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

НАУКА И МОЛОДЕЖЬ – 2018

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

СЕКЦИЯ

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

подсекция

ПРОГРАММНАЯ ИНЖЕНЕРИЯ

Барнаул – 2018

УДК 004

XV Всероссийская научно-техническая конференция студентов, аспирантов и молодых ученых "Наука и молодежь – 2018". Секция «Информационные технологии». Подсекция «Программная инженерия». / Алт. гос. техн. ун-т им. И.И.Ползунова. – Барнаул: изд-во АлтГТУ, 2018. – 201 с.

В сборнике представлены работы научно-технической конференции студентов, аспирантов и молодых ученых, проходившей 14 мая 2018 г.

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

Компьютерная верстка: Сорокин А.В.

СОДЕРЖАНИЕ

РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ ДЛЯ МОНИТОРИНГА И УПРАВЛЕНИЯ АВТОМАТИЧЕСКИМ ОБОРУДОВАНИЕМ

Гавриченков Е.А. – магистрант, Крючкова Е.Н. – к.ф.-м.н., профессор
Алтайский государственный технический университет

С середины прошлого века большую роль в жизни человечества стало играть разнообразное автоматическое оборудование, причем число таких оборудований и мест, в которых они используются, растет год от года. Из-за этого возрастает роль программного обеспечения, передающего данные с устройства и предоставляющего возможность управления. [1]

В данной работе рассматривается разработка информационной системы, позволяющей:

- проводить мониторинг оборудования в режиме реального времени;
- собирать и хранить данных о событиях и аварийных ситуациях;
- информировать пользователей о выявляемых событиях;
- удаленно управлять оборудованием, при наличии у оборудования подобной возможности.

Анализ различных систем привел к идее разделения всех функций системы на два типа рабочих мест: серверный и клиентский. К функциям серверного относятся: лицензирование, настройка параметров объектов и оборудования, настройка оповещений о событиях, работа с доступом к данным с клиентских приложений. Функциями клиентского являются: наблюдение состояния устройств, отправление управляющих команд, запрос событий, которые произошли на каком-то объекте или устройстве.

В системе было выделено три типа пользователей:

- Оператор: просмотр данных оборудования.
- Старший оператор: просмотр данных, управление устройствами.
- Администратор: настройка устройств, объектов, пользователей.

Рассмотрим архитектуру системы, представленную на Рисунке 1 и назначение ее программных компонентов:

- Сервер — WEB-приложение, осуществляет функции сбора первичной информации, ее обработку, хранение и передачу другим элементам системы.
- Клиент — настольное приложение, реализует функции мониторинга и управления.
- Утилита лицензирования – WEB-приложение, формирует ключ лицензии, указывая в нем допустимые разрешения для контрагента по договору с ним.

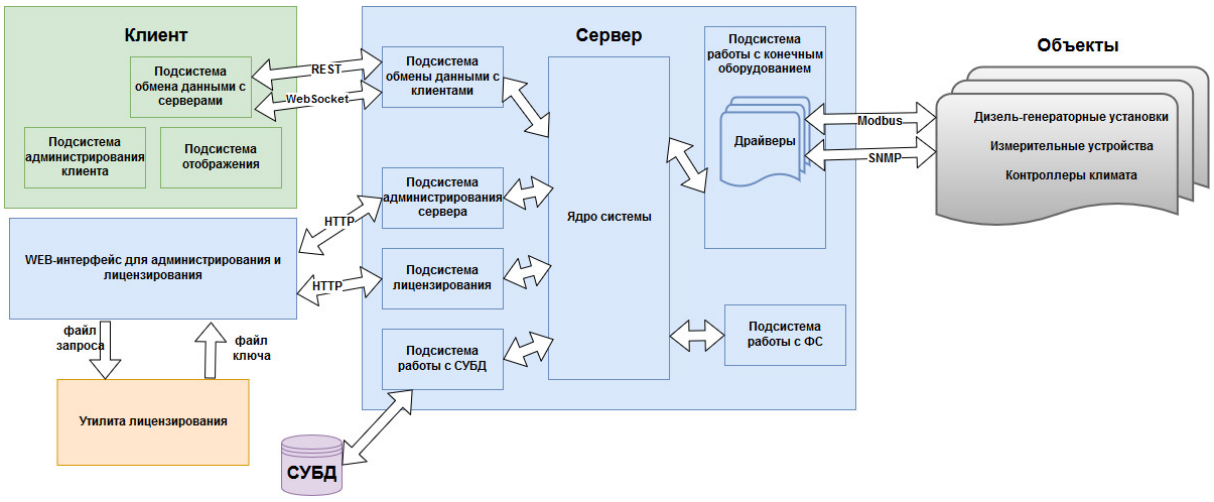


Рисунок 1 – Архитектура системы

Сервер включает в себя: подсистему работы с СУБД, подсистему работы с конечным оборудованием, подсистему администрирования, подсистему лицензирования, подсистему обмена данных с клиентом, подсистему работы с файловой системой, ядро системы.

Подсистема работы с конечным оборудованием. Реализует взаимодействие с более чем тридцатью видами устройств, использующими протоколы Modbus [2], SNMP [3], ISO1177[4], RTSP[5]. Каждый вид имеет свои регистры, команды, события, поэтому для унификации реализуются программные драйвера для каждого вида. Основные функции драйвера: получение данных с устройства и отправка команд управления.

Подсистема лицензирования – web-интерфейс для создания файла запроса лицензии, импорта полученного из утилиты лицензии лицензионного ключа, а также контроля действий пользователя в соответствии с лицензией.

Подсистема обмена данными с клиентским приложением - web-сервис для обработки запросов от клиента: управление оборудованием, запрос записей журнала событий и др.; обработки событий от сервера: появление нового события и новых данных с оборудования и т.п. При подключении клиента к серверу происходит создание Websocket [6] сессии. Она передает данные устройства от сервера к клиенту и команды управления от клиента к серверу. Пересылка асинхронных событий происходит через REST[7] сервис.

Клиент выполняет функции отображения значений параметров оборудования, фиксации факта получения информации о событии, работы с журналом событий, отправки на сервер команд для передачи на оборудования.

При подключении клиент запрашивает у сервера параметры устройств и дескрипторы панелей устройств, после чего сервер начинает отсылать данные от конечных устройств к клиенту. При этом пользователь управляет устройством, отсылая команды на сервер. Рассмотрим подробнее эти процессы с учетом усложнения обновлений, вызванного нахождением клиентов и серверов в местах без интернета.

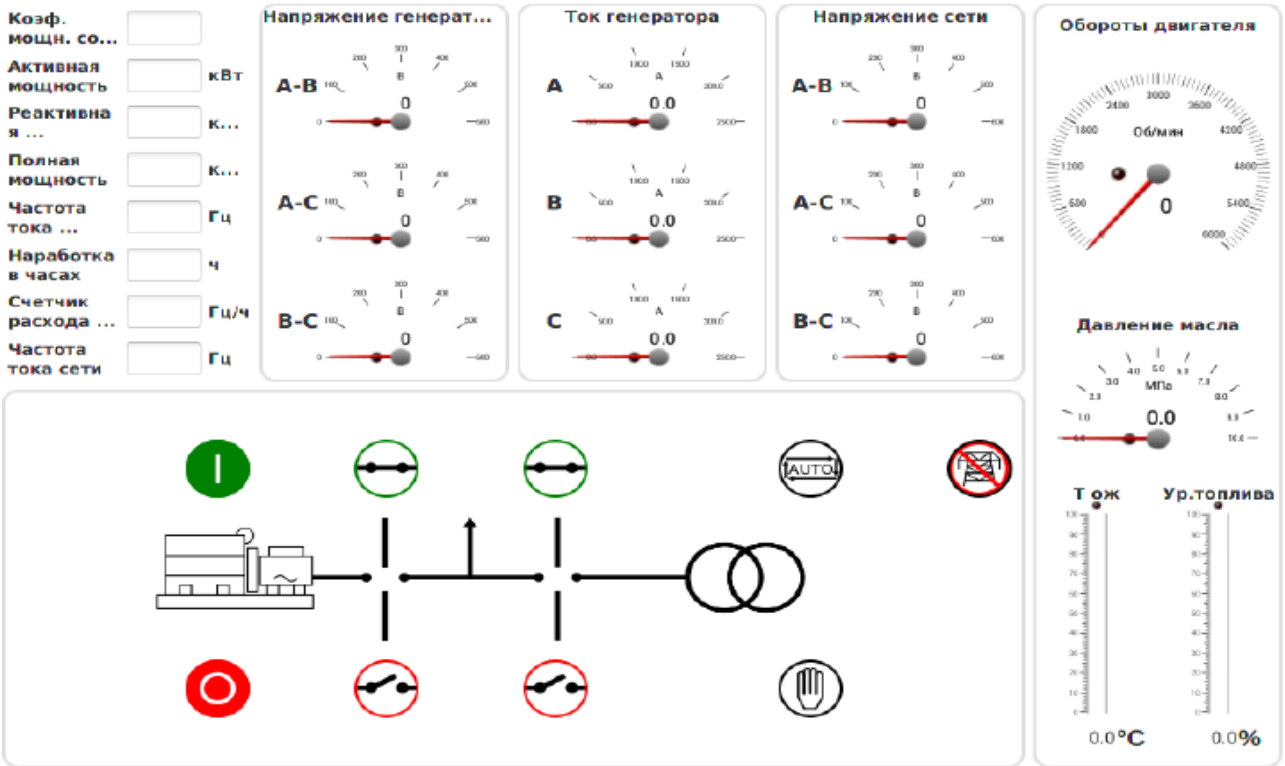


Рисунок 2 – Панель устройства DEIF GC-1F B3

При авторизации клиента на сервере формируется дескриптор, описывающий: параметры, отображаемые на панели устройства; команды, отправляемые с этой панели; компоненты, создаваемые на панели. Клиент анализирует описания компонентов и создает из них реальные объекты. Если на сервере добавляют описания компонентов, а клиент не успел обновиться, то создаются стандартные компоненты, которые не настолько визуально удобны, но позволяют продолжить работу пользователей. Пример панели устройства представлен на Рисунке 2. Число клиентов больше числа серверов, поэтому подобное решение позволит растянуть процесс обновления без осложнения работы пользователей.

Основным процессом системы является передача данных от конечного устройства к клиенту. Сервер периодически инициирует соединение с устройствами. При нескольких неудачных соединениях изменяется статус доступности устройства. При удачном соединении драйвер получает от устройства данные регистров и событий, преобразует их в общий формат и отправляет в подсистему обмена данных с клиентом. Она формирует из них сообщение веб-сокета и отправляет клиенту, причем данные отсылаются только клиентам, которые подписаны на это устройство, а события отсылаются всем пользователям. Клиент при получении параметров находит компонент с соответствующим кодом параметра и записывает в него значение параметра, а при получении аварии показывает модальное окно.

Процесс управления устройством с клиента представлен на рисунке (Рисунок 3).

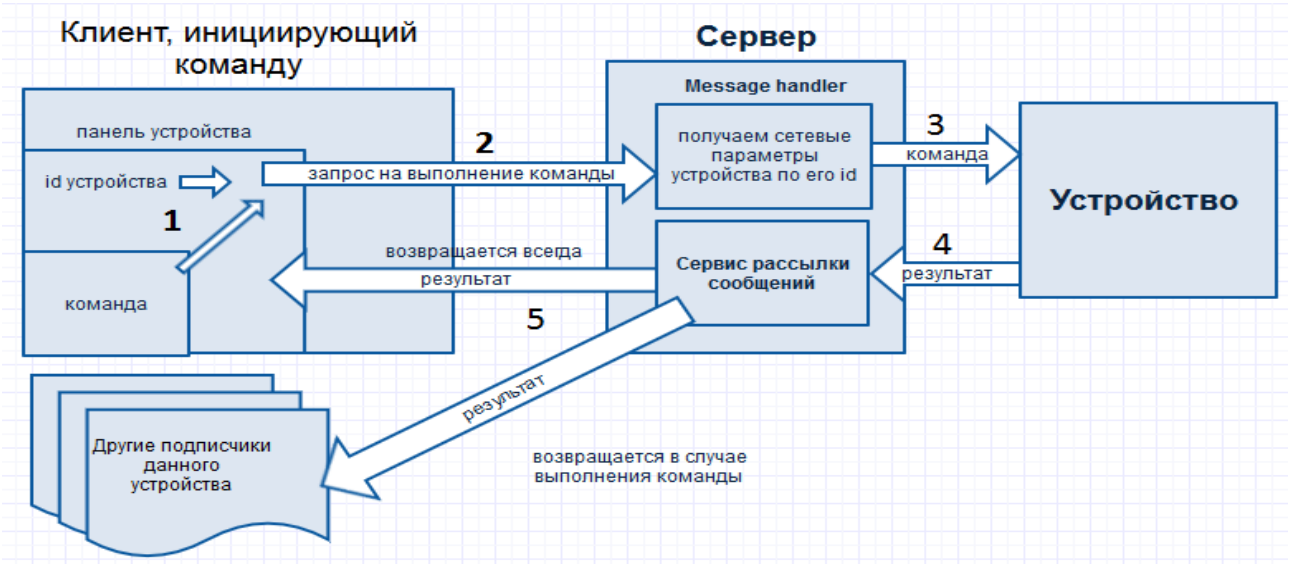


Рисунок 3 – Процесс управления устройства с клиента

На данный момент проект находится в стадии опытного тестирования.

Список литературы

1. Система учета энергоресурсов и управления объектами ЖКХ, АИИС КУЭ, АСУНО [Электронный ресурс]. – Режим доступа: http://www.gkh-pt.ru/
2. Modbus Application Protocol Specification [Электронный ресурс]. – Режим доступа: http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf
3. RFC 1098 A Simple Network Management Protocol / J. Case, M. Fedor, M. Schoffstall, J. Davin [Электронный ресурс]. – Режим доступа: https://tools.ietf.org/html/rfc1157
4. Character structure for start/stop and synchronous character oriented transmission Davin [Электронный ресурс]. – Режим доступа: https://www.iso.org/standard/5762.html
5. RFC 2326 Real Time Streaming Protocol (RTSP) / H. Schulzrinne, A. Rao, R. Lanphier. – Режим доступа: https://tools.ietf.org/html/rfc2326

6. RFC 6455 The WebSocket Protocol / I. Fette, A. Melnikov. – Режим доступа: https://tools.ietf.org/html/rfc6455
7. Fielding, R. Representational Style Transfer (REST) [Электронный ресурс]. - Режим доступа: http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm.

ПАКЕТ СЕРВИСОВ ДЛЯ ИНТЕГРАЦИИ СДО ILIAS С ИНФОРМАЦИОННОЙ ОБРАЗОВАТЕЛЬНОЙ СРЕДОЙ АЛТГТУ

Лебедев М.А. - магистрант, Андреева А.Ю. - к.ф.-м.н., доцент
Алтайский государственный технический университет (г. Барнаул)

В работе обоснована необходимость создания пакета прикладных программ для интеграции системы ILIAS в информационную образовательную среду АлтГТУ. При разработке пакета прикладных программ были реализованы следующие задачи: сбор статистики по системе ILIAS, интеграция системы ILIAS с системами Moodle, AST и электронной библиотекой АлтГТУ, автоматизация работы с оценками в ILIAS. В работе приведено описание методов решения и описание разработанного программного обеспечения.

Введение

С 2015 года коллективом разработчиков осуществлется внедрение в АлтГТУ системы управления обучением ILIAS [2,3]. Разработанный в рамках данной работы пакет прикладных программ направлен на решение ряда задач по интеграции системы ILIAS в информационную образовательную среду АлтГТУ.

К существующим методам интеграции информационных систем относятся [6-9]:

- Интеграция на уровне данных (файловый обмен, общая база данных, репликация данных);
- Интеграция на уровне пользовательских интерфейсов;
- Функциональная интеграция (использование распределенных объектов, интеграция, риентированная на сообщения, сервис-ориентированная интеграция);
- Семантическая интеграция.

В качестве основного метода интеграции выбрана сервис-ориентированная архитектура. Сервис-ориентированная архитектура обеспечивает слабое зацепление компонентов и позволяет создавать гибкие, легко масштабируемые и модернизируемые системы [6-8].

Кроме того, при проектировании и разработке программного обеспечения были использованы элементы других методов интеграции: файлового обмена, репликации данных, семантической интеграции. Разработанные библиотеки классов .NET и .NET Core могут быть использованы для интеграции с использованием распределенных объектов.

Система *мониторинга*

Система ILIAS предоставляет стандартные средства мониторинга и сбора статистики. Однако встроенные средства мониторинга ILIAS обладают рядом недостатков:

- Статистика ограничена одним клиентом ILIAS.
- Организационные структуры пользователей не соответствуют общепринятым в российском образовании (факультет — группа — студент, кафедра — преподаватель), как следствие невозможно получить статистику в разрезе групп/факультетов/кафедр.
- Отсутствует ряд параметров, необходимых для оценки качества обучения и качества

электронных курсов.

В рамках данной работы была разработана система мониторинга с учётом потребностей АлтГТУ. Разработан прототип системы мониторинга для множественных клиентов ILIAS. Разработан веб-сервис для экспорта статистики из базы данных мониторинга, реализующий арихитектуру REST. Для одноклиентной версии системы использована СУБД MySQL, для взаимодействия с базой данных было использовано PHP расширение mysqli. Для многоклиентной версии модуля использовалась СУБД MongoDB, для работы с базой данных MongoDB было использовано PHP расширение MongoDB.

Разработан плагин для ILIAS, реализующий репликацию данных статистики. Задача мониторинга, предоставляемая плагином мониторинга, вызывается менеджером задач крона ILIAS. Задача мониторинга получает данные из базы данных ILIAS, используя для выполнения запросов глобальный объект $ilDB, инкапсулирующий взаимодействие с СУБД. После этого задача крона делает запрос к базе данных мониторинга для получения данных синхронизации. Данные синхронизации используются чтобы обеспечить ссылочную целостность и исключить дублирование данных. Задача крона генерирует данные для сохранения в базу мониторинга, используя данные, полученные из ILIAS и данные синхронизации. После успешного сохранения данных в базу мониторинга задача крона завершает работу со статусом STATUS_OK.

Интеграция системы ILIAS с системами Moodle и ACT

На момент внедрения системы ILIAS, в АлтГТУ использовались другие системы управления обучением:

- СДО Moodle.
- Адаптивная Среда Тестирования — ACT-Тест.

В системах Moodle и ACT-Тест накоплено большое количество обучающих материалов, которые требуется повторно использовать в системе ILIAS. Поэтому одним из требований к разработанному ПО была возможность обмена материалами между этими системами.

Была разработана библиотека классов .NET и пользовательское приложение для генерации файлов импорта в систему дистанционного обучения ILIAS. Разработанное программное обеспечение позволяет создавать файлы импорта пулов вопросов, медиа пулов, обучающих модулей, а также конвертировать банки тестовых заданий из систем управления обучением Moodle и ACT-Тест. Было разработано кросс-платформенное приложение .NET Core и кросс-платформенный веб-сервис ASP.NET Core для конвертации файлов экспорта системы Moodle в файлы иморта системы ILIAS, а также генерации медиапулов.

В системах ACT-Тест и Moodle предусмотрены механизмы генерации файлов экспорта. Синтаксический анализ файлов экспорта из системы Moodle выполняется стандартными средствами платформы .NET, из системы ACT-Тест — библиотекой DocumentFormat.OpenXml. Семантический анализ выполняется классами-парсерами разработанной библиотеки классов. Результатом работы классов-парсеров является модель данных в классах .NET, описывающих модель предметной области. Полученная модель может быть использована классами-генераторами, содержащимися в разработанной библиотеке классов, для генерации файлов импорта в систему ILIAS, либо в форматы Moodle и ACT-Тест. Таким образом, разработанная библиотека классов позволяет реализовать миграцию данных из систем Moodle и ACT-Тест в систему ILIAS, а также миграцию данных между системами Moodle и ACT-Тест.

Схема интеграции с использованием библиотеки классов .NET Core аналогична схеме интеграции с использованием библиотеки классов .NET, с тем лишь различием, что библиотека классов .NET Core не поддерживает работу с системой ACT-Тест.

Интеграция с электронной библиотекой

Электронная библиотека образовательных ресурсов, введенная в действие в АлтГТУ в 2007 г., является одним из основных элементов интегрированной среды информационно-ресурсного обеспечения. Цель ее создания – повышение качества образовательных услуг за счет полного обеспечения всего учебного цикла учебно-методическими материалами [1].

Был разработан веб-сервис на PHP, реализующий архитектуру REST, для интеграции электронной библиотеки с системой ILIAS. Сервис предоставляет возможность импорта пользователей из электронной библиотеки в ILIAS, получение списка групп из электронной библиотеки, получение списка групп и списка курсов из ILIAS.

Для реализации интерфейса пользователя выбрана технология XMLHttpRequest и язык программирования JavaScript. Взаимодействие веб-сервиса с интерфейсом пользователя организовано в соответствии с архитектурой REST. Взаимодействие веб-сервиса с системой ILIAS организовано через интерфейсы SOAP и REST.

