные операционные системы (iOS, Android, Bada и т. д.). Готовое приложение компилируется в виде установочных пакетов для каждой мобильной операционной системы. Фреймворк хорош тем, что мы можем использовать различный JS библиотеки в разработке. Так например для разработки пользовательского интерфейса используется SemanticUI. Для реализации главной особенности редактора - рукописный ввод с масштабированием в строку используется Canvas. На начальном этапе разработки использовался простой элемент Canvas HTML5, но изображения на нем растровые, что сказывается на их качестве. Для решение данной задачи использована библиотека Paper.js, которая позволяет использовать Canvas с векторной графикой. PhoneGap в качестве основного языка разработки предполагает JavaScript, но в связи со слабо реализованным в нем ООП, принято решение использовать TypeScript.

TypeScript — язык программирования, представленный Microsoft в 2012 году и позиционируемый как средство разработки веб-приложений, расширяющее возможности JavaScript. TypeScript является обратно совместимым с JavaScript и компилируется в последний. Фактически, после компиляции программу на TypeScript можно выполнять в любом современном браузере. Так же выбор в пользу TypeScript обусловлен наличием в нем классов для работы с файловой системой, несмотря на web-ориентированность.

В результате работы была спроектирована следующая архитектура приложения, представленная на рисунке 1.

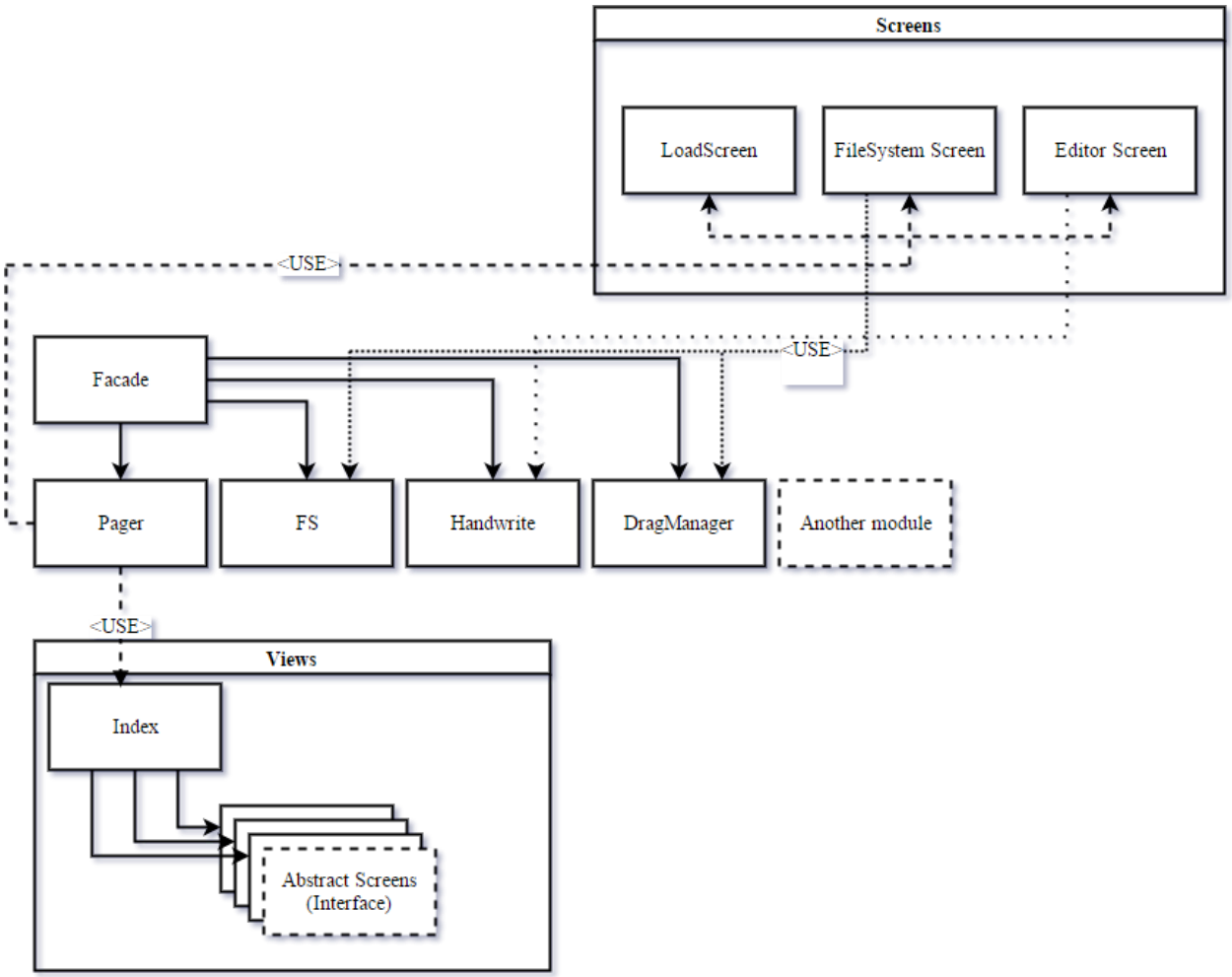


Рисунок 1 – Схема архитектуры приложения

Заметим, что получена необходимая модульность приложения, благодаря которой редактор можно будет легко оснастить различными дополнительными функциями, даже предполагающими отдельный экран.

На данный момент полностью готов прототип с реализованной файловой системой и комбинированным тексто-рукописным вводом. Демонстрация работы описанного тексто-рукописного режима ввода представлена далее подробна на схеме (рисунок 2) и скриншотах (рисунок 3, рисунок 4) непосредственно с приложения.