Пакет автоматизации работы с оценками

В ILIAS предусмотрены два основных интрумента контроля знаний [4]:

- Тест
  Студент выполняет тестовые задания, его ответы обрабатываются автоматически. Оценка за тест вычисляется на основании оценок за отдельные ответы.

- Упражение
  Студенту предлагается выполнить ряд заданий, сформулированных в свободной форме, которые будут вручную проверены преподавателем. Автоматическое вычисление итоговой оценки в системе ILIAS не реализовано.

Было разработано дополнение для автоматизированного расчёта итоговой оценки за курс на основании оценок за контрольные точки, а также был реализован автоматический расчет оценки за упраженение на основании оценок за отдельные задания.

Разработанный патч модифицирует классы ilLPMarks, ilLPCollectionSettingsTableGUI и ilLPListOfSettingsGUI сервиса LearningProgress и классы ilExAssignmentMemberStatus, ilExAssignmentEditorGUI и ilAssignmentsTableGUI модуля Exercise системы ILIAS. Для реализации функций автоматического расчёта оценок в патче был добавлен класс MarksClass.

Модульно рейтинговая система квалиметрии учебной деятельности студентов (МРСК), внедренная в АлтГТУ с 2005/2006 учебного года, является приоритетно важным элементом системы менеджмента качества образования в университете [1]. Возможность организации автоматического экспорта оценок из системы ILIAS является важным условием для интеграции ILIAS с системой «МРСК». Был разработан веб-сервис, реализующий архитектуру REST для экспорта оценок из системы ILIAS. Взаимодействие с системой ILIAS организовано через плагин Ilias.RESTPlugin.

Заключение

Разработанный пакет прикладных программ частично внедрен в АлтГТУ. Были решены поставленные задачи по интеграции системы ILIAS в информационную образовательную среду ВУЗа. Была организована интеграция системы ILIAS с системами Moodle, АСТ-Тест и электронной библиотекой, организован сбор статистики по системе ILIAS и автоматический расчёт оценок.

Список литературы

1. Щуревич В.А. Информационно-образовательное пространство вуза / В.А. Щуревич, П.И. Ананьев, Е.Г. Боровцов, А.Ю. Андреева // Высшее образование в России. – 2009. - № 4 - С. 71–76.

2. Андреева А.Ю. Выбор LMS для хранения электронных курсов в АлтГТУ/ А.Ю. Андреева, В.А. Крайванова // Развитие единой образовательной информационной среды: материалы XIV Международной научно-практической конференции. – Томск: Изд-во Томск. ун-та, 2015. – С. 117–122.
3. Внедрение LMS ILIAS в информационное образовательное пространство АлтГТУ: контакт.лицо [Текст]. В.А. Крайванова – Алт. гос. техн. ун-т им. И. И. Ползунова. – Барнаул. – 2015. – 5 с.
4. ILIAS E-Learning - User Documentation for ILIAS 5.2 [Электронный ресурс]. - Режим доступа: https://www.ilias.de/docu/ilias.php?ref_id=5651&cmd=frameset&cmdClass=ilrepositorygui&cmdNode=av&baseClass=ilrepositorygui
5. ILIAS E-Learning - Development Guide [Электронный ресурс] Режим доступа: https://www.ilias.de/docu/ilias.php?ref_id=42&obj_id=1&cmd=layout&cmdClass=illmpresentationgui&cmdNode=fr&baseClass=ilLMPresentationGUI
6. Кусов А.А. Проблемы интеграции корпоративных информационных систем // Управление экономическими системами: электроннный научный журнал. – 2011. – №. 28.
7. Морозова О.А. Интеграция корпоративных информационных систем: учебное пособие. - М.: Финансовый университет, 2014.
8. Шибанов С.В. и др. Обзор современных методов интеграции данных в информационных системах // Труды Международного симпозиума «Надежность и качество». – 2010. – Т. 1.
9. Вагин В.Н. Разработка метода интеграции информационных систем на основе метамоделирования и онтологии предметной области / В.Н. Вагин, И.С. Михайлов //Программные продукты и системы. – 2008. – № 1.

ТЕХНИЧЕСКОЕ ЗАДАНИЕ КАК ИНСТРУМЕНТ В СИСТЕМЕ УПРАВЛЕНИЯ IT-РИСКАМИ

Варзанова О.О., Волкова Е.К., Колесник Д.И. – студенты,
Астахова А.В. – к.э.н., доцент
Алтайский государственный технический университет

Запросы современного рынка IT-услуг, связанные с необходимостью разработки сложных проектов информационных, вычислительных, телекоммуникационных систем, адаптированных к условиям заказчиков, требуют от IT-компаний и отдельных разработчиков повышения качества IT-проектов и сокращения сроков их разработки. Одним из необходимых условий решения этой задачи является внедрение системы управления IT-рисками.

Проблема управления рисками имела место всегда: деятельность любой организации, в том числе, IT-компании, сопровождается появлением определенных рисков. Это опасность потери ресурсов, недополученная прибыль, срыв сроков выполнения работ, снижение качества отдельных программных продуктов, недостаточно высокая, или даже низкая, квалификация исполнителей, несоблюдение основных принципов менеджмента (руководителями различного уровня) и, как результат, потеря имиджа организации и, соответственно, потеря заказчиков на соответствующем рынке услуг. Руководители современных IT-компаний, так или иначе, ставят перед собой задачу управления рисками.

«Управление IT-рисками состоит из их периодической оценки и выполнения мероприятий по снижению выявленных рисков до приемлемого уровня» [2].

Решение проблемы управления рисками требует, прежде всего, дефиниции риска, а также предварительной классификации рисков с учетом специфики разрабатываемых IT-проектов и культуры управленческой работы, сложившейся в данной организации. Так, управление рисками для некоторых проектов предусматривает следующие основные категории: технологии разработки IT-проектов; безопасность проектных разработок; правовые аспекты; кадровое обеспечение проекта; финансовое обеспечение; инфраструктура.

С точки зрения авторов данных тезисов, наибольшую проблему в управлении рисками представляет управление такой категории, как «технология разработки проектов». В настоящее время существуют стандарты на разработку и внедрение IT-проектов различных классов, имеющие рекомендательный характер, которые на практике выполняются достаточно редко. Одной из причин отклонения технологии реализации проектов от стандартов является игнорирование необходимости разработки соответствующей проектной документации. Такую документацию желательно разрабатывать командой исполнителей на всех стадиях жизненного цикла IT-проекта.

Один из проектных документов, который в рамках IT-стандартов на разработку проектов является обязательным, – это техническое задание (ТЗ) на разработку проекта. ТЗ является также обязательным приложением к договору на выполнения исследовательских и конструкторских работ по разработке и внедрению IT-проекта [1].

От степени подробности ТЗ и четкости проработки требований к будущему IT-проекту во многом зависит количество неточностей и ошибок, которые неизбежны при реализации любого сложного комплекса работ. Другими словами, разработка качественного ТЗ позволяет уменьшить число ошибок проекта на ранних стадиях его разработки, поэтому можно утверждать, что техническое задание является эффективным инструментом в системе управления IT-рисками.

Сказанное проиллюстрируем на примере разработанного авторами тезиса ТЗ для создания IT-проекта подсистемы «Учет и реализация специального имущества» в АИС «Налоговая инспекция», отметив при этом возможности уменьшения рисков разработки некачественного программного продукта.

Существует несколько подходов к составу ТЗ. Нами приведен пример разработанного ТЗ для автоматизации управленческих работ в системе организационного управления. При этом разработка выполнена с учетом обязательного требования логической взаимосвязи всех параграфов технического задания. Вторым требованием является использование подхода к разработке ТЗ, аналогичного подходу к разработке моделей.

Нами выделено три основных модели: вербальная модель предметной области; модель процессов предметной области; информационная модель предметной области.

Формирование вербальной, или понятийной модели, предметной области аналитиком основано на процессе детального и подробного сбора и обобщения информации, представленной в виде как коммуникативных навыков, так и изучения нормативных правовых актов и нормативно-организационных документов, регламентирующих соответствующие производственные процессы.

Построение вербальной модели необходимо для полного описания предметной области и исчерпывающего понимания бизнес-аналитиком анализируемой системы; выявления проблемных ситуаций; обнаружения подсознательных либо интуитивных схем

взаимодействия сотрудников организации и связей между подразделениями и системами; а также анализа и изучения алгоритмов, по которым организация функционирует в настоящее время.

Работа налоговой инспекции включает в себя область деятельности по обнаружению, учету, оценке и реализации кладов, бесхозяйного имущества, а также конфискованного имущества в соответствии с постановлением от 29 мая 2003 года № 311 правительства Российской Федерации.

Бесхозяйными являются вещи, которые не имеют собственника или собственник которых не известен (ст. 225 ГК РФ и ст. 233 ГК РФ). Бесхозяйное имущество или клады принимаются на учет органом государственной налоговой инспекцией по заявлению о находке предметов.

При сдаче клада или бесхозяйного имущества налоговому органу (глава 33 ГПК ст. 290-293). Инспектор налоговой приглашает сотрудника отделения полиции для составления акта описи и предварительной оценки клада или бесхозяйного имущества. Акт составляется в четырех экземплярах, подписывается работниками налогового органа и отделения полиции, лицом, сдавшим клад или бесхозяйного имущества, и очевидцами. А также один экземпляр прилагается к сопроводительному письму при отсылке в последующем в Гохран.

До реализации клада или бесхозяйного имущества налоговый орган совместно с отделением полиции и с участием специалистов - экспертов местных органов культуры и торговли - проводят экспертизу предметов клада или бесхозяйного имущества. Результаты экспертизы и оценки предметов клада оформляются актом описи и оценки, который подписывается всеми лицами, принимавшими участие в экспертизе и оценке.

Оценка ценностей производится с участием специалиста - оценщика, приглашенного налоговым органом, и является предварительной. Окончательная оценка ценностей производится Главным управлением по контролю за использованием драгоценных металлов и драгоценных камней Министерства финансов, которое рассчитывается за ценности с налоговым органом. О произведенной оценке имущества составляется акт описи и оценки. Акт подписывается членами комиссии, а также специалистом - оценщиком, если он был приглашен для оценки имущества.

Предметы клада или бесхозяйного имущества в соответствие с актом приема-передачи ценностей, обращенных в собственность государства, передаются налоговыми органами в музеи. Драгоценные материалы, принятые в составе клада, пересылаются органами Министерства внутренних дел в Гохран. Физическое же лицо, обнаружившее клад, получает вознаграждение (п. 2, ст. 233 ГК РФ), основываясь на соглашении о разделе вознаграждения за обнаружение клада.

Бесхозяйное имущество, основываясь на договоре купли-продажи бесхозяйственного имущества, под контролем организаторов торгов фонда федерального имущества РФ выставляется на торги.

Конфискация имущества включает в себя принудительное безвозмездное изъятие у должника или иных лиц имущества, указанного в исполнительном документе, и передачу его государственным органам или организациям для обращения в государственную собственность в соответствии с их компетенцией, установленной Правительством РФ.

Передача конфискованного имущества государственному органу или организации производится по акту приема-передачи после снятия наложенного судебным приставом-исполнителем ареста на указанное имущество (ст. 104.1, УК от 13.06.1996 N 63-ФЗ).

Учет конфискованного имущества проводится в присутствии судебных приставов-исполнителей и понятых. Составляется отдельный акт описи и оценки. Прием и передача имущества подтверждаются подписями судебного пристава-исполнителя и представителя налогового органа на экземплярах акта описи (ареста) имущества. При переходе к государству имущества, в соответствие с актом приема-передачи ценностей, обращенных в собственность государства.

Конфискованное имущество, в соответствие с постановлением о передаче имущества должника на реализацию (ст. 87, 89; 02.10.2007 №229 - ФЗ РФ), оформляется договор купли-продажи арестованного и изъятого имущества, а также конфискованного выставляется на продажу.

Таким образом, данная функциональная подсистема будет проектироваться, во-первых, для инспекторов налогового органа, во-вторых, некоторый функционал, в виде отчетов и справки, будет доступен как физическим лицам, так и судебным приставам-исполнителям, лицам, представляющим правоохранительные органы.

Формализованное описание процессов в предметной области. Оно представляется при помощи инструментов автоматизации процессов проектирования, а именно – case-средств, направленных на облегчение понимания протекания процессов организации и удовлетворяющих предметной области. Для данного ТЗ выделены следующие процессы: деятельность по обнаружению, учету, оценке и реализации кладов, бесхозяйного имущества, конфискованного имущества. Эти процессы представляются в виде древовидной структуры, корень которой - контекстная диаграмма (см. Рисунок 1). Далее по иерархической лестнице следуют по порядку две функциональные декомпозиции в соответствии с описанием предметной области (см. Рисунок 2). После этого произведена декомпозиция каждого большого фрагмента системы на более детальные (см. Рисунок 3 и 4).

Создаем набор исходных данных, представленных в виде входных, выходных положений, а также нормативно справочной информации и действующими субъектами. На входе контекстной диаграммы имеем: клад, бесхозяйное имущество, конфискованное имущество; на выходе - акт приема-передачи ценностей, обращенных в собственность государства, соглашение о разделе вознаграждения за обнаружение клада, отчетность, договор купли-продажи бесхозяйного имущества; над управлением располагаются: административный регламент федеральной налоговой службы, нормативно правовые акты; так же используются механизмы: налоговые инспектора, судебные приставы-исполнители.

В итоге получаем четко выделенные работы, изображаемые в виде прямоугольников на диаграммах, которые создают кооперацию между собой, а также наглядно проиллюстрированные информационные, людские и производственные ресурсы, потребляемые каждой работой.

Модель бизнес-процессов обусловливает требования к информационному обеспечению системы. Это характеристика нормативно-справочной информации, основанная на концептуально полном подходе к формированию и функционированию справочных аналитических систем различного уровня; это требования к документообороту организации: к формам и содержанию входных, выходных и промежуточных документов, используемых в проекте; и, наконец, это требования к структуре базы данных. Эти требования позволяют сформировать актуальный макет базы данных в рамках методологии IDEF1X.

Структура и вариации организации данных в системе должны быть согласованны и сформированы на этапе технического проектирования. Для обеспечения надежности и сохранности данных, рассматриваемую систему целесообразно реализовать на основе СУБД.

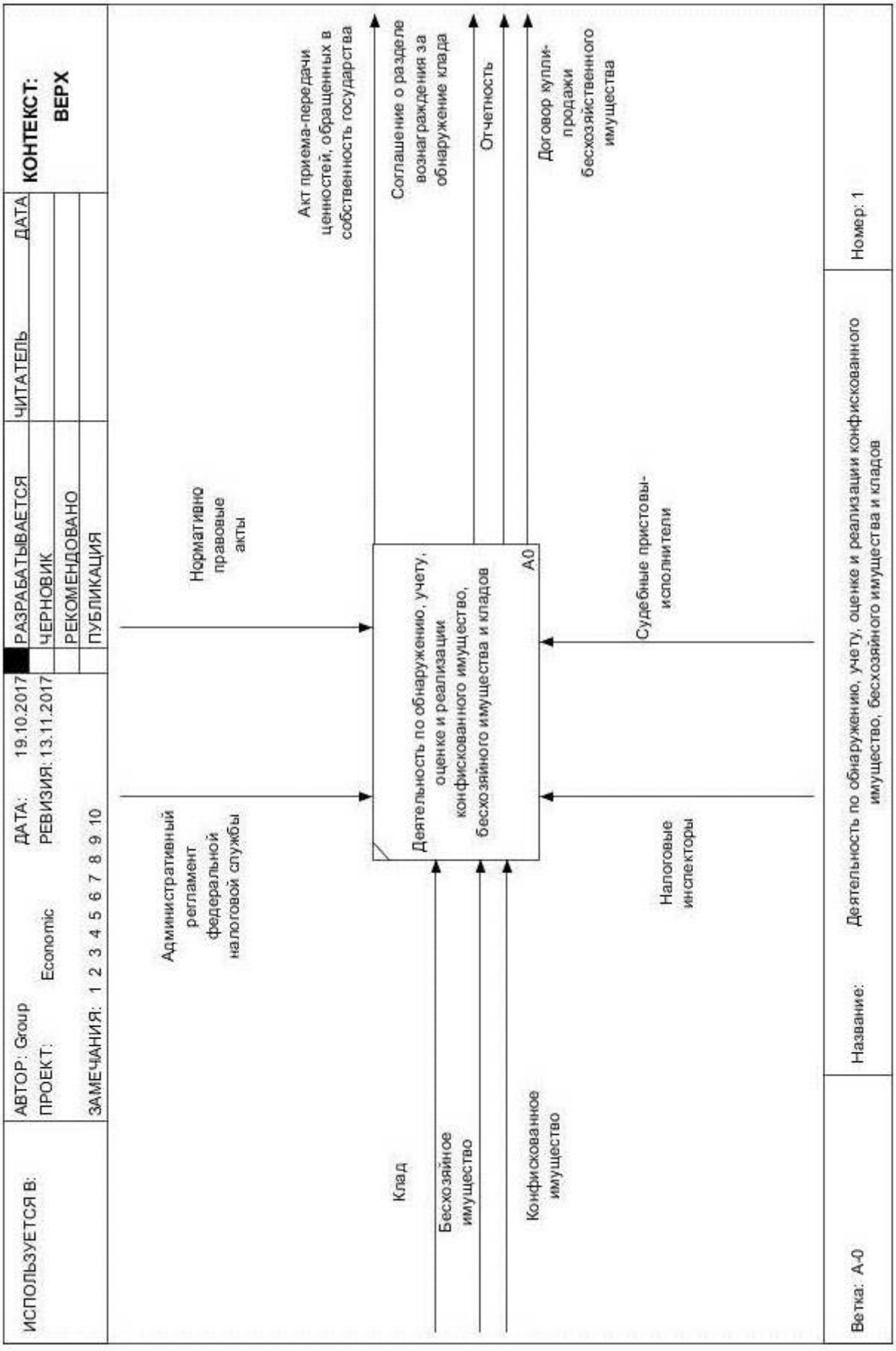


Рисунок 1 – Контекстная диаграмма функциональной модели IDEF0

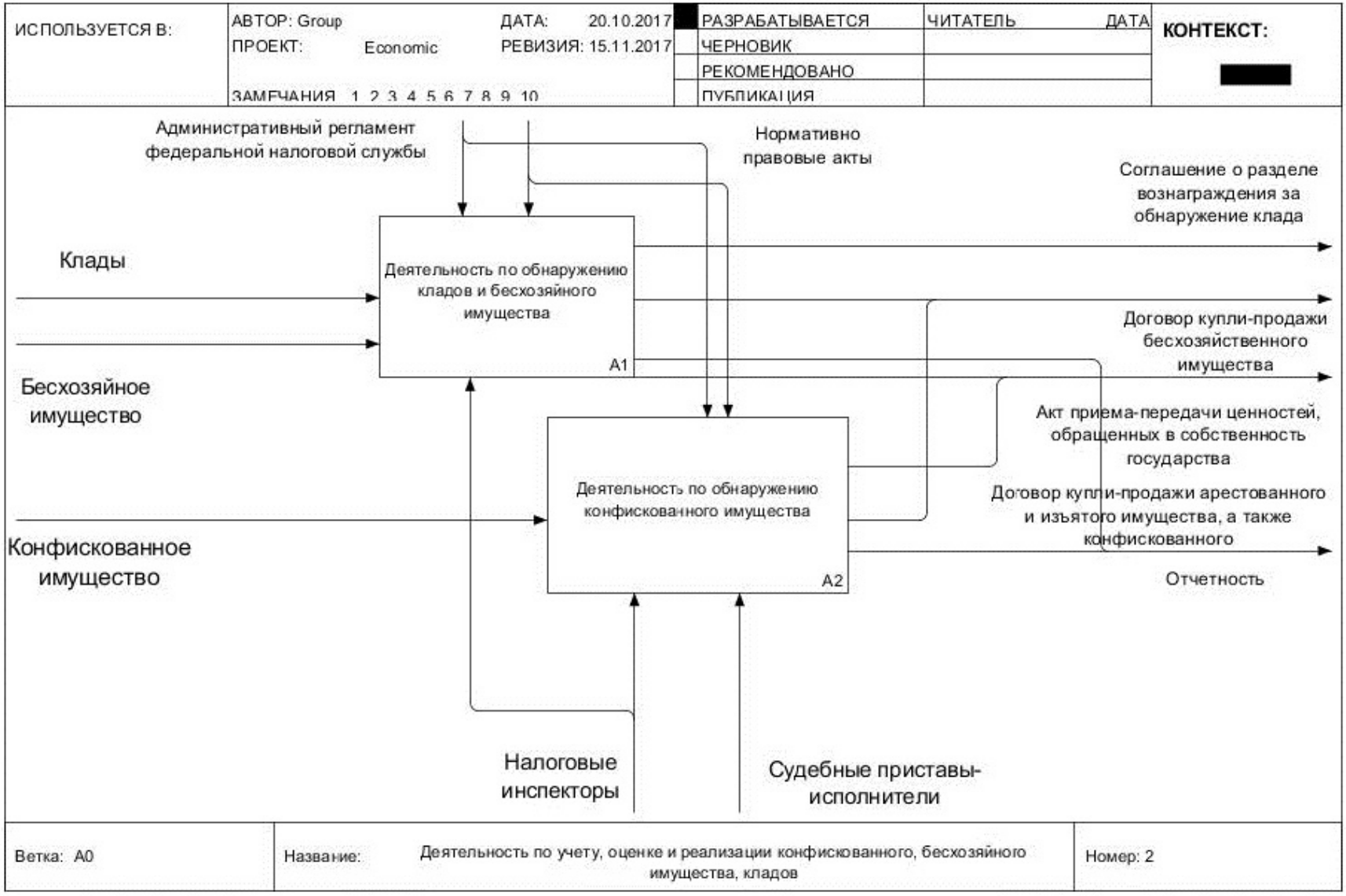


Рисунок 2 – Диаграмма декомпозиции процесса «Деятельность по обнаружению, учету, оценке и реализации конфискованного, бесхозяйного имущества и кладов»