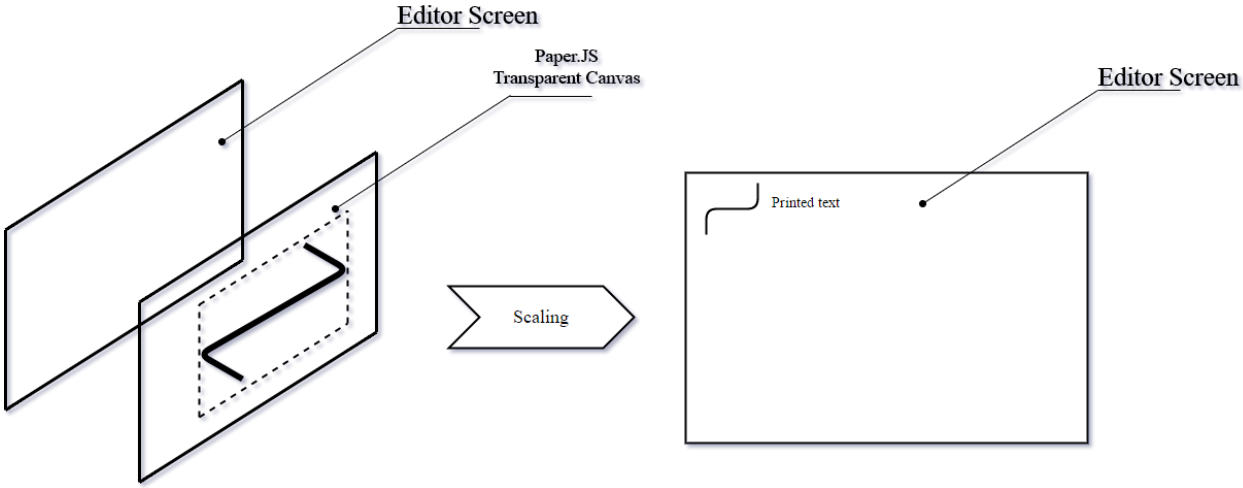


Рисунок 2 – Схема работы тексто-рукописного режима ввода

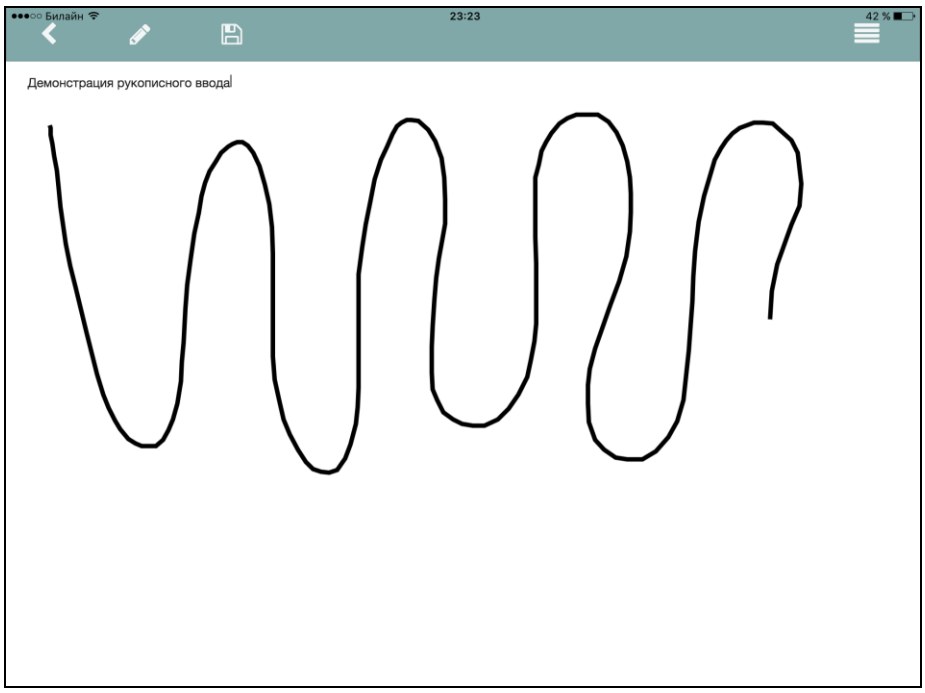


Рисунок 3 – Демонстрация работы тексто-рукописного режима ввода. Стадия до масштабирования.

Рисунок 4 – Демонстрация работы тексто-рукописного режима ввода. Итоговый результат.

В дальнейшем планируется добавление типичных для текстового редактора функций, а именно таблиц, вставки фото прямо с устройства и т.д. Дополнительной планируемой функцией так же является совместное редактирование документов, что сделает ведение конспектов еще более гибким, в связи с возможностью разделить задачи.

ИССЛЕДОВАНИЕ ВОЗМОЖНОСТЕЙ И ОГРАНИЧЕНИЙ КОНЕЧНО-ОБЪЕМНОЙ БИБЛИОТЕКИ OPENFOAM ПРИ РАЗРАБОТКЕ РЕШАТЕЛЯ КВАЗИОДНОМЕРНЫХ УРАВНЕНИЙ СЕН-ВЕНАНА

Зяблицкий А.С. – студент, Кантор С.А. – к.ф.-м.н., доцент,
Кошелев К.Б. – к.ф.-м.н., с.н.с.*

Алтайский государственный технический университет (г. Барнаул)
*Институт водных и экологических проблем СО РАН (г. Барнаул)

Уравнения Сен-Венана, которые также называют уравнениями мелкой воды, служат моделью для открытых течений, в которых горизонтальные расстояния много больше глубины. Такое ограничение возникает вследствие пренебрежения вертикальными составляющими скоростей с целью упрощения самих уравнений. Данные уравнения, в частности, применяются при прогнозировании наводнений во время половодий рек в краткосрочной перспективе (3-5 дней).

Библиотека OpenFOAM призвана максимально упростить процесс проведения численного эксперимента путем предоставления набора программ решателей для востребованных в инженерной практике типов дифференциальных уравнений, а также инструментов разработки таких программ. В связи с растущей популярностью OpenFOAM, расширение библиотеки решателей является актуальной задачей.

Целями данной работы являлось:

- реализовать программу для численного решения одномерного варианта уравнений Сен-Венана для течений в каналах прямоугольного сечения с использованием пакета OpenFOAM;
- провести численный эксперимент по моделированию прорыва дамбы;
- исследовать пригодность данного пакета для решения данной и аналогичных задач.

1. Математическая постановка задачи

Квазиодномерные уравнения Сен-Венана подходят для описания течений в криволинейных руслах с произвольной зависимостью площади сечения от глубины, то есть в реальных реках. Пространственная переменная X в этих уравнениях направлена вдоль, возможно криволинейного, русла реки.

Однако, как и любая модель, одномерные уравнения Сен-Венана имеют свои ограничения. Главные ограничения — длина русла должна быть много больше его ширины, то есть можно пренебречь величинами поперечной и вертикальной составляющими скорости. Другое ограничение — средний наклон поверхности воды должен быть небольшой. Например, у р. Обь этот наклон составляет 6-10 см на каждый километр вдоль русла, что позволяет использовать уравнения Сен-Венана. Для горных рек эта модель не применима. Стоит, также, заметить, что чем больше криволинейность русла и чем больше островов, осерёдков, тем больше ошибка этой модели. Однако, её цель не улавливать локальные детали, а определить практически важный параметр — уровень поверхности воды, который определяет в реальной жизни и на реальном рельефе поверхности земли зоны затопления при паводках и половодьях.

Квазиодномерные уравнения Сен-Венана представляют собой систему из двух уравнений [1]: уравнения неразрывности (1) и закона сохранения импульса (2). В уравнении (2) трение учтено по формуле Маннинга.

$$\frac{\partial W}{\partial t} + \frac{\partial Q}{\partial x} = q, \tag{1}$$

$$\frac{\partial Q}{\partial t} + \frac{\partial}{\partial x}\left(\frac{Q^2}{W}\right) + gW\frac{\partial z}{\partial x} + \frac{gn^2 Q|Q|}{WR^{\frac{4}{3}}} = 0, \tag{2}$$

где W - площадь живого сечения,

Q − расход воды,

z = δ + h −уровень поверхности воды,

n − коэффициент шероховатости по формуле Маннинга,

R − гидравлический радиус,

δ − отметка дна,

h − глубина.

Пространственная переменная x изменяется в интервале от 0 до L.

Временная переменная t изменяется от 0 до T.

Классическая постановка начальных и граничных условий для уравнений (1) и (2) имеет вид:

$$W(x,0) = W_{\text{нач}}(x), \tag{3}$$

$$Q(x,0) = Q_{\text{нач}}(x). \tag{4}$$

$$Q(0,t) = Q_0(t), \tag{5}$$

$$W(L,t) = W_L(Q(L,t)). \tag{6}$$

2. Описание библиотеки OpenFoam

OpenFOAM (Open Field Operation and Manipulation) — библиотека для численного моделирования задач механики сплошных сред. OpenFOAM разрабатывается компанией CFD Direct и сообществом OpenFOAM Foundation по лицензии GPLv3, то есть является открытым программным обеспечением. OpenFOAM имеет очень широкую область применения, включающую в себя такие направления, как гидродинамика, динамика твердого тела, электродинамика, моделирование химических реакций. Данная библиотека разработана на языке C++. В основе работы библиотеки лежит метод конечных объемов (МКО) [3].

Одним из преимуществ OpenFOAM является использование объектно ориентированного подхода разработки как самой библиотеки, так и программ решателей, которые её используют. Данный подход позволяет добиться очень компактного и понятного кода, инкапсулировать второстепенные детали, что делает код более надежным и облегчает процесс его модификации. Результатом использования такого подхода в OpenFOAM является глубокая насыщенность смыслом каждой строчки кода и сведение к минимуму количества служебных конструкций.

Можно выделить три основных этапа для решения некоторой вычислительной задачи средствами OpenFOAM:

1. Предварительная обработка. На этом этапе создается структура файлов и директорий описывающих вычислительную задачу (кейс). В этих файлах задаётся расчетная область (“mesh” в терминах OpenFOAM), устанавливаются начальные и граничные условия, определяются физические константы и параметры решателя. В параметры решателя, в числе прочего, входят описание способов аппроксимации дифференциальных операторов, описание методов интерполяции и решения систем алгебраических уравнений, выбор времени и шага интегрирования.
2. Запуск решателя. Запуск решателя производится из корневой директории кейса. В комплекте с OpenFOAM поставляется множество решателей различных уравнений, среди которых, к примеру, есть: программа решатель уравнения Лапласа, программа

решатель уравнения переноса, программа решатель для ламинарного течения несжимаемой ньютоновской жидкости. Стоит заметить, что то, какие конкретно уравнения будут решаться в данном кейсе, определяется тем, какой решатель запустит пользователь. Таким образом, на одном кейсе возможно решать разные уравнения (запускать различные решатели).

3. Экспорт результатов. Результаты работы программы решателя, среди прочего, можно визуализировать в программе paraView, использовать в качестве начальных условий для другого кейса или конвертировать в другой формат для обработки сторонним ПО.