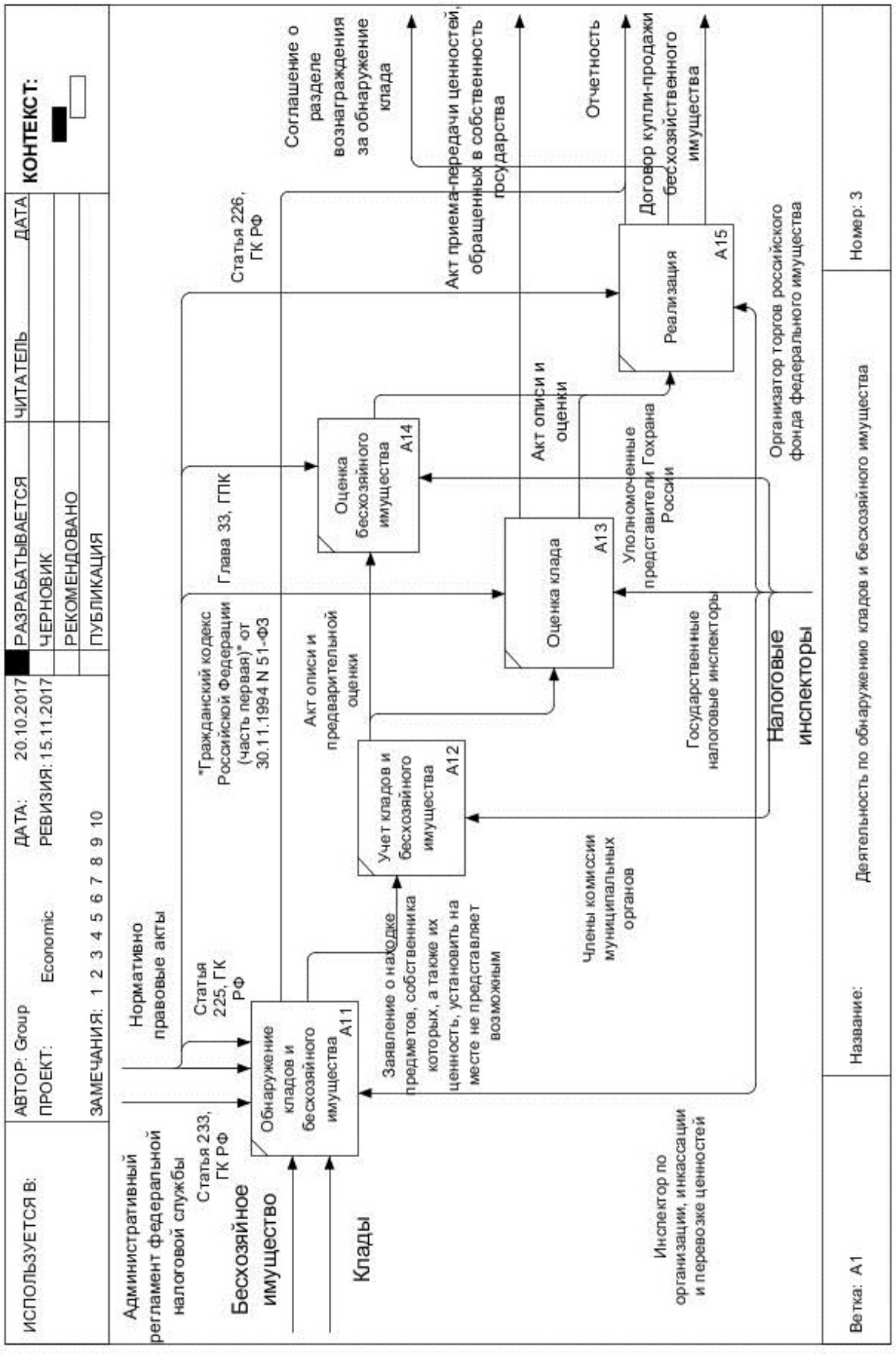


Рисунок 3 – Диаграмма декомпозиции процесса «Деятельность по обнаружению, учету, оценке и реализации бесхозяйного имущества и кладов»

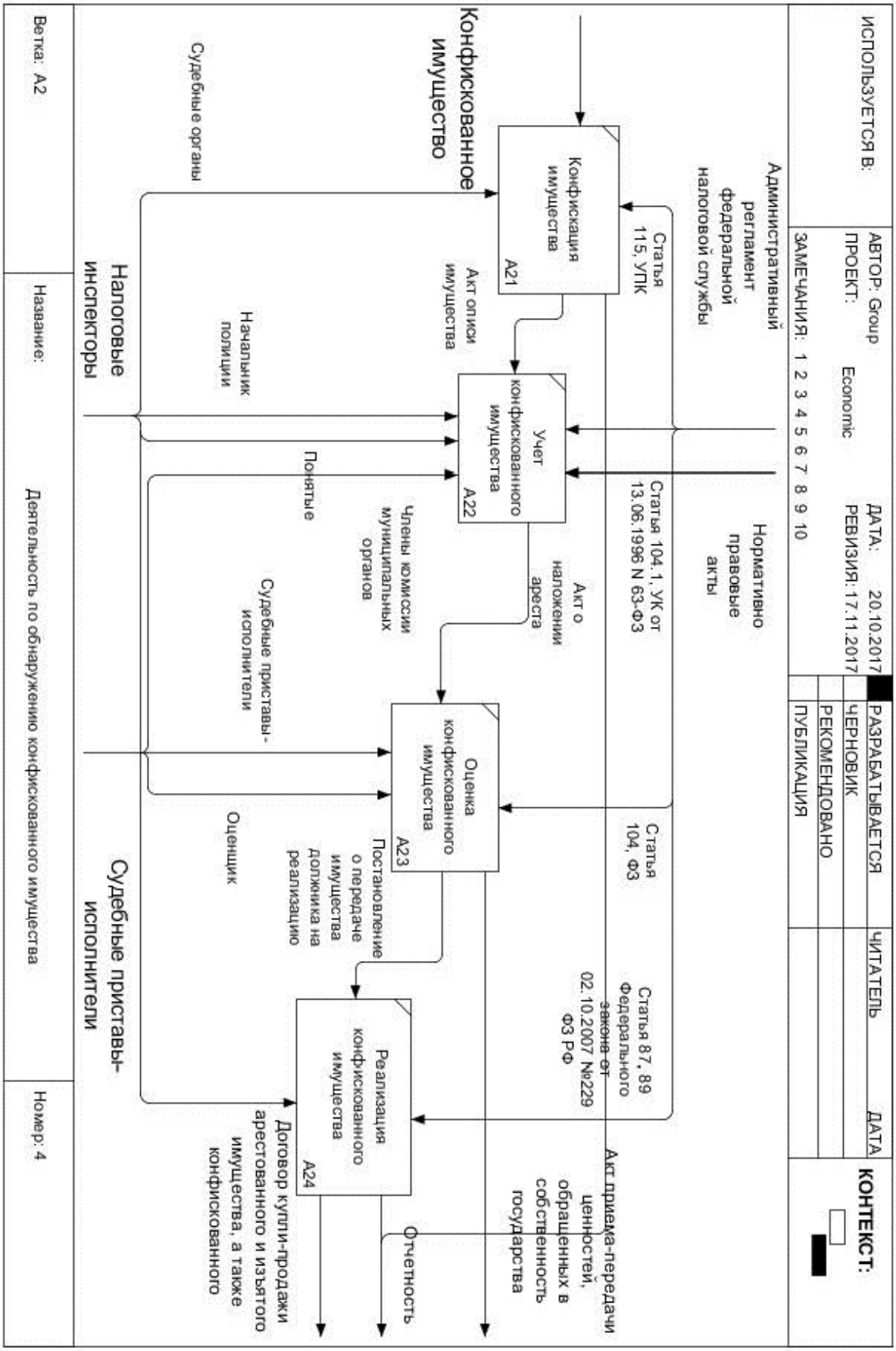


Рисунок 4 – Диаграмма декомпозиции процесса «Деятельность по обнаружению, учету, оценке и реализации конфискованного имущества»

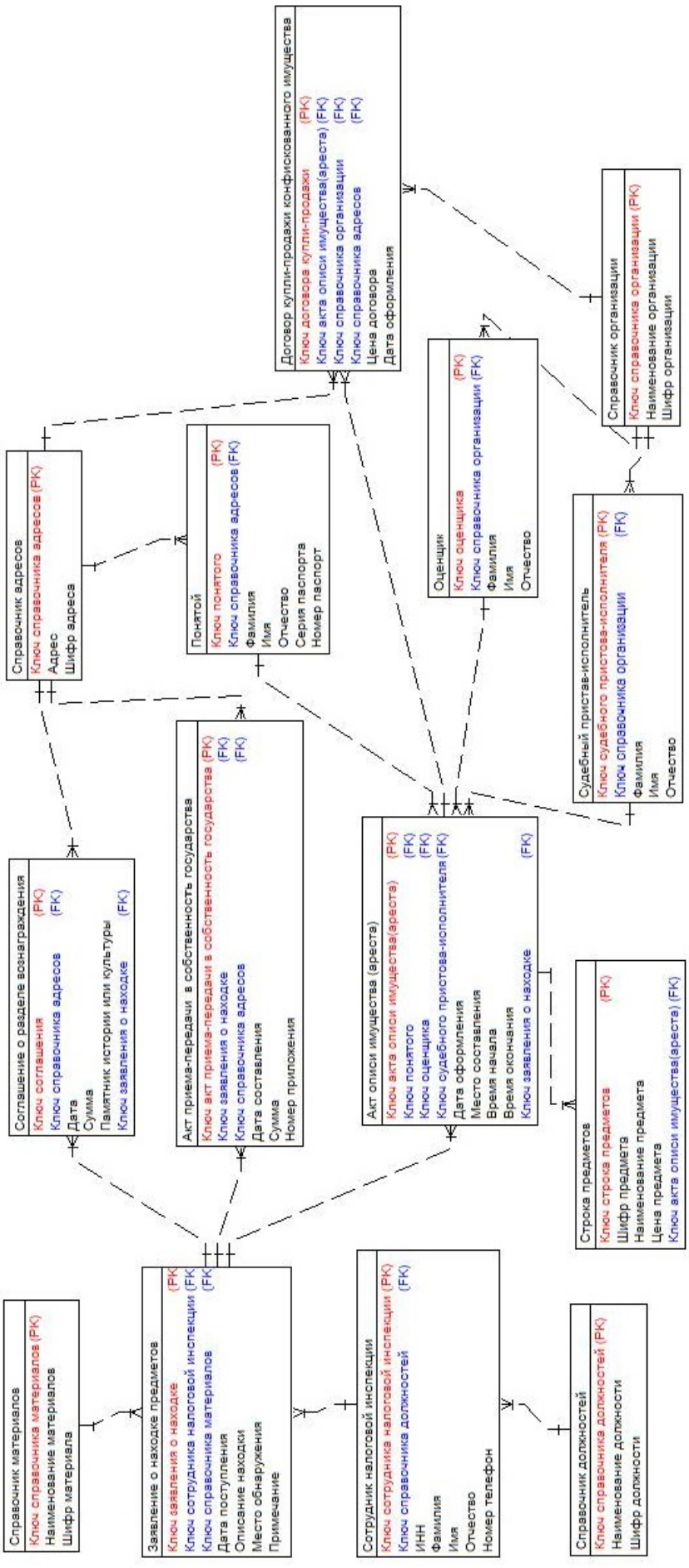


Рисунок 5 – Концептуальная модель

Реализуемая база данных должна в себя включать кодирование хранимой и обрабатываемой информации с учетом общероссийских классификаторов. Должны быть отражены и учтены все необходимые документы и информационные материалы, на основании которых разрабатывалось техническое задание. Базу данных необходимо спроектировать разумным способом, а именно, исключить возможность полной выгрузки информации, располагающуюся в базе данных системы. Акцентировать внимание будущих разработчиков на: во-первых, использовать современные технологий для обеспечения сохранности данных, резервирования и восстановления; во-вторых, должен быть реализован только авторизованный доступ к данным пользователей, которые имеют распределенные обязанности и соответствующие полномочия.

Авторами данных тезисов были разработаны основные параграфы технического задания, но не менее важными являются и ряд дополнительных параграфов, конкретизирующих ТЗ: полное наименование системы, ее условное обозначение и область использования, в которую разработанная система будет внедряться; цели автоматизации; требования к программному обеспечению (ПО), которые включают в себя требования к системному и прикладному ПО; требования к техническому обеспечению и план-график работ по проектированию, реализации и внедрению проекта.

Таким образом, благодаря предварительной разработке проекта, основанной на формировании технического задания, заказчик видит формализованное представление проекта, получает возможность сэкономить как материальные, так и трудовые затраты, и повысить в несколько раз шансы на успешность реализации проекта. Также уменьшить риски срыва сроков выполнения проекта позволяет построение графика работ по реализации и внедрению проекта. В некоторых случаях хорошо продуманное планирование работ дает возможность реализации проекта даже раньше назначенного срока.

Разработанное техническое задание может быть использовано в качестве примера проектной документации при изучении студентами направления подготовки «Программная инженерия» учебной дисциплины «Основы экономики программной инженерии и управление проектами».

Список литературы

1. Раздел VII. Права на результаты интеллектуальной деятельности и средства индивидуализации [Электронный ресурс] // Consultant.ru. – КонсультантПлюс – Режим доступа: http://www.consultant.ru/document/cons_doc_LAW_64629/
2. Методология управления ИТ-рисками [Электронный ресурс] // Osp.ru — Открытые системы. – Режим доступа: https://www.osp.ru/os/2006/08/3584582/
3. Постановление Правительства РФ от 29.05.2003 N 311 (ред. от 21.02.2017) "О порядке учета, оценки и распоряжения имуществом, обращенным в собственность государства" [Электронный ресурс] // Consultant.ru – КонсультантПлюс. – Режим доступа: http://www.consultant.ru/document/cons_doc_LAW_42464/b7c150185c78ad756c47fede00ccadf04b1c8a61/
4. "Гражданский кодекс Российской Федерации (часть первая)" от 30.11.1994 N 51-ФЗ (ред. от 05.12.2017) / ГК РФ Статья 225. Бесхозяйные вещи [Электронный ресурс] // Consultant.ru – КонсультантПлюс.– Режим доступа: http://www.consultant.ru/document/cons_doc_LAW_5142/9ff952dc782e98c012ec915600891da688cc7bf3/
5. "Гражданский кодекс Российской Федерации (часть первая)" от 30.11.1994 N 51-ФЗ (ред. от 05.12.2017) / ГК РФ Статья 233. Клад [Электронный ресурс] // Consultant.ru – КонсультантПлюс. – Режим доступа: http://www.consultant.ru/document/cons_doc_LAW_5142/9b86dee7016d8fc633a322e0541683e964042979/
6. "Гражданский процессуальный кодекс Российской Федерации" от 14.11.2002 N 138-ФЗ (ред. от 30.10.2017) / ГПК РФ Глава 33. Признание движимой вещи бесхозяйной и признание права собственности на бесхозяйную недвижимую вещь [Электронный ресурс] // Consultant.ru – КонсультантПлюс. - Режим доступа:

http://www.consultant.ru/document/cons_doc_LAW_39570/abcd43a500fd03e292f361e082d0a2a2b8a37ceb/
7. "Уголовный кодекс Российской Федерации" от 13.06.1996 N 63-ФЗ (ред. от 29.07.2017) (с изм. и доп., вступ. в силу с 26.08.2017) / УК РФ Статья 104.1. Конфискация имущества [Электронный ресурс] // Consultant.ru – КонсультантПлюс. – Режим доступа: http://www.consultant.ru/document/cons_doc_LAW_10699/f22429461fc4befb140b98a33cf3521eea282f7d/
8. Федеральный закон от 02.10.2007 N 229-ФЗ (ред. от 05.12.2017) "Об исполнительном производстве" Статья 89. Реализация имущества должника на торгах [Электронный ресурс] // Consultant.ru – КонсультантПлюс. - Режим доступа: http://www.consultant.ru/document/Cons_doc_LAW_71450/ee5f1dab23ec884a064265cbda2028d57da9d360/

СИСТЕМА УВЕДОМЛЕНИЯ КЛИЕНТОВ БАНКА ОБ ОПЕРАЦИЯХ С ИСПОЛЬЗОВАНИЕМ ЭЛЕКТРОННОГО СРЕДСТВА ПЛАТЕЖА

Горбаненко С.С., Чибриков Д.С. – студенты, Троицкий В.С. – к.ф.-м.н., доцент
Алтайский государственный технический университет (г. Барнаул)

В соответствии с Федеральным законом от 27.06.2011 N 161-ФЗ "О национальной платежной системе" оператор по переводу денежных средств обязан информировать клиента о совершении каждой операции с использованием электронного средства платежа путем направления клиенту соответствующего уведомления в порядке, установленном договором с клиентом. Это означает, что банк обязан иметь некую технологию уведомления своих клиентов.

Наиболее распространенная такая технология на сегодня - SMS-информирование. Например, вы получаете SMS-сообщение рассчитавшись пластиковой картой в магазине или получая наличные рубли в банкомате.

В настоящее время для отправки SMS сообщений клиентам банк использует сервисную платформу "Интерэктив Сервисез" (ИЭС). Реализован односторонний режим обмена сообщениями (Банк –> ИЭС). Взаимодействие осуществляется по протоколу HTTPS. Обмен происходит через отправку стандартного POST-запроса, содержащего соответствующий XML-пакет. Все отправляемые SMS сообщения являются платными, т.е. банк платит ИЭС за каждое отправленное сообщение.

В целях экономии денежных средств и для удобства клиентов, банк рассматривает возможность частичной замены SMS на передачу сообщений через сеть Internet (для клиентов, у которых есть смартфон).

При реализации этой идеи, желательно сохранить технологию взаимодействия между автоматизированной банковской системой (АБС) и сервисной платформой отправки сообщений (СП). Из этого требования вытекает архитектура будущей системы: необходима серверная часть, например, WEB-сервер (СПW), который будет принимать сообщения от АБС по протоколу похожему на уже используемый банком и передавать клиентам по их запросу. Также, нужна клиентская часть, программа для Android и iPhone, устанавливаемая на смартфон клиента (ПОКл). ПОКл будет регулярно считывать сообщения с СПW и отображать их в привлекательном виде на экране смартфона. Очевидно, что сообщения должны доставляться адресно, а значит, необходима система авторизации пользователей и разграничения доступа. Также необходимо продумать систему обеспечения безопасности при передачи этих сообщений.

Для решения поставленной задачи, нами ведется разработка специализированного Web-сервера и клиентского приложения.

Разработана общая архитектура системы, спроектированы собственные интерфейсы взаимодействия СПW-АБС и СПW-ПОКл, а также реализована поддержка API ИЭС. Интерфейс СПW-АБС повторяет используемый сегодня API ИЭС в части отправки СМС-сообщений, что позволит внедрить наше ПО без существенных доработок АБС. Кроме того, разграничение уровней взаимодействия значительно повышает общий уровень безопасности. Для реализации проекта СПW были использованы Maven, Spring, Hibernate, Apache Tomcat и PostgreSQL.

В качестве клиентской части разрабатывается приложения для Android с начальным функционалом, включающим получение сообщений, их вывод на экран и последующее хранение. Для работы приложения требуется доступ к интернету и Android версии не ниже 4.4. Безопасность данных приложения, а также доступа к нему контролируется стандартными средствами Android (шифрование данных, вход по пин-коду или отпечатку пальца). В качестве дальнейшего развития системы будет разработано приложение для IOS и добавлены новые функции для Android, такие как показ актуальных акций и предложений банка и пр. При разработке клиентского приложения были использованы: JavaFX, Gradle, SQLite, Hibernate, Android SDK.