3. Разработка решателя и тестового примера “прорыв плотины”

В рамках данной работы была реализована численная схема из [4]. Данная схема является полунеявной и построена на основе метода расщепления по физическим процессам [2]. Уравнения решаются относительно W и Q.

Благодаря применению концепций объектно-ориентированного программирования, код, использующий OpenFOAM для решения уравнения, становится визуально похож на само уравнения. Для иллюстрации этого факта рассмотрим конкретные примеры кода из данной работы:

- Уравнение

$$\zeta_Q^0 = \left(-\frac{\partial}{\partial x}\left(\frac{Q^2}{W}\right) - gW\frac{\partial(\delta + h)}{\partial x} - \frac{gn^2 Q|Q|}{WR^{4/3}} \right)^l \tag{7}$$

решается явно с помощью кода:

```
DQ = -fvc::div(V, Q) - g*W*fvc::grad(delta+h) -
- (g*n*n*Q*mag(Q)) / (W*Foam::pow(R, 4.0/3));
```

- Уравнение

$$\frac{\zeta_Q^1 - \zeta_Q^0}{\tau} = -2\frac{\partial}{\partial x}\left(\left(\frac{Q}{W}\right)^l \zeta_Q^1\right) - 2\left(\frac{gn^2|Q|}{WR^{4/3}}\right)^l \zeta_Q^1 \tag{8}$$

решается неявно с помощью кода:

```
solve(
        fvm::ddt(DQ)
        ==
        -2*fvm::div(V, DQ) +
        fvm::Sp(-2*((g*n*n*mag(Q))/(W*Foam::pow(R, 4.0/3))), DQ)
);
```

Остальные уравнения решаются аналогичным образом.

Перейдем к созданию тестового примера. Канал будет иметь форму прямоугольного параллелепипеда длины 1000 метров. OpenFOAM поддерживает только трехмерные сетки, поэтому одномерность будем эмулировать задавая ширину и высоту канала в одну ячейку. В длину канал будет разбит на 500 ячеек.

Пусть дамба располагается по середине канала, тогда установим уровень воды левее дамбы 0.4 метра, а правее — 0.1 метра. Установим все остальные поля в нулевое значение (поле площадей живых сечений будет рассчитано решателем в начале цикла по времени и примет корректное значение).

Выберем длину канала так, чтобы волна не успевала достигнуть его границ. Тогда

можно поставить граничные условия следующим образом: уровень воды и площадь живого сечения на концах расчетной области сохраняют первоначальное значение 0.4 и 0.1 метра соответственно. Расход воды на границах нулевой. Производные по времени от расхода и площади живого сечения тоже нулевые.

Установим теперь способы аппроксимации дифференциальных операторов. Дивергенция в конвективных слагаемых аппроксимируется против потока. OpenFOAM не позволяет установить аппроксимацию “по потоку” для градиента глубины и аппроксимацию против потока для дивергенции расхода жидкости в уравнении неразрывности. Такие ограничения связаны с тем, что OpenFOAM не может определить направление самого потока, а для аппроксимации по потоку схема просто не реализована на данный момент. По умолчанию они будут аппроксимированы центральными разностями.

Определим время интегрирования от 0 до 100 с шагом в 0.1.