По нашим расчетам к июню 2018 готовая система будет передана в опытную эксплуатацию заказчику.

СИСТЕМА АВТОМАТИЗАЦИИ ВЕРБАЛЬНОГО АНАЛИЗА ПРИНЯТИЯ РЕШЕНИЙ

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

В современном мире часто возникают проблемы, связанные с принятием решений, последствия которых могут быть серьезными. При принятии таких решений часто возникают задачи выбора, классификации и ранжирования альтернатив [1]. Примерами таких задач являются выбор оптимальных инвестиционных решений [2], выбор оптимальной схемы бизнес-процессов [3] и т.п. Главная задача – это сравнение имеющихся альтернатив и выбор наилучшей из них.

Предлагаемое решение

Целью данной работы является разработка программного обеспечения, реализующего методики принятия решений, используя многокритериальную теорию полезности, с дальнейшей возможностью расширения функционала.

Один из способов решения не формализованных задач, который используют лица, принимающие решение (ЛПР), - это упорядочивание альтернативных решений, а затем последующий выбор оптимального по некоторым критериям решения из полученного упорядоченного списка. В связи с большим количеством альтернатив, которое нужно подвергнуть анализу, ведутся исследования по автоматизации этого процесса. Одним из подходов, на котором строится автоматизация процесса, является многокритериальная теория полезности **MAUT** (Multi-Attribute Utility Theory). MAUT представляет собой набор основных этапов для решения [1]:

1. Разработка перечня критериев;
2. Построение функции полезности по каждому из критериев;
3. Проверка некоторых условий, определяющих вид общей функции полезности;

4. Построение зависимости между оценками альтернатив по критериям и общим качеством альтернативы;
5. Оценка всех имеющихся альтернатив с выбором наилучшей альтернативы.

MAUT строится на основе двух групп аксиом: аксиомы общего характера и аксиомы независимости альтернатив.

Первая группа - аксиомы общего характера:

- *Аксиома полноты*: может быть установлено отношение между полезностями любых альтернатив - либо одна из них превосходит другую, либо они равны.
- *Аксиома транзитивности*: из превосходства полезности альтернативы ***A*** над полезностью альтернативы ***B*** и превосходства полезности ***B*** над полезностью ***C*** следует превосходство полезности альтернативы ***A*** над полезностью альтернативы ***C***.
- *Для соотношений между полезностями* альтернатив ***A***, ***B,*** ***C***, имеющими вид $U(A) > U(B) > U(C), 0 \le \alpha \le 1, 0 \le \beta \le 1$, можно найти такие числа, что: $\alpha U(A) + (1 - \alpha)U(C) = U(B)$, $(1 - \beta)U(A) + \beta U(B) > U(B)$.**

Вторая группа – аксиомы независимости:

- *Независимость по разности*. Предпочтения между двумя альтернативами, отличающимися лишь оценками по порядковой шкале одного критерия c_1, не зависят от одинаковых (фиксированных) оценок по другим критериям $c_2, \dots, c_N$.
- *Независимость по полезности.* Критерий c_1 называется независимым по полезности от критериев $c_2, \dots, c_N$, если порядок предпочтений лотерей, в которых меняются лишь уровни критерия c_1, не зависит от фиксированных значений по другим критериям.
- *Независимость по предпочтению.* Два критерия c_1 и c_2 независимы по предпо-чтению от других критериев $c_2, \dots, c_N$, если предпочтения между альтернативами, различающимися лишь оценками по c_1, c_2, не зависят от фиксированных значений по другим критериям.

Для построения функции полезности требуется определить эквивалент определенности для лотереи. Это можно сделать, используя **теорию ожидаемой полезности фон Неймана-Моргенштерна [4].**

Аксиоматическое обоснование теории:

- *Аксиома полноты.* Для любых ***A***, ***B*** должно выпол-няться соотношение $A > B$, $B > A$ или $A = B$. То есть, при выборе между А и В игрок должен или предпочитать вариант А, или предпочитать вариант В, или ему должно быть всё равно.
- *Аксиома транзитивности*. Если $A > B$ и $B > C$, то $A > C$. То есть, если игроку А кажется лучше, чем ***B***, а ***B*** - лучше, чем ***C***, то для него ***A*** будет лучше, чем ***C***.
- *Аксиома независимости.* Предположим, что $A > B$ и $p \in (0;1]$, тогда для любого ***C*** имеет смысл неравенство $pA + (1 - p)C > pB + (1 - p)C$. То есть, если для игрока ***A*** лучше, чем ***B***, то он предпочтёт замену ***B*** на ***A*** (с той же вероятностью ***p***), независимо от третьей альтернативы ***C***.
- *Аксиома непрерывности.* Предположим, что $A > B > C$, тогда можно представить в виде $pA + (1 - p)C$, где $p \in (0;1]$. То есть, если игроку вариант ***A*** нравится больше, чем ***B***, а ***B*** - больше, чем ***C***, то существует такая вероятность ***p***, что игроку будет всё равно, получит ли он ***B*** гарантированно или положится на случай, который предоставит ему либо более полезный, чем ***B***, вариант ***A*** с негарантированной вероятностью ***p***, либо менее полезный ***C***.

Определим весовые коэффициенты критериев на основе экспертной оценке ЛПР, тогда функцию полезности будем вычислять в виде суммы однокритериальных функций полезности:

$$U(x) = \sum_{i=1}^{N} \omega_i U_i(x).$$

Разработанная система, реализует методы, описанные выше. Она способна находить наилучшую альтернативу для конкретной задачи при учёте предпочтений ЛПР. В качестве дальнейшего расширения системы, можно внести новый модуль нахождения наилучшей альтернативы, используя другой метод (паттерн стратегия), сформировать модуль для выбора той или иной стратегии и проводить сравнительный анализ, исходя из результатов работы.

Список литературы

1. Ларичев О.И. Вербальный анализ решений. – Наука, 2005. – 296 с.
2. Мартынов С.В. Исследование многокритериальных технологий в системе оптимальных инвестиционных решений / С.В.Мартынов, М.А. Шаталов // Современное общество и власть. - 2017. - № 4. - С. 46–48.
3. Серенков П.С. и др. Применение теории полезности для формирования ядра экспертной системы. // Методы менеджмента качеств. – 2013. - № 7. – С. 24–29.
4. Зинченко А.Б. Теория полезности фон Неймана-Моргенштерна и расчет оптимальных портфелей для различных моделей финансовых рынков - Ростов-на-Дону, 2007. – 67 с.

ОБЗОР ПОДХОДОВ РЕШЕНИЯ ЗАДАЧИ СИНХРОНИЗАЦИИ ДАННЫХ В РАСПРЕДЕЛЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ

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

Введение

Задачи синхронизации данных возникли вслед за распространением сетевых технологий. В решении задач синхронизации заинтересованы многие организации. С ростом объемов информации для повышения отказоустойчивости и ускорения обработки запросов применяют репликацию данных. В медицинских системах актуально синхронизировать данные, чтобы можно было работать с историей наблюдений независимо от местоположения пациента.

1. Разновидности синхронизаций

Наиболее распространенным типом синхронизации является репликация данных. В этом случае имеется мастер-копия (origin), а также зависимые реплики. Основное назначение

такой синхронизации – повысить доступность ресурса, в случае отказа мастера, а также распределить нагрузку, в случае если мастер не справляется с обслуживанием запросов.

Также различают пессимистическую и оптимистическую синхронизацию данных. При пессимистической синхронизации данных все действия с данными проходят через главную реплику (primary replica).

Ответственность по поддержанию копий лежит на главной реплике. Основная идея пессимистической репликации – блокирование доступа к репликам, пока мы не уверены в их актуальности. В случае, если главная реплика недоступна происходят перевыборы главной реплики. Пессимистическая синхронизация жизнеспособна в небольших локальных сетях и при наличии хотя бы какой-либо пусть не стабильной связи [1].

Пессимистическая синхронизация, тем не менее, не подходит для сред с необходимостью командной работы и взаимодействия людей. Оптимистическая синхронизация позволяет клиентам работать параллельно. Содержимое реплик может разойтись и отлично справляется в случае полного пропадания связи между репликами. Оптимистической она называется потому, что делается предположение о том, что конфликты будут возникать редко. Несмотря на кажущиеся преимущества у оптимистической синхронизации имеются недостатки. Там, где пессимистическая среда ожидает, оптимистическая среда обрабатывает конфликты [1].

2. Обзор существующих алгоритмов синхронизации данных

2.1. Синхронизация на основе дайджеста сообщений

Синхронизация на основе дайджеста сообщений (SAMD) основана на наличии центрального сервера, который занимается вычислениями и синхронизацией. Клиенты данного алгоритма – различные мобильные устройства, персональные ассистенты и смартфоны. Телефоны обладают ограниченными вычислительными мощностями и ограниченной емкостью батареи. Алгоритм позволяет получить лишь часть данных, которые имеются на сервере. Проблема многих алгоритмов синхронизации данных заключается в их сильной зависимости от баз данных, так как используют метаданные, триггеры и штампы времени. В то время как SAMD использует стандартные SQL запросы и позволяет использовать различные СУБД на клиенте и сервере [2].

Для каждой таблицы на клиенте и на сервере создается вспомогательная таблица с дайджестом сообщений. На клиенте в такой таблице хранится первичный ключ (совпадает с ключами таблицы данных), хэш ряда, флаг. На сервере дополнительно хранится идентификатор клиента. В случае изменения ряда хэш на клиенте пересчитывается и выставляется флаг необходимости синхронизации. Для отслеживания изменений не ведется журнал изменений. В случае удаления ряда значение хэша выставляется в null.

Для синхронизации в таком случае не требуется пересылать все данные, а лишь данные таблицы с дайджестом сообщений. Сами данные отсылаются в случае наличии изменений.

2.2 Транзакционный алгоритм проталкивания результирующих множеств

Транзакционный алгоритм проталкивания результирующих множеств (Transaction Level Result-Set Propagation TLSRP) [1] предлагает оптимистический алгоритм репликации данных. Важным моментом данного алгоритма является возможные допущения отсутствия соединения между репликами. В модели репликации данных данного типа предлагается архитектура, при которой имеется слой серверов (tier) с фиксированным быстрым соединением, а также слой с поддерживающими мобильными станциями. В слое серверов

используется традиционный процесс репликации. Мобильные станции предполагают непостоянное соединение со слоем серверов. На рисунке 1 представлена архитектура данной системы.

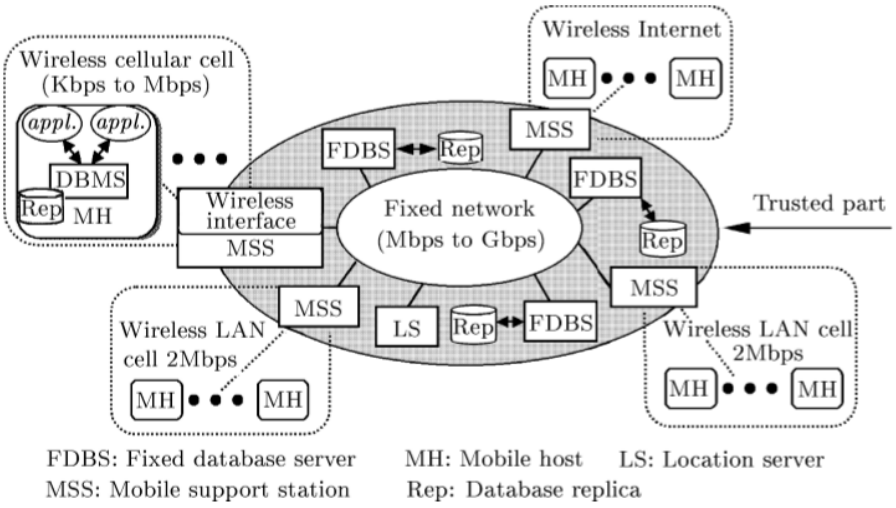


Рисунок 1 – Архитектура системы репликации данных

В данном алгоритме для отслеживания изменений используется упрощенный журнал транзакций (для экономии пространства). Алгоритм синхронизирует данные в две транзакции, сначала на сервер отправляются изменения, затем загружаются изменения с сервера. Разрешением конфликтов занимается серверная часть. Для этого составляются наборы множеств и исходя из доказанных автором теорем и ревизии изменений можно делать выводы относительно необходимости применения транзакций. Часть транзакций таким образом отклоняется, другие применяются на серверной стороне. После этого клиент загружает необходимые изменения с сервера и обновляет данные. На стороне клиента фиксируется момент последней синхронизации для загрузки только необходимых транзакций с сервера.

2.3 Синхронизация данных в медицинских информационных системах

В работе [3] рассматриваются подходы, применяемый для синхронизации между медицинскими серверами в Сербии. Характерной особенностью описанной медицинской системы было отсутствие постоянного соединения между узлами. Архитектура системы серверов представляла древовидную структуру, что означает наличие центральных узлов, а также филиалов. От министерства здравоохранения Сербии было требование не менять существующую схему таблиц. Использовать проприетарные решения для репликации данных было неосуществимо, потому что не на всех узлах была поддержка данного функционала. В данном алгоритме создавались вспомогательные таблицы и при выполнении DML-выражений (Data manipulation language statement), текст этих запросов записывался в специальную вспомогательную таблицу.

Кроме того, интересным моментом данной статьи был способ обработки конфликтов автоинкрементных полей. Для решения данной проблемы, каждому серверу назначался свой диапазон значений для автоинкремента.

К недостаткам данного подхода можно отнести невозможность синхронизации бинарных данных, поскольку в запросе сам бинарный объект нельзя прописать.

2.4 Синхронизация файлов Dropbox

Для синхронизации файлов Dropbox использует специальный журнал файлов Server File Journal (SFJ). В данном журнале хранится список (blocklist) хэшей фрагментов (chunks)

файла, а также номер ревизии Journal ID (JID), для поддержания актуального состояния файлов. Фрагмент (chunk) имеет размер 4 мегабайта. На рисунке 2 приведен пример разбиения файла на фрагменты. Если у него поменялась только часть, то хэш-значения совпадут, и на сервер будет грузиться только необходимый фрагмент [4].

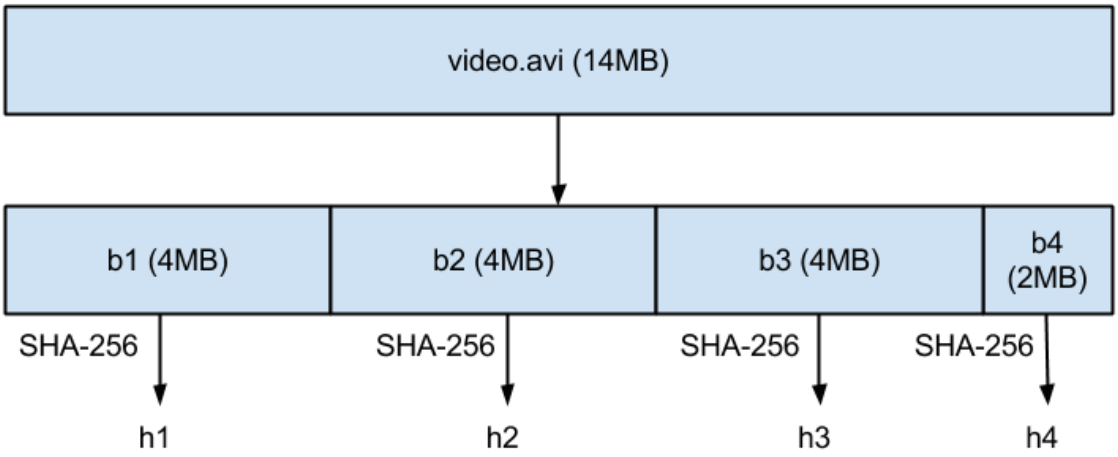


Рисунок 2 – фрагменты файла для синхронизации файлов Dropbox

2.5 Синхронизация данных в приложении Evernote

В алгоритме синхронизации Evernote EDAM (Evernote Data Access and Management) [5] существенным отличием от описанных ранее алгоритмов является тот факт, что конфликты решает клиент, а не сервер. Сервер не хранит состояние и не содержит никакой логики.

Для отслеживания изменений на клиенте у каждой записи выставляется флаг изменений (dirty flag).

Для разрешения конфликтов используется номер ревизии (Update Sequence Number). Для бинарных данных (blob) пересылаются не сами данные, а метаданные о том, где их можно извлечь. Бинарные данные обрабатываются отдельно от других данных.

Для обеспечения кроссплатформенности объекты для синхронизации и программные интерфейсы описываются языком описания интерфейсов Thrift (Thrift interface definition language). [6] После этого на основе IDL генерируется платформозависимые объекты. Таким образом поверх протокола http передаются бинарные объекты с клиентов на сервер.

Также различают полную и инкрементную синхронизацию.

Вместо того, чтобы отсылать весь объект целиком изменения синхронизируются небольшие порциями объектов (chunks).

2.6 Библиотека кроссплатформенной синхронизации

В диссертации [7] рассказываются детали реализации синхронизации на основе ведения журнала изменений. Данное решение реализовано на языке C++, для работы на основных платформах написаны специальные обертки, код можно вызывать из нативного для каждой платформы кода (C#, Swift, Java).

Для начала в серверные таблицы добавляется дополнительная колонка с GUID. GUID – 128 битный уникальный идентификатор. Данная сущность была придумана, чтобы убрать необходимость в централизованном узле для обеспечения уникальной идентификации информации в распределенных системах. Также в базу данных добавляется журнал изменений и дополнительная таблица с метаинформацией о синхронизации. В таблице 1 приведен пример записи журнала изменений.

Синхронизация начинается со стадии инициализации, при которой с сервера считывается схема, создается аналогичная схема на клиенте и заполняется данными.

Таблица 1 – Журнал изменений

Guid ряда	Операция	Раунд синхронизации
5551e6a3-932a-4d37-b534-40bdc9425cc7	Update	0

Журнал позволяет отслеживать изменения ряда, а также разрешать конфликты. При таком подходе необходимо выполнять оптимизацию журнала при синхронизации, чтобы не выполнять неактуальные действия с данными и избежать аварийное завершение приложения при попытке синхронизировать изменения в уже удаленном на другом устройстве ряде.

В предложенном алгоритме инициализация выполняется однажды, и схема не поддерживается в актуальном состоянии, не говоря уже о возможности менять схему на клиенте.

В работе [7] нет специализированного сервера, библиотека соединяется напрямую к локальной и удаленной базе данных. Это обеспечивает тот факт, что клиент разрешает конфликты. Данный момент является существенным нарушением безопасности системы. Сервер базы данных должен быть изолирован от мобильного приложения. Также следуют понимать, что не для всех баз данных есть коннектор, работающий под ARM-архитектуру.

Заключение

Синхронизация данных на протяжении нескольких десятков лет является насущной проблемой, о чем говорят регулярные публикации на эту тему. Кроме того, технологии претерпели значительные изменения и алгоритмы подстраивались под эти изменения.

В данной статье были рассмотрены различные алгоритмы синхронизации данных. В эволюции алгоритмов прослеживается тенденция к переносу логики синхронизации с сервера на клиент и все большего применения оптимистической синхронизации.

Список литературы

1. Saito Y. Optimistic Replication / Y. Saito, M. Shapiro // ACM Computing Surveys. – 2005. - P. 1-44.
2. Choi M.Y. A synchronization algorithm of mobile database for ubiquitous computing / M. Y. Choi, E.A. Cho, D.H. Park, J.Y. Bae, C.J. Moon, and D.K. Baik // NCM 2009 - 5th International Joint Conference on INC, IMS, and IDC. – 2009. – P. 416–419.
3. Stankovic T. Platform Independent Database Replication Solution Applied to Medical Information System. / T. Stankovic, S. Pesic, D. Jankovic, P. Rajkovic // 14th East-European Conference on Advances in Databases and Information Systems, NoviSad: Springer. – 2010. – P. 587–590.
4. Koorapati N. Streaming File Synchronization [Электронный ресурс]. – Режим доступа: https://blogs.dropbox.com/tech/2014/07/streaming-file-synchronization/
5. Engberg D. Evernote Synchronization via EDAM [Электронный ресурс] / D. Engberg, S. Hitchings // Режим доступа: http://dev.evernote.com/media/pdf/edam-sync.pdf
6. Главная страница фреймворка Apache Thrift [Электронный ресурс]. – Режим доступа: https://thrift.apache.org/
7. Kalapos G. Database synchronization between mobile devices and classical relational database management systems / G. Kalapos - Linz. - Магистерская диссертация университета Johannes Kepler. – 2015. – 120 p.

СИНХРОНИЗАЦИИ РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ МЕЖДУ МОБИЛЬНЫМИ УСТРОЙСТВАМИ С ИСПОЛЬЗОВАНИЕМ ОБЛАЧНОГО ХРАНИЛИЩА

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

Введение

Мобильные устройства стали неотъемлемой частью повседневной жизни человека и зачастую используются чаще, чем персональные компьютеры, поскольку они более доступны финансово и всегда под рукой. Мобильный трафик сети интернет превышает трафик с персональных компьютеров. Каждый третий человек на земле является владельцем смартфона.

Характерной особенностью мобильной связи являются неполное покрытие и нестабильное соединение. Связь заметно улучшилась в последнее время, однако при удалении от городских центров ситуация заметно хуже. В горах, пустынях и вблизи дикой природы связи нет вообще. Например, в горном Алтае связь есть только вдоль трасс и в населенных пунктах. В данном случае клиент-серверные приложения просто-напросто не функционируют.

Не для всех приложений требуется Интернет-соединение. Существуют также офлайн приложения. В век облачных технологий никто не хочет потерять данные. Фотографии синхронизируются в облачное хранилище, документы в Dropbox, Google Drive, One Drive. Однако такой сценарий работы предполагает, что файл редактируется только с одного устройства. А фотографии просто копируются в облачное хранилище, поскольку конфликтов в данном случае быть не может.

Рассматриваемый класс задач представляет интерес с научной точки зрения. В связи с большим количеством ограничений в каждом из алгоритмов синхронизации приходится идти на компромиссы. При улучшении одних характеристик другие ухудшаются. Высокая стоимость готовых решений, быстрое устаревание технологий, быстрое улучшение качества связи является характерной особенностью данной области.

Для работы с базами данных на мобильных устройствах как правило используются два взаимоисключающих подхода: клиент-серверное приложение, либо офлайн база данных. В данной работе рассматривается алгоритм синхронизации данных и схемы динамической неизвестной заранее по структуре базы данных с использованием облачного хранилища в качестве сервера синхронизации.

Описание структуры хранения данных

Описываемая в работе база данных подразумевает наличие метаданных. Сами данные реляционной базы данных хранятся в обычных таблицах. Для работы мобильного приложения помимо самих данных необходимо также хранить информацию о формах ввода, отчетах и других сущностей. На рисунке 1 представлен фрагмент диаграммы шаблона базы данных.

Для корректной работы приложения нужно обладать сведениями о метаданных:

- Database – сущность для описания информации о базе данных;
- Template – сущность для сохранения/загрузки шаблона;
- TableScheme – сущность для описания таблицы, содержит список колонок из которых состоит таблица;
- Column – колонка базы данных;

- MetaType – высокоуровневый тип данных, который умеет восстанавливать и записывать данные из низкоуровневых типов данных;
- Table – класс для работы с данными таблицы, позволяет извлечь ряды данных;
- DataRow – строка данных;
- EditForm – форма редактирования данных.

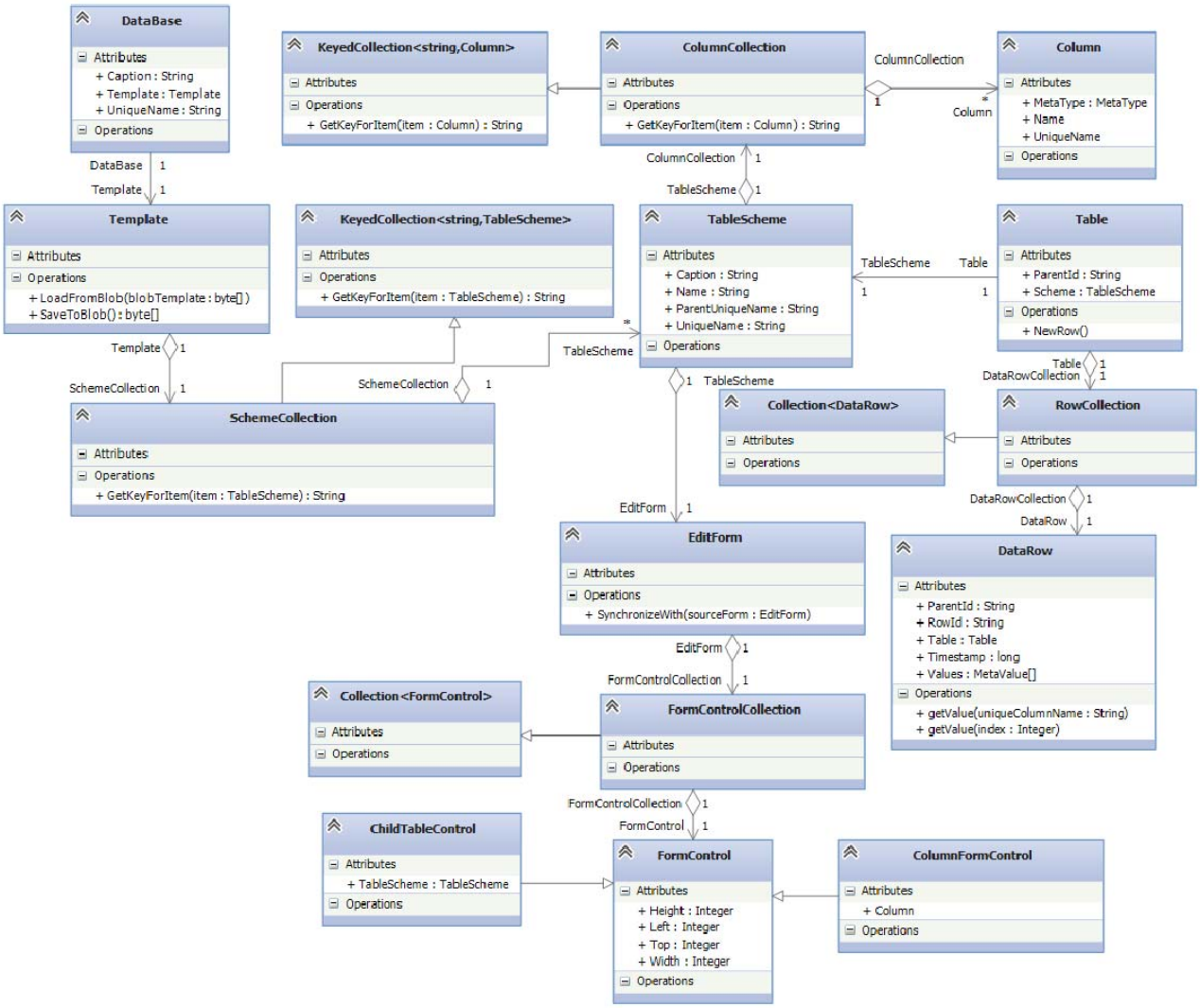


Рисунок 1 - Метаданные базы данных

Для корректной синхронизации требуется синхронизировать данные, а также метаданные. Рассмотрим, каким образом предлагается решать данные проблемы.

Синхронизация данных

Следует заметить, что в случае пассивного облачного хранилища логика синхронизации должна располагаться на клиенте. Современные мобильные устройства обладают достаточными мощностями для обработки синхронизации. При таком подходе следует понимать невозможность блокировки данных при синхронизации. В этом случае синхронизацию называют оптимистической и следует предусмотреть логику разрешения конфликтов [2].

Прежде всего для синхронизации требуется распределенный механизм генерации идентификаторов рядов. Для этого в каждую таблицу добавляем GUID-колонка. Однако при

таком подходе данные возвращаются не в том порядке, в котором их добавили. Для этого дополнительно добавляем колонку с датой добавления ряда Timestamp.

Для синхронизации данных требуется отслеживать изменения. Для этого предлагается дополнительно в базе данных вести специальную таблицу журнала изменений. Изменения было решено хранить с точностью до колонки, чтобы возникало меньше конфликтов на изменение и было меньше потерь данных при синхронизации. Т.е. наличие изменений в разных колонках не считается конфликтом.

Структура таблицы журнала изменений представлена в Таблице 1.

Таблица 1 – структура журнала изменений

Название	**Назначение**
GuidColumn	Идентификатор
TimestampColumn	Время изменения
ActionColumn	Тип изменений
TableUniqueNameColumn	Уникальное имя таблицы
ColumnUniqueNameColumn	Уникальное имя колонки
RecordGuidColumn	Уникальный идентификатор ряда с изменением
IsActualColumn	Актуальность изменений

При изменениях и удалениях ненужные более записи помечаются как неактивные и очищаются перед подготовкой файла-патча для сервера. Для ведения журнала в слое данных помимо самих манипуляций данных делается дополнительная запись в журнал изменений.

Типы изменений бывают трех видов: RecordAdded, RecordUpdated, RecordRemoved.

Для разрешения конфликтов изменений используется Timestamp в UTC.

Первый шаг синхронизации аналогичен существующим системам репликации данных. Как, например, описано в [1] при репликации MSSQL. Для синхронизации сначала необходимо выполнить шаг инициализации: загрузить моментальный снимок (snapshot) на сервер. Моментальный снимок – содержимое всей базы данных. В нашем случае это копия всех данных и метаданных. Последующие изменения накапливаются инкрементально в виде патчей.

Алгоритм синхронизации:

1. В случае, если синхронизация в облако происходит впервые: загрузить моментальный снимок в облако;
2. Считать идентификаторы файлов-патчей (имена файлов патчей) из облачного хранилища;
3. Отбросить патчи, которые уже обработаны. Для этого ведется специальная таблица с именами примененных патчей;
4. Применить патчи с сервера последовательно на клиенте согласно дате их загрузки на сервере;
5. Восстановить ссылочную целостность, сбросить связи на удаленные записи;
6. Выполнить упаковку журнала путем удаления неактуальных записей (IsActualColumn=0);
7. Выгрузить изменения из журнала в файл и загрузить на сервер. При считывании учитывать дата последней синхронизации и выгружать только необходимые изменения.

Синхронизация метаданных и схемы

Для синхронизации метаданных и схемы ведется отдельный журнал изменений. Сложность синхронизации метаданных в том, что помимо самих метаданных необходимо менять схему данных (таблицы с данными).

Синхронизация схемы данных выполняется непосредственно перед синхронизацией данных. В журнале синхронизации метаданных и схемы хранится тип действия с метаданными. В файле патча записывается весь шаблон, а также список действий, которые были выполнены с метаданными. Почти для каждой сущности метаданных (которую предполагается синхронизировать) есть ряд аналогичных изменениям данных действий: сущность добавлена/изменена/удалена. В случае, если при изменении метаданных нет надобности менять схему данных синхронизация предельно проста – необходимо считать тип действия из патча, извлечь из приложенного шаблона объект (или удалить из локального шаблона в случае удаления) и применить соответствующее действие к локальному шаблону.

Для синхронизации метаданных, которые влияют на схему данных помимо модификации шаблона дополнительно собираются изменения схемы данных, которые применяются к схеме базы данных после применения всех действий с метаданными.

Изменения схемы данных возникают при удалении/добавлении таблицы и колонок. Для каждой таблицы, с которой проводятся манипуляции со схемой данных ведется список добавленных/удаленных колонок. После обработки всех изменений метаданных, новые метаданные, а также информация о необходимых манипуляциях со схемой передаются слою данных, где перезаписываются метаданные, а также добавляются/удаляются колонки, таблицы.

Синхронизация форм ввода выполняется в стадии завершения слияния метаданных локальных и удаленных метаданных. Для синхронизации форм локальной (local) и удаленной (remote) определяется основная форма исходя из хронологии последних манипуляций с формой. Форма с более поздними изменения становится основной. Сначала в результирующую форму добавляются не удаленные элементы из основной формы, затем определяются не удаленные, вновь добавленные элементы. Эти элементы размещаются последовательно внизу под основной формой.

Заключение

В данной статье представлен оптимистический алгоритм синхронизации реляционной базы данных с использованием облачного хранилища без состояния.

К преимуществам данного подхода можно отнести независимость от конкретной базы данных, отслеживание изменений на уровне колонки, умение обрабатывать изменения схемы и данных, а также поддержка синхронизации неплоских иерархических типов данных.

К недостаткам данного подхода относятся: необходимость создания метаданных в качестве первичной информации о базе данных, отсутствие возможности организовать синхронизацию данных уже имеющихся баз данных.

В связи с описанными выше недостатками простого способа сделать универсальную библиотеку синхронизации данных нет, поскольку у каждой конкретной задачи есть свои метаданные и реализация слияния сущностей разнится от задачи к задаче. Однако в алгоритме синхронизации представлены общие принципы синхронизации, которые можно использовать для реализации синхронизации в других проектах.

Список литературы

1. Meine S. Fundamentals of SQL Server 2012 Replication / S. Meine. - Simple Talk Publishing, 2013. – 376 p.
2. Saito Y. Optimistic Replication / Y. Saito, M. Shapiro // ACM Computing Surveys. – 2005. - P. 1-44.

АНАЛИЗ ТОНАЛЬНОСТИ КОРОТКИХ ТЕКСТОВ НА ОСНОВЕ СЕМАНТИЧЕСКОГО ГРАФА

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

1. Введение

За последнее десятилетие интернет стал неотъемлемой частью повседневной жизни. Социальные сети, онлайн-магазины, новостные порталы, образовательные ресурсы становятся все более популярными. По данным ВЦИОМ число людей, которые не пользуются интернетом, в период с 2011 по 2017 год сократилось вдвое, и вдвое возросло число людей, которые пользуются им ежедневно. В таких условиях объем информации, которая генерируется пользователями, растет экспоненциально.

Среди всей информации, попадающей в сеть, особый интерес представляет мнение пользователей по каким-либо вопросам, будь то отзыв о новом товаре или отклик на политическое событие. Представители бизнеса, политики, общественные деятели заинтересованы в изучении данной информации. Однако обрабатывать ее вручную не представляется возможным, и поэтому используются системы анализа тональности текстов.

Анализ тональности представляет собой класс методов в компьютерной лингвистике, направленных на извлечение эмоций и мнений из текстов на естественном языке.

Существует несколько подходов к анализу эмоциональной составляющей текста. Одни основаны на использовании словарей, другие – на машинном обучении, третьи представляют собой гибрид первых двух. Построение словарей и последующая оценка тональности на их основе хорошо подходит для небольших текстов, при анализе которых нет необходимости в глубоком семантическом разборе.

2. Цель работы

На сегодняшний день формирование словарей, используемых для оценки тональности, происходит двумя способами – либо автоматически, либо посредством ручной разметки. Оба способа имеют существенные недостатки. В случае ручной разметки требуются колоссальные ресурсы, чтобы покрыть достаточный объем лексического запаса языка. При автоматической разметке даже большой объем обучающей выборки не может гарантировать, что эмоциональные метки будут присвоены достаточному количеству слов, существующих в современном языке.

Перед авторами данной работы стояла цель – на основе обучающей выборки сформировать тоновый словарь, в котором для каждого понятия оценки тональности вычислялись бы автоматически, а так же предложить механизмы, позволяющие расширить тоновый словарь и компенсировать недостаток словарного запаса.

3. Модель оценки тональности

На этапе исследования была использована обучающая выборка, представленная в работе [1]. Корпус состоит из 114,911 положительных, 111,923 отрицательных записей. Каждая запись представляет собой twitter-пост длиной как минимум 40 символов и содержащий только один тип эмоций – положительный или отрицательный.

По корпусу текстов были подсчитаны статистические данные. Учитывались два основных показателя – количество вхождений слова в положительный и отрицательный наборы твитов. При формировании лексикона учитывались только значимые части речи

(существительное, прилагательное, глагол, причастие, деепричастие или наречие). Все слова приводились к начальной форме.

На втором этапе рассчитывались производные характеристики. Пусть f_{pw} – частота вхождения слова w в корпус позитивных твитов, а f_{nw} – в корпус негативных. Рассмотрим следующие величины:

1. Эмоциональная окраска слова w:

$$s_w = \frac{f_{pw} - f_{nw}}{f_{pw} + f_{nw}},$$

2. Степень распространенности слова w в словаре W:

$$o_w = \frac{f_{pw} + f_{nw}}{\max\limits_{l \in W}(f_{pl} + f_{nl})}.$$

Значения величины s изменяются в пределах [-1;1], значению -1 соответствуют «сильно негативные» слова, значению 1 – «сильно позитивные». Величина o характеризует степень принадлежности слова к числу общеупотребимых. На этапе предварительного анализа ожидалось, что при больших значениях o_w значение s_w будет небольшим, и наоборот. Предположения строились на основе природы естественного языка – слова, употребляемые очень часто, скорее всего, не имеют ярко выраженной эмоциональной окраски и формируют «нейтральный» каркас языка.

Таким образом, каждому слову поставлена в соответствие двумерная оценка $\boldsymbol{t_w = (s_w, o_w)}$. Графическое представление распределения оценок представлено на рисунке 1.

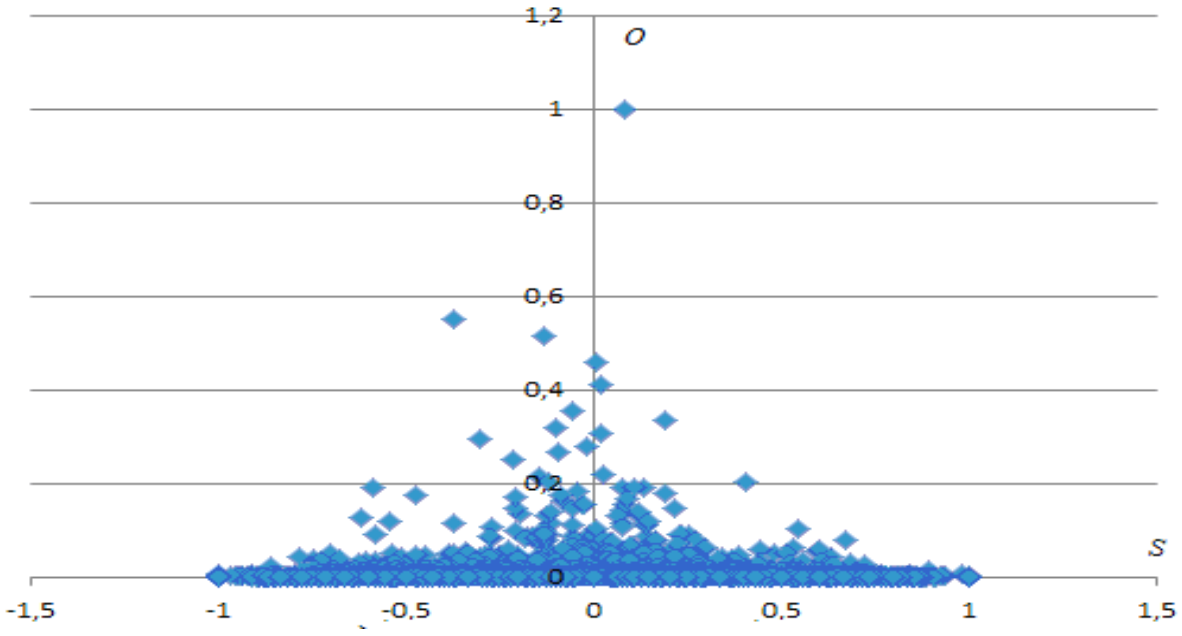


Рисунок 1 – Графическое представление оценок тональности слов

Представленный график подтверждает высказанные предположения.

Кроме того, оценивалось распределение слов по шкале эмоциональной окраски. Была построена гистограмма, представленная на рисунке 2.

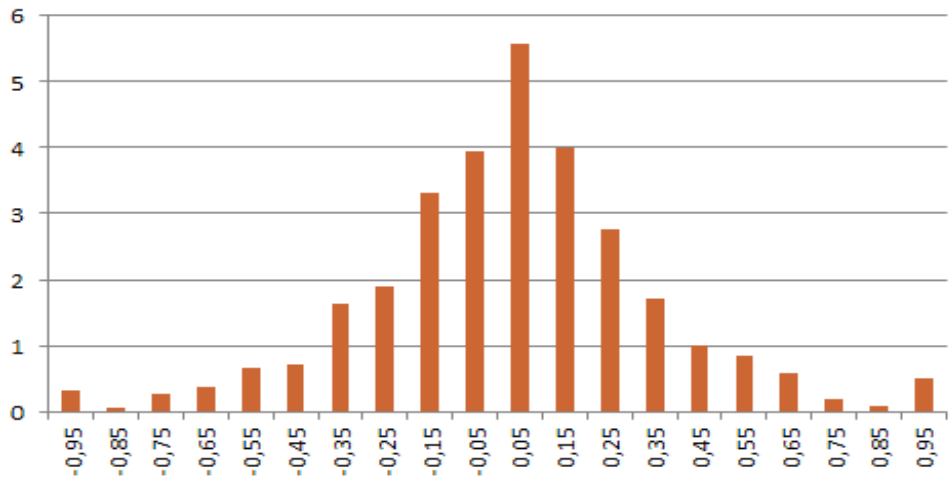


Рисунок 2 – Гистограммы распределения слов по шкале эмоциональной окраски.

Характер распределения слов по шкале эмоциональной окраски не противоречит здравому смыслу – основную массу словарного запаса составляют слова без ярко выраженной эмоциональной нагрузки, и тональность текстов формируется за счет употребления отдельных слов с сильной эмоциональной составляющей.

4. Взвешенный лексикон как источник семантической информации

В силу того, что современный интернет-язык обладает своей спецификой, связанной с преобладанием простых слов и конструкций, огромный пласт словарного запаса остается незатронутым в процессе формирования тонового лексикона.