Результаты моделирования представлены на рисунке 1

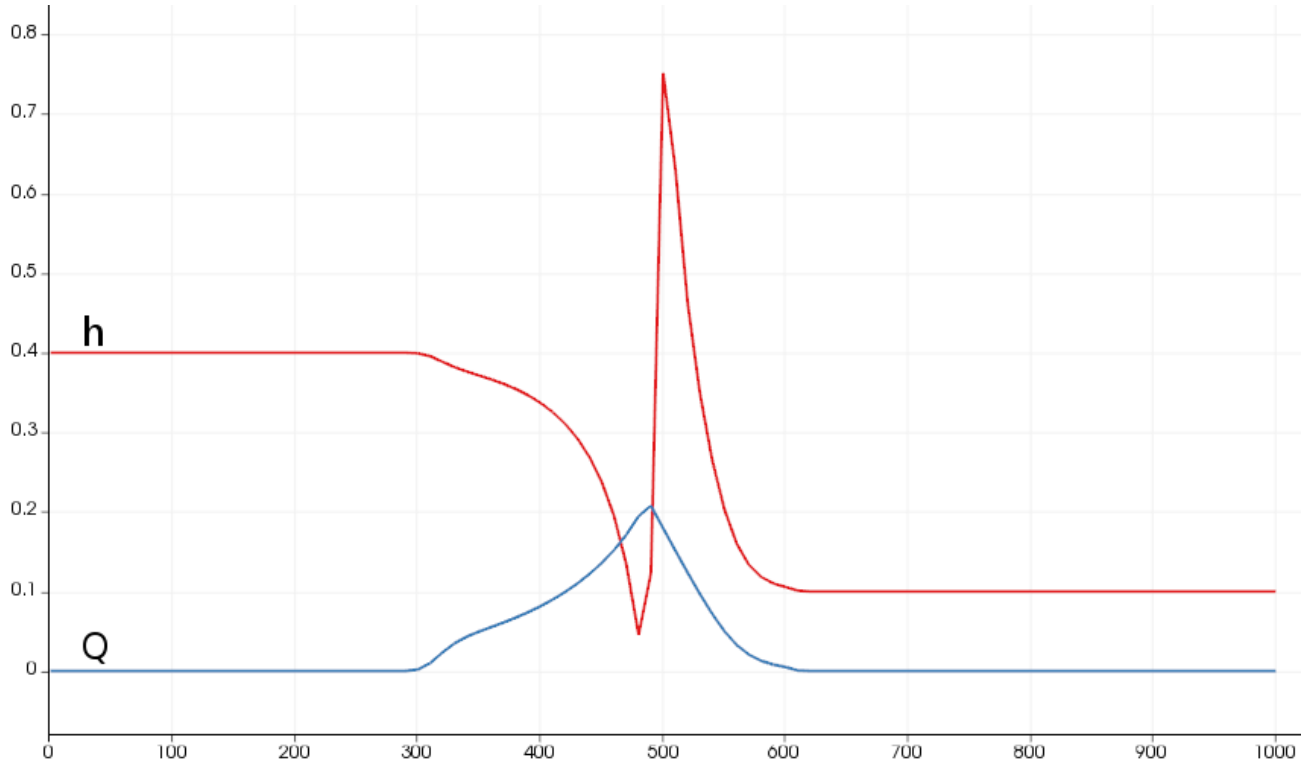


Рисунок 1 – результаты симуляции в момент времени 90

Как видно из рисунка 1 решение качественно отличается от ожидаемого.

Предположительно, возникшая немонотонность может быть вызвана следующими факторами:

- отсутствие в библиотеке схемы аппроксимации по потоку, которая нужно для аппроксимации градиента уровня воды (давления), что влияет на устойчивость решения [3];
- отсутствие аппроксимации против потока дивергенции расхода в уравнении неразрывности.

Также на качество решение могло повлиять отсутствие в OpenFOAM возможности использовать разнесённую (шахматную) сетку, что не позволяет естественным образом аппроксимировать уравнение неразрывности и градиент давления (глубины в данном примере) со вторым порядком точности. В разнесённой сетке площадь живого сечения рассчитывается в центре разностной ячейки, а расход – на ее границах. Таким образом, можно избежать лишних интерполяций расхода.

Большинство факторов, вызывающих немонотонность решения из предыдущего подраздела вызваны особенностями работы библиотеки OpenFOAM. Таким образом, для их устранения нужно вносить изменения в саму библиотеку. Так как это выходит за рамки данной работы, ограничимся решением для данного частного случая. Воспользуемся тем, что в данном примере работаем со структурированной равномерной сеткой и напишем недостающие аппроксимации по методу конечных разностей в обход библиотеки. Результаты работы модифицированного решателя приведены на рис. 2.