Необходим механизм, который позволил бы делать косвенные выводы о тональности слов, встречающихся недостаточно часто для прямого вычисления оценки. Подобные выводы можно делать, основываясь на семантических связях между словами (к примеру, синонимии и обобщения). На сегодняшний день существуют подобные словари, один из них - WordNet Affect[2], в основу которого положены синсеты – наборы слов, связанных между собой отношением синонимии. Однако в данном словаре другие семантические отношения связывают не отдельные слова, а целые синсеты, что существенно затрудняет анализ тональности в определенном контексте.

В качестве источника семантической информации предлагается использовать разработанный авторами лексикон [3]. Данный лексикон представляет собой граф G, в вершинах которого расположены слова, а дуги представляют собой направленные взвешенные связи трех типов – синонимии, ассоциации и обобщения. Граф автоматически построен на основе Толкового словаря С.И.Ожегова и Словяря синонимов русского языка, в результате чего отдельные связи устанавливаются между словами, имеющими контекстную зависимость в общеупотребительном контексте.

Введенные отношения между словами позволяют устанавливать степень близости слов g_A и g_B. Для вычисления словарной близости слов, связанных напрямую, используется формула:

$$Dict(g_A, g_B) = Max(Syn(g_A, g_B), Def(g_A, g_B), Assoc(g_A, g_B)),$$

где $Syn(g_A, g_B)$, $Def(g_A, g_B)$, $Assoc(g_A, g_B)$ – веса отношений синонимии, обобщения и ассоциации между g_A и g_B соответственно. Для слов, которые не связаны отношениями напрямую, вычисляется максимальное произведение значений ***Dict*** на пути между ними в

графе. Основываясь на понятии близости, можно строить семантические окрестности слов и получать понятия, в заданной мере связанные с ключевым.

В работах [3,4] показано, что изменение начальных весовых характеристик графа существенно меняет характер извлекаемой информации. Верно подобранные значения позволяют извлекать релевантную информацию в рамках отдельной задачи.

5. Метод вычисления тональности коротких текстов

Пусть имеется фраза *S* на русском языке. Введем следующие обозначения:

w_i – отдельные слова, для которых напрямую вычислена оценка тональности t_{wi},

g_i – отдельные слова из графа G, для которых неизвестна оценка тональности в W.

Будем считать, что различные слова должны учитываться при оценке тональности с различным весом. Для исследования была предложена весовая функция $WS(w_i)$. Ключевым параметром для построения весовой функции является степень покрытия тонового словаря в процентах (выбирается значение o_{max} так, чтобы нужный процент слов имел меньшую степень распространения в выборке). Характер функции таков, что наибольшую значимость имеют слова с выраженной окраской, которые при этом достаточно часто появляются в выборке. Нейтрально окрашенные слова или редко употребляемые имеют низкий вес. Один из вариантов весовой функции приведен на рисунке 3.

Вычисление тональности фразы происходит в следующем порядке:

1. Выявляются все слова w_i с известной оценкой, вычисляется вес каждого слова.
2. Для остальных слов формируются семантические окрестности в графе G с заданным порогом близости.
3. Из окрестности для каждого незнакомого слова выбираются слова w_i с известной оценкой, вычисляются их весовые характеристики, затем вычисляется центр масс полученного набора слов. Вычисленные величины **(*s*,*o*)** представляют собой синтетическое «слово», которого нет в первоначальной выборке.
4. Для каждого из синтетически полученных «слов» вычисляется весовая функция с некоторым коэффициентом, который определяет степень доверия системы словарю.
5. Весь полученный набор слов рассматривается как единая система, для которой вычисляется центр масс. Координата s вычисленного центра масс представляет собой итоговую оценку тональности фразы.

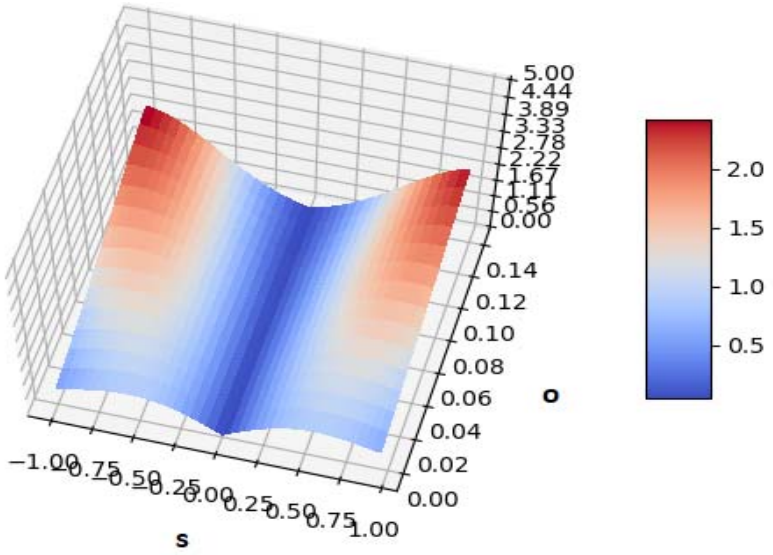


Рисунок 3 – Весовая функция при o_{max}=0.15

6. Результаты работы и перспективы развития

В результате предварительных экспериментов были получены оптимистичные результаты – система способна определить эмоциональную окраску произвольных коротких текстов, однозначно отличает позитивные тексты от негативных. Однако существуют сложности с определением силы эмоциональной окраски.

Для повышения качества оценки планируется сделать следующее:

1. Расширить тоновый словарь за счет биграмм. Из обучающей выборки будут извлечены пары слов, расположенных рядом и связанных между собой семантически. Это так же позволит работать с устойчивыми словосочетаниями как с единым целым. Биграммы планируется оценивать статистически, так же как отдельные слова.
2. Предложить механизм обработки отрицаний. Отрицания – одна из ключевых проблем оценки тональности, их корректная обработка существенно увеличивает точность результатов.
3. Рассмотреть возможность постоянного дообучения системы на основе анализируемых текстов.

7. Заключение

Основываясь на полученных данных, можно сделать вывод, что описанная модель оценки тональности пригодна для работы с короткими текстами на русском языке, а семантический граф позволяет восполнить недостаток словарного запаса обучающей выборки. Тем не менее, существует ряд возможностей для улучшения качества оценки за счет биграмм, первичного семантического анализа с выявлением отрицаний и постоянного самообучения системы.

Список литературы

1. Рубцова Ю.В. Построение корпуса текстов для настройки тонового классификатора // Программные продукты и системы. – 2015. - №1(109). – С. 72-78.
2. Strapparava C., Valitutti A. WordNet-Affect: an Affective Extension of WordNet. // 4th International Conference on Language Resources and Evaluation (LREC 2004), May 2004: Proceedings. – Lisbon, 2004. – P. 1083-1086.
3. Крайванова В.А., Кротова А.О, Крючкова Е.Н. Построение взвешенного лексикона на основе лингвистических словарей // Материалы Всероссийской конференции «Знания — Онтологии — Теории». – 2011. – Том 2. – С. 32-38.
4. Казаков М.Г., Крючкова Е.Н. Классификация сложных изображений на основе семантического графа понятий. // Прикладная информатика. – № 6(54). – 2014. – С. 79-89.

ВЕБ-ПРИЛОЖЕНИЕ ОБМЕНА СООБЩЕНИЯМИ С ВОЗМОЖНОСТЬЮ ИНТЕГРАЦИИ В КОРПОРАТИВНЫЕ СИСТЕМЫ

Варзанова О.О. – студент
Алтайский государственный технический университет (г. Барнаул)

Мессенджеры представляют собой программное обеспечение, направленное на мгновенный обмен сообщениями как текстовыми, так и аудио, и видео режимами. Современные мессенджеры имеют расширенный коммуникационный спектр возможностей для полноценного и комфортного функционирования пользователей в сети, которые обладают не только ключевыми функциями, но и множеством других, например, обмен файлами, веб–конференции. Существует большое количество мессенджеров, но в данной

статье мы рассмотрим лишь некоторые из них, которые являются основными лидерами на рынке "виртуального общения".

Наиболее популярными мессенджерами в настоящее время являются: WhatsApp, Telegram [1].

Рассмотрим достоинства и недостатки мессенджер WhatsApp:

- Интуитивный интерфейс, также в мессенджере возможно менять внешний вид и формировать по–своему вкусу фоновый облик приложения.
- Система сохранения информации обладает двустороннее шифрование, что позволит сберечь переписку пользователей.
- Определение географического местоположения пользователя, удобная и вспомогательная функция в экстренных ситуациях.
- Автоматическая синхронизация контактов телефонного справочника пользователя, которые уже используют данное приложение.
- Возможность организации групповых чатов, что позволяет осуществить обмен с несколькими собеседниками.

Но преобладает ряд отрицательных моментов: приложение не работает в рамках отсутствия sim-карты, связи с тем, что регистрация пользователя жестко привязана к номеру телефона; имеется только мобильная и web версии приложения.

Telegram – значительно новый мессенджер на рынке ИТ-продукции, который закрепил себя как самый безопасный и защищенный от взлома мессенджер. Одним из основных привилегий приложения являются функции: персонального оформления, создания групповых чатов и определения географического местоположения. Однако присутствует и ряд существенных отличий:

- Системы защиты данных, основанные не только на двустороннем шифровании, но и на создание временных секретных чатах, которые могут автоматически удаляются в установленный срок.
- Присутствует возможность создания собственных стикеров.
- Возможность создавать множество разных каналов по любой тематике.
- Имеется функция «автоматических собеседников» или ботов, умеющих отвечать на несложные запросы и предоставлять информацию.

Также у приложения имеется, значительный, минус – отсутствие видеозвонков.

В рамках корпоративных решений необходимо иметь не такой широкий функционал, как предлагают выше перечисленные приложения. Требуется осуществить унификацию назначений для узкоспециализированной системы с целью: минимизировать механические и временные затраты при работе клиента в мессанджере, направив взаимодействие клиента и приложения в русло по решению конкретной проблемы, без внедрения излишних внешних факторов.

Участниками корпоративной системы являются пользователь и группы, в которых располагаются представителя отдела сопровождения. Клиенты владеют информацией о внутренней иерархии организации и знают, кому им стоит обратиться за помощью. Также за каждым клиентом закреплен куратор (не из службы поддержки), к которому они могут обращаться за помощью. В общем случае, клиент сам решает, к кому ему обратиться – в отдел сопровождения или к куратору.

Пользователь не должен видеть сообщения в поддержку других пользователей, если то индивидуальный чат. Если групповой, т.е. человек с группой, то любой человек из группы может ответить пользователю, и все эти сообщения должны быть видны в чате между пользователем и группой. Это необходимо, чтобы все члены поддержки видели весь диалог с пользователем (не задавали одних и тех же вопросов, видели предыдущие рекомендации и т.п.). Представитель группы может входить в несколько групп, а группа по сопровождению может входить только в один отдел. Преобладает возможность индикации непрочитанных

сообщений как в чате группы, так и в личном диалоге. При наличии непрочитанных сообщений: открыть чат с сообщениями и отправить ответы на интересующие вопросы.

Рассмотрим возможные сценарии пользовательских историй, чтобы выявить какие, именно, необходимы требования и функции для корпоративного мессенджера.

Общие сведения:

- индикация непрочитанных сообщений;
- дата и время отправки (прибытия) сообщения;
- содержимое сообщения (скриншот экрана, текст) и ФИО пользователя либо представителя группы.

Сценарий пользователя и группы:

1) У пользователя возникла проблема, на рабочем окне появилась неизвестная ошибка. Как исправить он не знает, следовательно, ему нужно эту ошибку показать представителям из технической поддержки.
2) Поверх ошибки пользователь открывает модальное окно, в котором выбирает адресата (группу), в которую будет набирать сообщение об ошибки.
3) Пользователь набирает сообщение с вопросом о проблеме, и прикреплять образ экрана с ошибкой. Могут быть варианты исхода действий:
   а. Если ранее возникала подобная ошибка, то пользователь выбирает группу и в строке поиска вводит эту проблему, где при помощи поиска по чату можно будет найти необходимый ответ.
   б. Если эта ошибка возникла впервые, то пользователь набирает ее в текстовом поле и прилагает скриншот.
4) Отправляет сообщение в чат группы, где прослеживается: какие представители поддержки онлайн в данный момент.
5) Затем пользователь может остаться ожидать ответ, либо продолжить свою работу, а как индикатор покажет наличие сообщений, зайти в приложение и прочитать ответ:
   а. Если инструкции по устранению проблемы пользователя удовлетворили, то он завершает работу в чате и возвращается к своей.
   б. Если инструкции не помогли или возникли еще какие-либо проблемы, то пользователь возвращается к шагу №4 и повторяет все действия.

Сценарий пользователя и представитель групп (индивидуальный чат):

1) У пользователя возникла проблема, на рабочем окне появилась неизвестная ошибка.
2) Поверх ошибки пользователь открывает модальное окно, в котором выбирает адресата (члена группу, например Сидорова Ивана) и проверяет его наличие (индикатор онлайн/офлайн), которому будет в личном чате вводить сообщение, где опишет возникшую проблему. Варианты действий:
   а. Если индикатор офлайн и проблема не была ранее решена с эти представителем, то пользователь возвращается на шаг №2.
   б. Если индикатор офлайн, но проблеме ранее решалась с данным адресатом, то пользователь переходит на шаг №3.
3) Работа с текстом сообщения о проблеме. Могут быть варианты исхода действий:
   а. Если ранее возникала подобная ошибка, то пользователь выбирает группу, в которой преобладает необходимый адресат (Иванов Иван) и заходит в их личный чат, где в строке поиска вводит эту проблему, где при помощи поиска по индивидуальному чату находит ответ.
   б. Если эта ошибка возникла впервые, то пользователь набирает ее в текстовом поле и прикрепляет скриншот.
4) Отправляет сообщение в индивидуальный чат, где прослеживается: какие представители поддержки онлайн в данный момент.

5) Затем пользователь может остаться ожидать ответ, либо продолжить свою работу, а как индикатор покажет наличие сообщений, зайти в приложение и прочитать ответ:
   а. Если инструкции по устранению проблемы пользователя удовлетворили, то он завершает работу в чате и возвращается к своей работе.
   б. Если инструкции не помогли или возникли еще какие-либо проблемы, то пользователь возвращается к шагу №3 и повторяет все действия.

Если все сообщения имеют ответ, и больше нет вопросов в личном чате, то пользователь выходит из модального окна и продолжает свою работу.

Все выше изложенное продемонстрируем на диаграмме активностей (рисунок 1), с помощью которой можно отобразить бизнес–процессы предприятия [2].

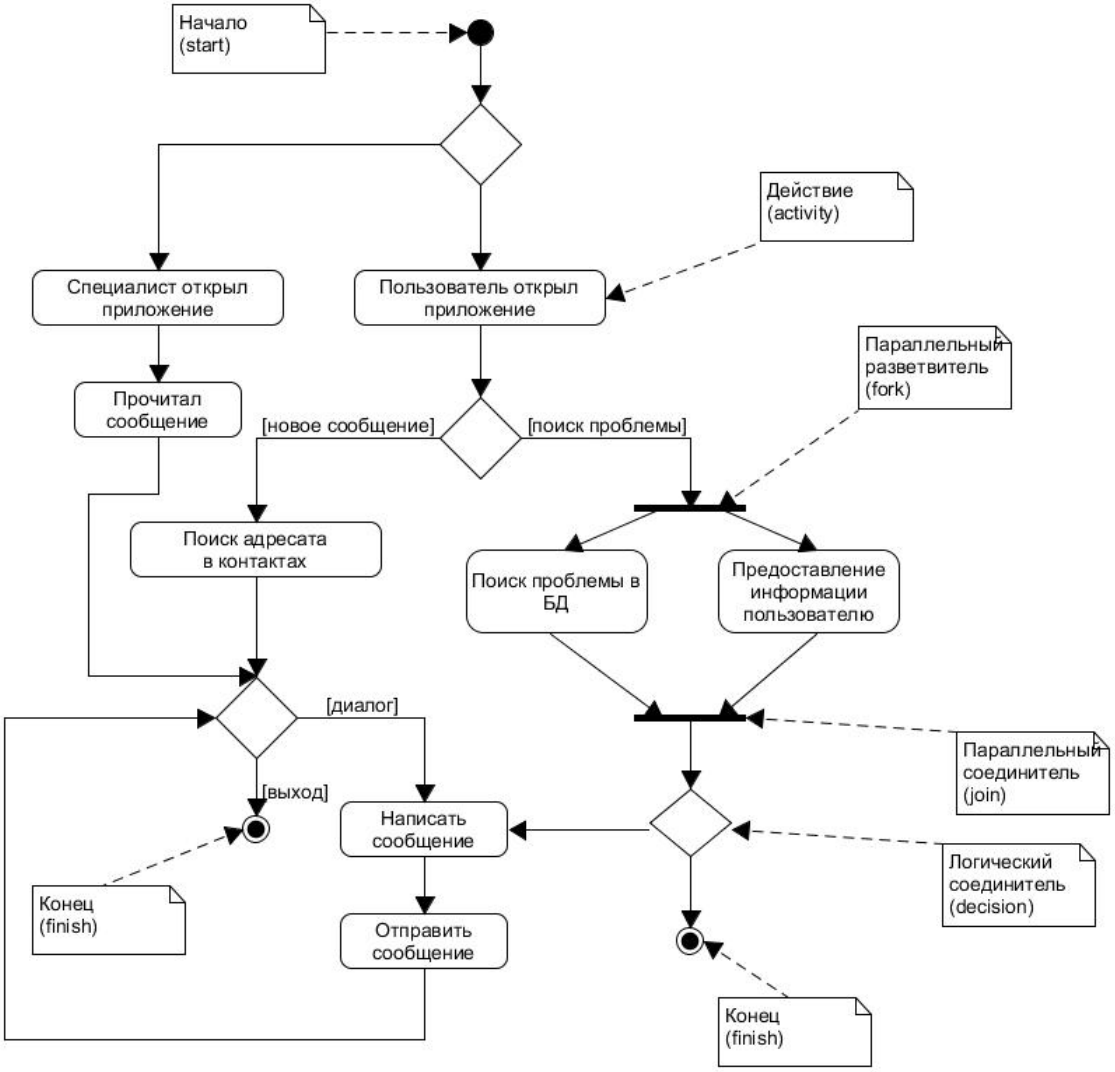


Рисунок 1 – Диаграммы активностей (activity diagrams)

Список литературы

1. Россияне все чаще отказываются от сотовой связи в пользу звонков в мессенджерах [Электронный ресурс] // www.dgl.ru – Режим доступа: https://www.dgl.ru/news/rossiyane-vse-chashhe-otkazyvautsya-ot-sotovoy-svyazi-v-polzu-zvonkov-v-messendjerah_13254.html.
2. Фаулер М. UML. Основы, 3–е издание. – Пер. с англ. – СПб: Символ–Плюс, 2004. – 192 с.

ПРОЕКТИРОВАНИЕ СИСТЕМЫ АНАЛИЗА ПРОИЗВОДИТЕЛЬНОСТИ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ НА ОСНОВЕ ФУНКЦИОНАЛЬНОЙ ДЕКОМПОЗИЦИИ

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

В настоящие время наука сильно опирается на теорию, эксперименты, наблюдения, а также математическое моделирование. Обработка большого количества исходных данных требует масштабные затраты временных ресурсов для их выполнения. Связи с этим становятся актуальны многопроцессорные системы, которые включают в себя программы распараллеливания при помощи того или иного алгоритма на параллельные ветви, каждая из которых осуществляет свою работу на отдельном процессоре. Благодаря использованию n-процессорной системы: все ресурсы направлены на выполнение одной программы, что позволяет ожидать n-кратного ускорения выполнения системы.

Параллельная программа – это множество взаимодействующих параллельных процессов. Основной целью распараллеливания процессов при выполнении вычислительных задач является эффективность и ускорения параллельных алгоритмов. Параллельные программы имеют ряд унифицируемых особенностей таких, как осуществление управления работой множества процессов; организация по обмену данными между процессами; значительно, уменьшается детерминизм поведения из-за асинхронности доступа к данным; формируются проблемы масштабируемости программы и балансировки загрузки вычислительных узлов; превалируют нелокальные и динамические ошибки; возникают возможности тупиковых ситуаций.

В основание показателей оценки располагается эффективность параллельной программы, а именно:

- **Ускорение параллельной программы (алгоритма)**, получаемое при запуске программы на системе с p процессорами, – это отношение $S = \frac{T_1}{T_p}$, где T_1 – время выполнения программы на одном процессоре; T_p – время выполнения программы на системе из p процессоров.
- **Эффективность** использования параллельной программой ресурсов многопроцессорной вычислительной системы при решении задачи определяется соотношением $E_p = \frac{S_p}{p}$. При создании высокопроизводительных программ необходимо стремиться к тому, чтобы $S_p \to p$ и $E_p \to 1$.