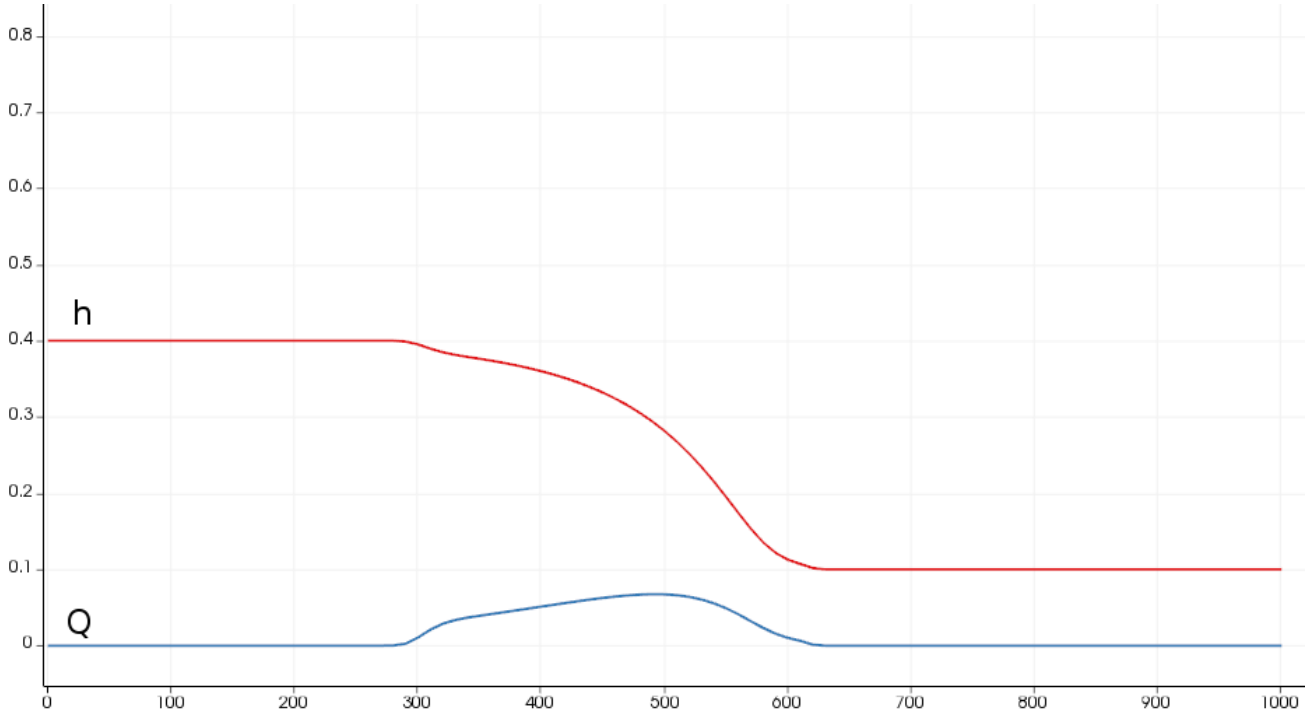


Рисунок 2 — Результат работы исправленного решателя в момент времени 90

Заключение

В результате выполнения данной работы были решены следующие задачи:

- изучены принципы работы с библиотекой OpenFOAM и выявлены её слабые стороны;
- сформированы предложения по улучшению библиотеки до уровня, достаточного для решения уравнений типа Сен-Венана, в частности, требуется возможность аппроксимировать производные в направлении потока и возможность использовать шахматную сетку;
- разработан решатель квазиодномерных уравнений Сен-Венана;
- выполнены тестовые расчеты для задачи моделирования разрушения плотины в канале прямоугольного сечения;
- найдено решение проблемы связанной с немонотонностью получаемого решения, применимое в классе частных случаев.

Список литературы

1. Гришанин К.В. Теория руслового процесса [Текст] / К.В. Гришанин. – М.: Транспорт, 1972. – 214 с.
2. Кантор С.А. Основы вычислительной математики [Текст] : учебное пособие / С.А. Кантор ; Алтайский государственный технический университет им. И. И. Ползунова. – Барнаул, 2010. – 357 с.

3. Ковеня В.М. Методы конечных разностей и конечных объемов для решения задач математической физики [Текст] / В.М. Ковеня, Д.В. Чирков. – М.: НГУ, 2013. – 87 с.
4. Кошелев К.Б. Экстремальный дождевой паводок 2014 года в бассейне верхней Оби: причины прогнозы и натурные наблюдения [Текст] / К.Б. Кошелев // Водное хозяйство России. – Москва : Изд-во Наука, 2015. – No 6. – Ч. 1. – С. 93-104.