Фундаментальной проблемой параллельных алгоритмов является неразумное распределение балансируемой рабочей нагрузке между процессов. В результате, процессы с более легкой нагрузкой будут простаивать, а те, у кого более тяжелая нагрузка будут располагаться в состояние занятости выполнения своих задач.

Рассмотрим различные пути решения, в виде ряда схем для сопоставления задач на процессы с основным представлением о балансировки рабочей нагрузки выполняемых задач на процессах и минимизации их временных ресурсов, а именно, простоя. Существует два методы сопоставления, используемые в параллельных алгоритмах: статические и динамические. А также на основании парадигмы параллельного программирования и

характеристики задач можно провести анализ, по результатам которого будет выявлено какой из методов сопоставление наиболее приемлемое для той либо иной поставленной задачи.

Методы статического сопоставления распределяют задачи между процессами до выполнения алгоритма. Для реализации оптимального сопоставление необходимо не только обладать знаниями о размерах задач, данных, связанных с задачами; характеристиками межзадачных взаимодействий, но и парадигму параллельного программирования. При этом, чтобы получить оптимальную картину для неравномерных задач потребуется разрешение NP-полной проблемы. Однако для многих практических случаев относительно недорогие эвристики обеспечивают довольно объективные и, достаточно, приближенные решения оптимальной задачи статического отображения.

Методы динамического сопоставления распределяют работу между процессами во время выполнения алгоритма. Данная методика является наиболее подходящей, когда размеры задач неизвестны; так как статическое сопоставление может привести к серьезному дисбалансу распределения нагрузки между процессами.

Оба метода вносят весомый вклад в разработку алгоритмов для проектирования параллельных программ в соответствии с выполняемой задачей. Но, заметим, что алгоритм, направленный на использование динамического сопоставления является наиболее сложным, особенно в парадигме программирования передачи данных; в свою же очередь, метод статического сопоставления имеет, сравнительно, облегченный процесс разработки и программирования.

Авторам данной статьи будет рассмотрен, а впоследствии и реализован, метод статического сопоставления, который используется в комбинации с декомпозицией, основанной на разделении данных, в виде массивов.

Блочные распределения является способом распределения массива, и осуществляет назначение однородных смежных частей массива различным процессам. Приемлемость данного распределения массивов надлежит, когда имеется локальность взаимодействия, а именно, для вычисления элемента массива требуются другие соседние элементы в этом массиве. Благодаря данному распределению можем произвести балансировку нагрузки на множество параллельных вычислений, которые работают с многомерными массивами.

Рассмотрим матричное умножение n–мерных матриц: $C = A$ x B – одним из способов разложения является разбиение выходной матрицы C. Так как каждый элемент матрицы C выполняется по аналогичному алгоритму вычисления, то можно сбалансировать нагрузку, используя одно– или двумерное распределение блоков для равномерной концентрации вычислений между p доступными процессами. Получим, что при одномерном разбиении по строкам каждый процесс должен получить доступ к соответствующим (n / p) – строкам матрицы A и всей матрицы B, как показано на рисунке 1(a) для процесса P_5. В свою очередь, при двумерном распределении каждый процесс должен получить доступ к $(n / \sqrt{p})$ строкам матрицы A и $(n / \sqrt{p})$ столбцов матрицы B, как продемонстрированно на рисунке 1(b) для процесса P_5.

Таким образом, общий объем данных, которые необходимо получить для двух типов распределения блоков между p процессами составляет: в первом случаи – $O(n^2 / p + n^2)$, а во втором – $O(n^2 / \sqrt{p})$, что, значительно, меньше, чем в первом случаи.

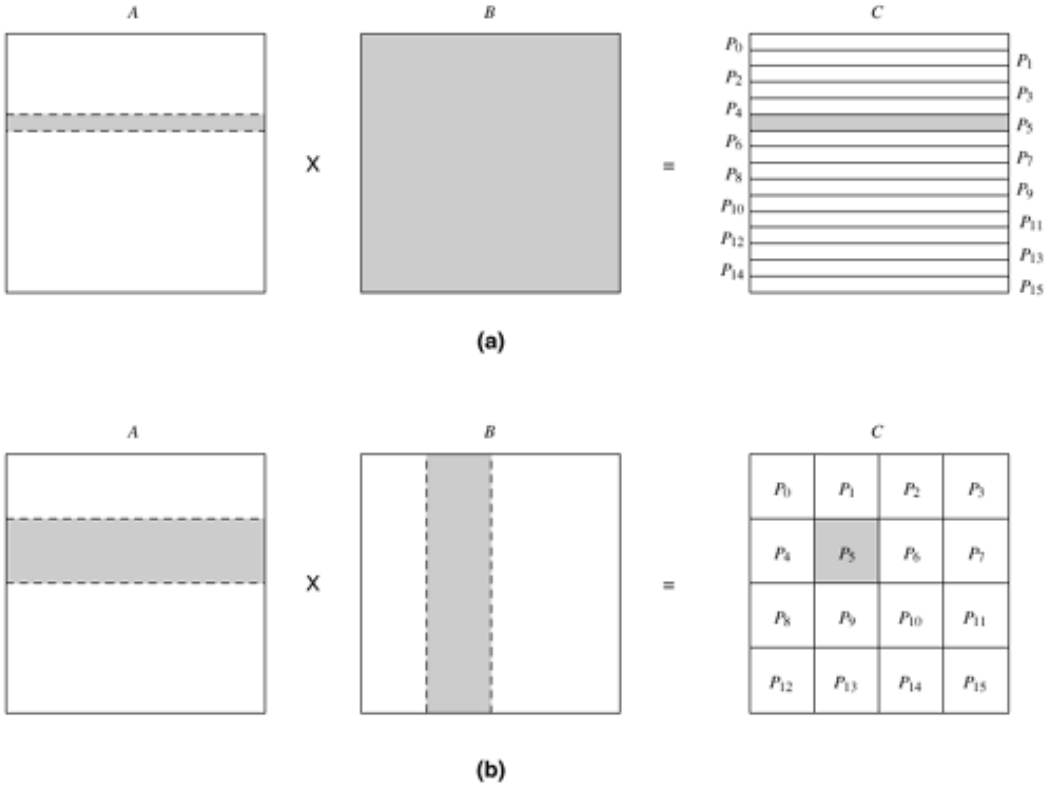


Рисунок 1 – Обмен данными необходим для матричного умножения

В дальнейшем, разумнее будет вести разделения методов для распараллеливания в зависимости от конкретно поставленной задачи, а именно этого можно достичь, благодаря разделению программы с помощью того или иного механизма на параллельные ветви, каждая из которых выполняется на отдельном процессоре. Статическая балансировка эффективна при условии, что априорной информации достаточно для предварительного распределения общей вычислительной нагрузки поровну между процессорными узлами. Динамической балансировки загрузки перед стартом процесса вычисления не владеет информацией, какие именно узлы сетки будут обработаны тем или иным процессором, а именно, процессоры в ходе обработки поступающих данных получают задания динамически, что обеспечивает равномерную загрузку процессорных узлов при наличии большого набора независимых заданий.

Список литературы

1. Mapping Techniques for Load Balancing [Электронный ресурс]. – Режим доступа: http://parallelcomp.uw.hu/ch03lev1sec4.html / Introduction to Parallel Computing
2. Афанасьев К.Е. Многопроцессорные вычислительные системы и параллельное программирование: Учебное пособие / Афанасьев К.Е., Стуколов С.В., Демидов А.В., Малышенко В.В.; Кемеровский госуниверситет. – Кемерово: Кузбассвузиздат, 2003. – 182 с.
3. Старченко А.В., Берцун В.Н. Методы параллельных вычислений. – Томск: Издательство Томского университета, 2013. – 224 с.

ПРОЕКТИРОВАНИЕ СИСТЕМ ДЛЯ РАСЧЕТА, ВИЗУАЛИЗАЦИИ И МОДЕЛИРОВАНИЯ СТРУКТУР ГЕННОЙ ИНЖЕНЕРИИ

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

Актуальность

Развитие современной генетики, науки о закономерностях наследственности и изменчивости организмов, опирается на научные труды австрийского исследователя Грегора Менделя. В том числе центральная концепция современной молекулярной генетики - проблема гена - берет свое начало в результатах фундаментальных работ Менделя, так как открытые им гены - это участки в клетке молекул ДНК, а механизм их состоит в кодировании белков, регулирующих все химические реакции в организмах. Успехи генетики в значительной мере связаны именно с основанным Менделем гибридологическим анализом.

Фундаментальные законы наследственности

Грегор Мендель сформулировал три фундаментальных закона наследственности, которые дали начало появлению науке генетике с момента их научного признания обществом. Законы были сформулированы после проведения ряда обширных экспериментов с тщательно отобранными сортами гороха, самоопыляющегося растения, особи которого легко дают чистые линии благодаря собственно самоопылению.

Первый закон Менделя. Доминирование или закон однообразия первого поколения. При скрещивании двух особей, различающихся по одной паре альтернативных признаков, потомство единообразно по фенотипу и проявляет доминантный признак одного из родителей, но при этом гетерозиготно по генотипу [5].

Второй закон Менделя. Закон расщепления при моногибридном скрещивании. При скрещивании двух гетерозиготных особей первого поколения в потомстве второго поколения наблюдается расщепление по фенотипу в соотношении 3 : 1 и по генотипу в соотношении 1 : 2 : 1 [4].

Третий закон Менделя. Закон независимого наследования. При скрещивании двух особей, отливающихся друг от друга по двум и более парам альтернативных признаков, гены и соответствующие им признаки наследуются независимо друг от друга и комбинируются во всех возможных сочетаниях [2].

При иллюстрации законов Менделя используются так называемые таблицы Пеннета, они же решетки Пеннета (названы именем предложившего их использование Реджинальда Пеннета). Они представляют собой двумерные таблицы и выступают в качестве инструмента графической записи для определения сочетаемости аллелей из родительских генотипов [3]. Вдоль одной стороны такой таблицы располагают женские гаметы, а вдоль другой - мужские. Эти решетки позволяют наглядно представить генотипы и фенотипы, которые получаются при скрещивании родительских гамет.

Множественный аллелизм

В изложенных выше тезисах один и тот же локус гомологичных хромосом представлен только двумя аллелями. Однако, помимо такого представления, может быть и другое, гораздо более сложное: один и тот же ген может изменяться в ряд состояний, которых бывает немалое количество. Такое многократное мутирование одного и того же гена — это серия множественных аллелей, а само явление — множественный аллелизм [8].

Исследование множественного аллелизма имеет прикладное применение в различных областях, в том числе и в наследовании групп крови человека. Это четыре известные группы

крови, которые обозначаются следующим образом по системе АВ0: 0, А, В, АВ (первая, вторая, третья и четвертая группы крови соответственно), они не изменяются в течение жизни человека. При этом наследование групп осуществляется настолько четко, что это используется в судебной медицине для исключения отцовства и в некоторых других случаях. Поэтому понимание множественного аллелизма не менее важно, чем трех классических законов Менделя.

Однако это не вся важность множественного аллелизма в группах крови человека: если человеку провести переливание крови, несовместимой с его родной группой крови, то он может погибнуть. Почему группы крови оказываются несовместимыми и приводят к таким последствиям? Происходит это потому, что антитело α агглютинирует эритроциты групп крови А и АВ, а антитело β в свою очередь — эритроциты групп крови В и АВ [7].

При рассмотрении множественного аллелизма так же можно применять решетки Пеннета.

Описание разработанного программного обеспечения

Предлагаемое программное обеспечение позволяет моделировать генетические эксперименты, которые не только иллюстрируют классические законы Менделя, но и позволяют проводить генетические эксперименты по видам скрещивания для лучшего понимания основ генетики и природы живых организмов. Предлагаемое решение обладает следующими характеристиками:

- приложение опирается исключительно на установившиеся еще со времен Менделя понятия генетики,
- интерфейс использует классические генетические термины и обозначения,
- приложение гибко работает с решеткой Пеннета, моногибридным и дигибридным скрещиванием,
- в качестве результатов анализа выводится результирующие вероятности соотношений генотипа и фенотипа на основе математической статистики, применяемой Грегором Менделем.

Приложение охватывает как дигибридное и моногибридное скрещивания по классическим законам Грегора Менделя, так и закономерности множественного аллелизма в его практическом применении, связанном с определением возможных групп крови потомков. Это делает разработанное приложение универсальным инструментом для научного моделирования структур экспериментальной генетики и изучения метода генетического анализа Грегора Менделя.

Приложение разработано с учетом современных требований и стандартов к архитектуре проектирования программного обеспечения и включает в свою внутреннюю структурную организацию ряд рекомендуемых паттернов для возникших в процессе разработки тех или иных проблем:

- паттерн Фасад (скрывает сложную реализацию),
- паттерн Стратегия (реализация стратегий расчета решетки Пеннета для дигибридного и моногибридного скрещивания),
- паттерн Мост (организация вывода решетки Пеннета для разных платформ),
- паттерн Адаптер (позволяет ввести в систему независимый класс для реализации множественного аллелизма на примере групп крови человека),
- паттерн Информационный эксперт (реализует расчету результатов скрещивания: процентное соотношение потомства от скрещивания по генотипу и фенотипу)..

Диаграмма классов системы в упрощенном виде представлена на Рисунке 1. Построение структуры приложения на основе концепции применения паттернов проектирования позволило эффективно организовать проектные решения и алгоритмы, сделать код удобочитаемым для стороннего пользователя и удовлетворяющим современным стандартам

программирования объектно-ориентированных приложений, а так же упростило реструктуризацию системы и возможность ее будущей модификации и расширения.

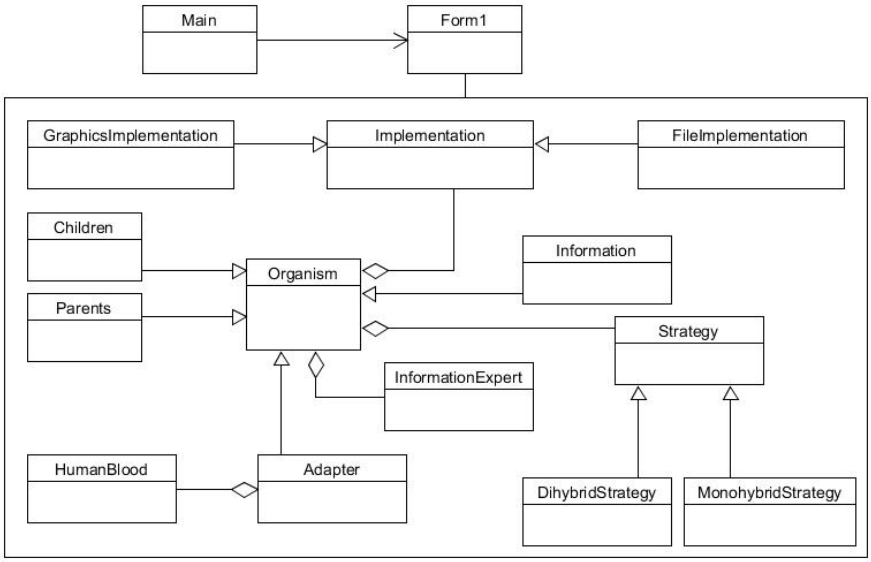


Рисунок 1 – Диаграмма классов системы

Заключение

Таким образом автором данных тезисов представлено разработанное приложение для работы со структурами менделевских скрещиваний и множественным аллелизмом в классической генетике. Приложение помогает предоставить комфортные условия для проведения соответствующих генетических экспериментов и получения достоверных удовлетворительных результатов для их последующего анализа.

Список литературы

1. Законы Менделя [Электронный ресурс] // VseProGeny. - Режим доступа: http://vse-pro-geny.ru/ru_zakony-mendelya.html
2. Основные понятия генетики [Электронный ресурс] // Pidruchniki. – режим доступа: http://pidruchniki.com/1137012263235/psihologiya/osnovnye_ponyatiya_genetiki
3. Решётка Пеннета [Электронный ресурс] // Wikipedia. – Режим доступа: https://ru.wikipedia.org/wiki/Решётка_Пеннета
4. Второй закон Менделя (Закон расщепления) [Электронный ресурс] // Biologylife. – Режим доступа: biologylife.ru/gibridologicheskij/vtoroj-zakon-mendelja-zakon-rasshheplenija.html
5. Первый закон Менделя. Типы взаимодействия аллельных генов [Электронный ресурс] // Foxford. – Режим доступа: https://foxford.ru/wiki/biologiya/pervyy-zakon-mendelya-tipy-vzaimodeystviya-allelnyh-genov
6. Мендель Грегор [Электронный ресурс] // Piplz. – Режим доступа: http://www.piplz.ru/page-id-210.html
7. Множественный аллелизм [Электронный ресурс] // Activestudy. – Режим доступа: http://www.activestudy.info/mnozhestvennyj-allelizm/
8. Множественный аллелизм. Генетика групп крови человека [Электронный ресурс] // Biomed. – Режим доступа: http://biomed.szgmu.ru/SZGMU_SITE/M_Genetics/Multiple_allelism_Genetics_of_human_blood_groups.html
9. Вклад Менделя в развитие экспериментальной генетики [Электронный ресурс] // ReferatiBesplatno. – Режим доступа: http://referati-besplatno.ru/vklad-mendelya-v-razvitie-eksperimentalnoj-genetiki

ИСПОЛЬЗОВАНИЕ ИГРОВОГО ДВИЖКА UNITY В РАЗРАБОТКЕ КОМПЬЮТЕРНЫХ ИГР

Волкова Е.К. – студент
Алтайский государственный технический университет (г. Барнаул)

Разработчик в сложном секторе игровой индустрии с ее уникальными идеями, алгоритмами и инструментами должен проявлять гибкость в использовании функциональных возможностей программного обеспечения, спецификациях, проектировании и управлении, а так же тонко сочетать креативность и технологии многодисциплинарным образом. Возможно вы, как и я, представляли себя в такой многообещающей роли и когда-нибудь, хоть и мысленно, но думали о разработке видеоигры, однако программирование таковых упирается в значительное число препятствий, особенно если речь идет о работе с трехмерным пространством. Сейчас практически никто не пишет серьезные игры на «чистом» Си. Да, конечно, если вы не обременены фантазией и ведете разработку очередного клона шахмат или "2048", то вам хватит и такого развития событий. Но как же обстоит дело с теми случаями, когда вы хотите реализовать куда более сложные проекты? Открытый собственный огромный мир, искусственный интеллект, множество персонажей со своими характеристиками, текстуры и шейдеры, окружающая природа - вряд ли будет целесообразно разрабатывать их в обычной среде. Это, как минимум, трата времени программистов, а для издателя еще и потеря денег, срыв сроков релиза и повод для падения в глазах конкурентов. И это только вершина айсберга, под которой скрывается много тонкостей, сюрпризов и интересных задач разработки видеоигр.

Для того, чтобы упростить жизнь разработчика и приблизить его к мечте разработки игр, существуют среды проектирования игр, так называемые конструкторы, или игровые движки (game engine), которые стали незаменимой технологией в игровом секторе. От графических фреймворков они переросли в отдельную индустрию. Часто получается так, что отделить собственно игру от игрового движка довольно сложно, настолько расплывчатая граница между ними. Крупные игры от известных мировых издателей имеют свои собственные узкоспециализированные игровые движки, которые разрабатываются специально для них, а затем претерпевают изменения: например, модернизируются для новой части продукта. Поэтому разработка игр AAA-класса происходит обычно по схеме: сперва разрабатывается движок для нее, а затем идет разработка собственно игры.

Если смотреть списки популярных (узкоспециализированные не в счет) игровых движков, то возглавлять их практически всегда будет Unity от компании Unity Technologies. И это не случайно, Unity - лидер индустрии, это профессиональный игровой движок, который по праву является одним из самых мощных и популярных, обладая при этом достойным сообществом. Если появляется какая-то новинка в игровой индустрии - будь то технология, концепция или библиотека - то есть высокая доля вероятности, что в скором времени она появится в Unity. Так, например, VR (Virtual Reality) - технология виртуальной реальности, начавшая массовое распространение не так давно - уже доступна в Unity. Сам движок написан на языках C/C++ и еще некоторых других, однако исходный код системы при этом закрыт, то есть модифицировать те или иные внутренние структуры движка под свои нужды невозможно, однако этот недостаток сполна компенсируется достоинствами инструмента, к тому же его свободным распространением, если разрабатываемый вами продукт удовлетворяет правилам бесплатного пользования (которые с каждой версией продукта только упрощаются), установленным компанией Unity Technologies. Unity - это не только движок для инди-проектов независимых студий или индивидуальных разработчиков, как может показаться сначала, его используют такие мировые издатели, как Ubisoft, Blizzard, EA и другие.

Ко всему прочему, игровой движок Unity достаточно прост в сравнении с конкурентами

, и может использоваться как конструктор для разработки игр не только программистами, но и другими членами команды разработчиков, вовлеченными в процесс создания продукта, но выполняющими другие функции, взять хотя бы художников, не говоря уже о геймдизайнерах, левелдизайнерах, моделлерах и других. Так, например, аниматором вполне может быть самостоятельно освоено первоначальное создание анимации в игре.

Метод разработки

Автором представляемого материала разрабатывается компьютерная игра для платформ Windows. Чтобы разрабатываемый проект удовлетворял всем требованиям к современной компьютерной игре, прибегаем как раз к игровому движку Unity и используем его основные концепции в своей разработке. Unity 5 поддерживает несколько языков написания скриптов - это специальная модификация JavaScript и C# (третий язык - Boo, диалект Python, - в последней версии был вырезан). Для описываемой разработки был выбран язык программирования C#, так как у него, во-первых, наиболее мощный функционал, а во-вторых, большое сообщество и уже много готовых решений тех или иных проблем, возникающих в процессе разработки.

А проблем в проектировании видеоигр достаточно, так как это далеко не только кодирование, математика, физика и техническая документация. Нам придется самостоятельно охватить все аспекты геймдизайна: это анимация, изобразительное искусство, архитектура мира игры, брейнмастеринг (генерирование идей), звуковое оформление, дизайн и история игрового мира [2]. Так что, помимо программирования, за нами в разрабатываемой компьютерной игре лежит так же концепция и сюжетная линия игры, проработка мира, проектирование интерфейса и даже в какой-то степени графический дизайн.

Постановка задачи

В основу проектируемой системы положена идея игрового жанра RPG (Role-Playing Game) с элементами экшн-сеттинга: суть ее заключается в управлении игроком уникальным персонажем (или персонажами) - главным героем игры - с определенным набором численных характеристик и умений. Персонаж оснащен инвентарем, предметы которого воздействуют на его характеристики в лучшую или худшую сторону, элементы инвентаря покупаются в игровом мире или предоставляются в качестве награды за задания (квесты). Помимо основного персонажа в игре есть и другие герои, действия которых управляются прописанным искусственным интеллектом, большая часть этих персонажей враждебна по отношению к главному герою игры. Данный игровой жанр предоставляет больше всего возможностей для реализации как идей геймдизайна, так и механик программирования разной сложности.

Планируется использование в разработке как 2D, так и 3D составляющих игрового контента, что позволит не только реализовать разнообразные игровые сцены и подходы к игровому миру, но и покажет больше функциональности и достоинств Unity Game Engine и в принципе роли движков в искусстве видеоигр.

Основные концепции Unity, используемые автором в разработке

В разработке планируется применять (а некоторые уже реализованы) ряд концепций, предоставляемых средствами среды для геймдевелопмента. Пусть перечислены далеко не все концепции, но уже они наглядно демонстрируют все возможности этой сферы проектирования.

Помимо перечисленных ниже концепций Unity в проекте используются и более атомарные, фундаментальные: это игровые объекты (GameObjects - персонажи, элементы

ландшафта или, проще говоря, это практически каждый элемент игрового мира); сцены (Scenes); компоненты (это скрипты, изображения, элементы физики и поведения, которые привязываются к объектам); многоуровневые связи наследования GameObjects; материалы, шейдеры и текстуры процесса рендеринга; анимация; спрайты; импортируемые ассеты (файлы, например, 3D модели) и другие концепции [3]. Без обращения к этим фундаментальным элементам в принципе не построить игровую разработку.

Интерфейс пользователя

Интерфейсу уделен отдельный этап разработки, проходящий в три шага для получения качественного красивого результата, а не просто беспорядочного набора первых пришедших в голову кнопок и слайдеров: первичное проектирование интерфейса, собственно проектирование интерфейса и, наконец, реализация интерфейса с использованием возможностей GUI Unity 5, который претерпел значительную модернизацию в сравнении со своими более ранними версиями. Раньше разработка была ахиллесовой пятой движка, так как приходилось прописывать его полностью программно в скриптах, причем код в результате работал медленно и неэффективно [1]. Сейчас же подход к разработке интерфейса полностью модифицирован и не отнимает много времени, которое лучше уделить проработке самой механики игры.

Камера

Камера захватывает и отображает вид сцены для пользователя: это могут быть как классические виды от третьего лица или от первого, так и вид для 2D сцены. Вид камеры и ее перемещение по сцене обычно задают при помощи векторов, лучей и численных методов для плавности самого перемещения

Ладшафт

Ландшафт создается из прямоугольной плоскости, которая далее искривляется вручную для создания реалистичного окружения игры. На самом деле работа с ландшафтом вручную - не самый лучший вариант, по крайней мере, для игр с большой площадью игрового мира. Обычно используют карты высот - изображения, пиксели которых совпадают с пикселями ландшафта. На карте высот чем светлее участок, тем он выше. Карту высот можно создать при помощи, например, шумов Перлина - это математический алгоритм по генерированию процедурной текстуры псевдослучайным методом. Но это для случайного ландшафта, гораздо реалистичнее и красивее - создание карты высот посредством графического редактора и с использованием тех же спутниковых карт, например. Для повышения производительности Unity использует функцию Occlusion Culling: она отключает отрисовку объектов, которые не видны камерой или которые заслонены другими объектами на сцене. Для двумерных проектов ландшафт заменяет фоновое изображение, по которому перемещается камера, отображая игроку его текущее местоположение.

Источник света

Unity предоставляет несколько вариантов источников света. Чтобы реализовать Солнце, можно использовать не требовательный к ресурсам Direction Light (бесконечно далекий источник, рисунок 1), который представляет собой бесконечное множество параллельных друг другу лучей. Каждая сцена в Unity содержит источник направленного света по умолчанию. Положение солнца на карте позволяет имитировать освещение в зависимости от времени суток: сверху - дневное освещение, сбоку - сумеречное.

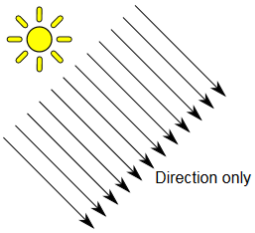


Рисунок 1 – Испускание света Direction Light

Для большей реалистичности источника света к нему привязывается блик, представляющий собой текстуру, содержащую несколько различных элементов. Эти элементы блика распределены вдоль линии, которая вычисляется с помощью сравнения положения GameObject (с компонентом блика) и центром экрана. Все элементы каждого блика должны находиться на одной и той же текстуре для увеличения производительности. Распределение элементов блика по текстуре выполняется различными вариантами. На рисунке 2 показано распределение текстур для больших солнечных бликов, один элемент имеет высокое качество (текстура 0 на рисунке 2), в отличие от остальных четырех [4].

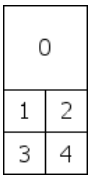


Рисунок 2 – Вариант распределения текстур блика

Тени

Тени реализуются по технологии карт теней (shadow mapping) с использованием еще одной аналогичной технологии - карт глубины (depth mapping) [6]. Последняя нужна для хранения расстояний для каждой освещаемой поверхности, что помогает определить то, какая из поверхностей заслонена другой. Для создания карты глубины камера сцены осуществляет отрисовку от позиции источника света внутрь камеры, после чего позиция каждого пикселя для вида из камеры преобразуется в позицию в световом пространстве. Остается только определить дистанцию, используя соответствующий пиксель, полученный после рендеринга карты.

Skybox

Чтобы карта имела эффект законченности мира, задействуется панорамный вид - так называемый Skybox, посредством которого моделируется небо, это простой и лаконичный способ добавить реализма к сцене, который к тому же совсем не требователен к ресурсам. Skybox представляет собой куб, вдоль шести граней которого натягиваются текстуры. Полученное панорамное текстурное изображение рисуется позади всех объектов сцены и в итоге дает непрерывный окружающий образ, который пользователь рассматривает "изнутри коробки", при этом края текстур выглядят так, что эффекта самого нахождения внутри коробки нет [5].

Заключение

На сегодняшний день игровые движки предоставляют наилучшие платформы для разработки видеоигр на разный платформах: консолях, персональных компьютерах, планшетах и мобильных устройствах. Продукт от Unity Technologies, представляющий собой интегрированную среду разработки игр, занимает среди своих конкурентов в этой сфере лидирующие позиции. Unity Game Engine позволяет в полной мере реализовать механики,

технологии и эстетику в разрабатываемой игре.

Поэтому именно он применяется автором в реализации проекта - игры жанра RPG - достаточно сложного и требующего не одного этапа проектирования и проработки каждого шага: начиная от персонажа, управляемого игроком, мира и искусственного интеллекта неиграбельных персонажей и заканчивая внутриигровым интерфейсом. В дальнейшем так же систему вполне легко расширять, например, добавлением серверной части, позволяющей осуществлять игру по сети.

Список литературы

1. Учебник по новому GUI в Unity. Часть 1 [Электронный ресурс] // Websketches. - Режим доступа: http://websketches.ru/blog/uhebnik-po-novomu-gui-v-unity-1
2. Schell J. The Art of Game Design. - Burlington, USA, 2008. - P. 2–5.
3. Основные концепции Unity [Электронный ресурс] // IT-школа SAMSUNG. – Режим доступа: http://docplayer.ru/38565094-Osnovnye-koncepcii-unity3d-8-noyabrya-2016.html
4. Flare [Электронный ресурс] // Unity3d. – Режим доступа: https://docs.unity3d.com/ru/530/Manual/class-Flare.html
5. Использование skyboxes Unity3d. Шейдеры и свет [Электронный ресурс] // MyUnity3d. – Режим доступа: http://my-unity3d.blogspot.ru/2015/02/skyboxes-unity3d.html
6. Справка по освещению [Электронный ресурс] // Графика в Unity 3D. – Режим доступа: https://sites.google.com/site/rusewyl/grafika/spravka-po-graficeskim-vozmoznostam/spravka-po-osveseniu

ОБЗОР СУЩЕСТВУЮЩИХ АРХИТЕКТУР СВЕРТОЧНЫХ НЕРОЙННЫХ СЕТЕЙ

Шахов Д.Е. – магистрант
Алтайский государственный технический университет (г. Барнаул)

Идея сверточных сетей берет свою начало из работ по нейробиологии. В классической работе Д. Хьюбела и Т. Визела [1,2] говорится, что зрительная кора головного мозга представляет собой сложный комплекс клеток («простых» и «сложных»), каждая из которых чувствительна только к ограниченному участку поля зрения. Такие участки, иначе называемые рецептивными полями, стыкуются вместе, обеспечивая перекрытие всего поля зрения. «Простые» клетки при этом выполняют роль локальных фильтров входных данных, реагируя на присутствие в собственном рецептивном поле некоторых примитивных структур, таких как края и границы. Было обнаружено также существование так называемых «сложных клеток», имеющих более широкие рецептивные поля, и демонстрировавших инвариантность по отношению к точному расположению объекта в поле зрения. Достигается это тем, что «сложные» клетки относятся к следующему уровню обработки, получая на вход результат активации «простых» клеток [11].

Первой настоящей реализацией сверточной нейросети можно назвать Neocognitron Кунихико Фукусимы, появившийся в 1979-1980 годах [3]. Стоит отметить, что Фукусима не использовался градиентный спуск и обучение с учителем, со временем о его работах прочно забыли. В современной форме сверточные сети появились в работах группы Яна ЛеКуна в конце 1980-х годов [4, 5], с тех пор они успешно применяются для распознавания изображений и в других задачах [12, 13].

Первой предложенной сверточной нейронной сетью была модель Яна ЛеКуна – LeNet. Это была первая модель, состоящая из двух чередующихся слоев свертки и субдискретизации, а также трех полносвязных слоев. Данная архитектура с параметрами, приведенными на рисунке 1, считается классической [4].

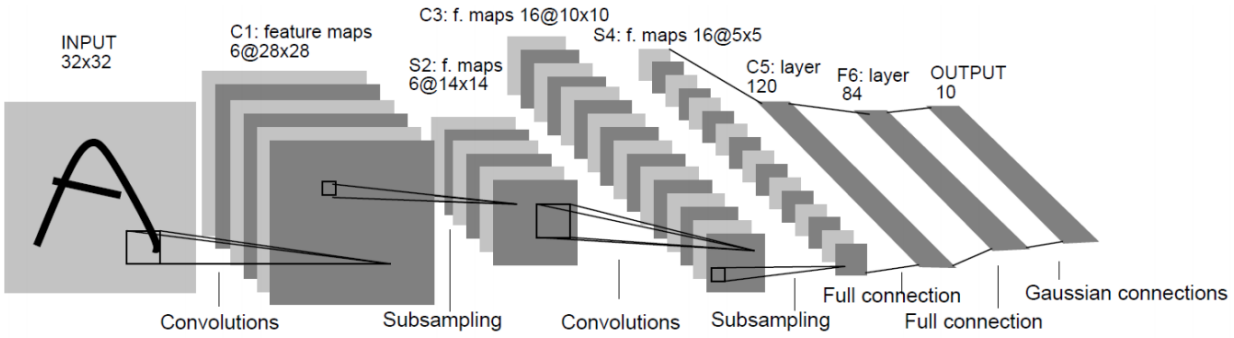


Рисунок 1 – Классическая архитектура сверточной нейронной сети

Следующей значительной сверточной нейронной сетью после LeNet была сеть AlexNet, предложенная Алексом Крижевским [6]. Данная сеть очень похожа на сеть LeNet, однако отличается от неё более сложной архитектурой. Сеть имеет всего лишь один сверточный слой и несколько слоев субдискретизации по принципу выбора максимального значения. Для улучшения работы сети использовались новейшие на тот момент техники: функция активации ReLU (Rectifier Linear Unit) и отсев (Dropout [10]). Применение ReLU позволяет решить проблему затухания градиентов, свойственную другим функциям активации (сигмоида, гиперболический тангенс и другие). Dropout выполняет роль регуляризатора, не позволяя сети переобучаться. Сверточная нейронная сеть AlexNet в 2012 году стала победителем в конкурсе ImageNet. Схема архитектуры сети представлена на рисунке 2.

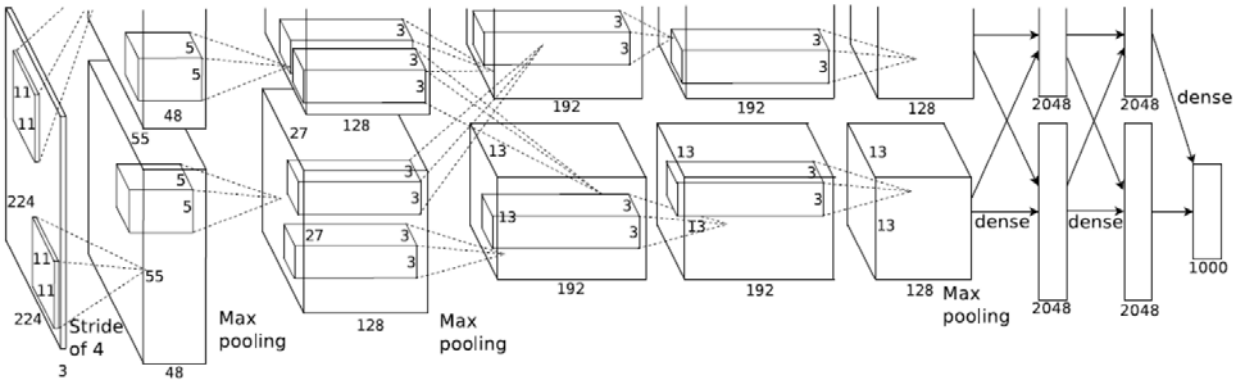


Рисунок 2 – Архитектура сети AlexNet

Следующим значительным вкладом в развитие построения высокоэффективных архитектур сверточных нейронных сетей стала сеть ZF Net, созданная Метью Цейлером и Робом Фергюсом в 2013 году [7]. Данная сеть представляет собой модификацию сети AlexNet, основная особенность которой заключается в более удачном наборе параметров сети: увеличение размеров внутренних сверточных слоев сети, а также уменьшение размеров смещения и размеров фильтров в первом сверточном слое. Архитектура сети ZF Net представлена на рисунке 3.

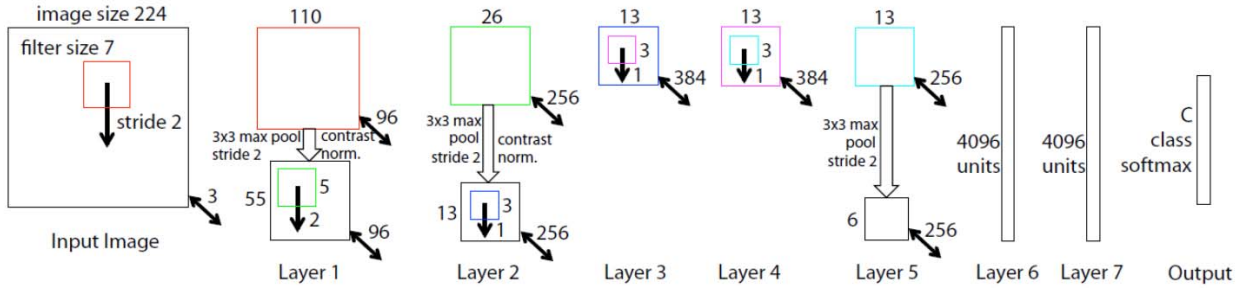


Рисунок 3 – Архитектура сети ZF Net

в 2014 году компания Google представила свою архитектуру сверточной нейронной сети GoogLeNet [8].

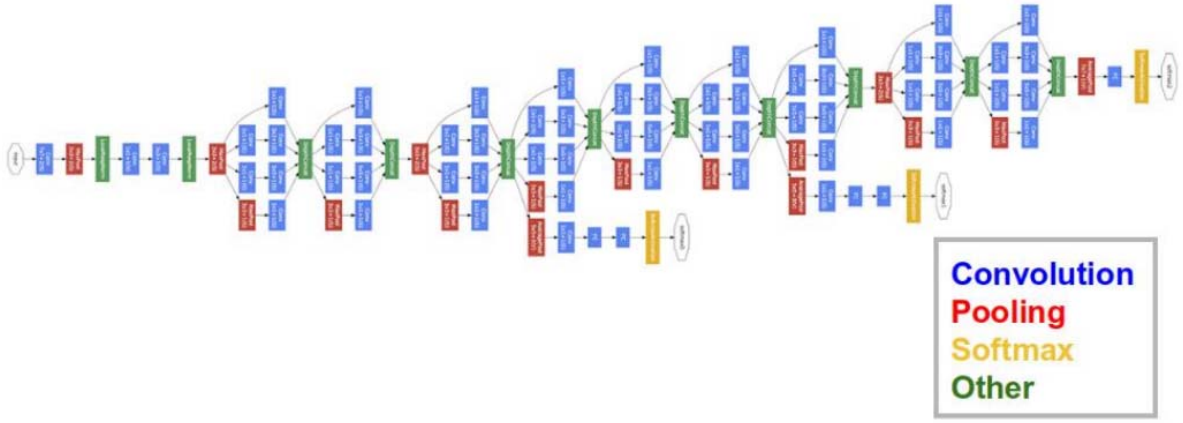


Рисунок 4 - Архитектура сети GoogLeNet

Данная сверточная сеть является очень глубокой – до 22 слоев. Но, несмотря на это, имеет в 10 раз меньше параметров, чем сеть AlexNet, что положительно сказывается на производительности и расходовании памяти. Число параметров удалось уменьшить при помощи замены полносвязного слоя в конце сети на слой субдискретизации, реализованный по принципу выбора среднего значения (Average Pooling). Сверточная сеть GoogLeNet стала победителем конкурса ILSVRC 2014. Архитектура данной сети представлена на рисунке 4.

Одной из наиболее современных сверточных нейронных сетей на сегодняшний день является сеть ResNet (Residual Network), ставшая победителем на конкурсе ILSVRC 2015 [9]. Архитектура данной сети, разработаной в Microsoft Research, предполагает большое количество сверточных слоев, содержащих большое количество (до 512) фильтров малого размера (3 * 3). Глубина сети может достигать 152 слоев. На примере данной сети было установлено, что сверточная нейросеть может использовать только сверточные слои и при этом качество распознавания значительно улучшится при увеличении глубины сети. Сеть ResNet является одной из наиболее эффективных сверточных нейронных сетей на сегодняшний день.

Список литературы

1. Hubel D.H. Brain and visual perception / D.H. Hubel, T.N. Wiesel. - ISBN13, 2005. - P. 36–46.
2. Hubel D.H. Eye, brain, and vision / D.H. Hubel. - New York: Scientific American Library, 1988. – P .85–87.
3. Fukushima K. Neocognitron: A Self-Organizing Neural Network for a Mechanism of Pattern Recognition Unaffected by Shift in Position // Biological Cybernetics. – 1980. - vol. 36. - no. 4. – P. 658 - 665.
4. LeCun, Y. Backpropagation applied to handwritten zip code recognition / Y. LeCun // Neural computation. - 1989. - no. 4. - P. 541–551.
5. Gradient-Based Learning Applied to Document Recognition / Y. LeCun et al. // Proc. IEEE. – 1998. - vol.86. - no. 11. – P. 2278-2324.
6. Krizhevsky A. Imagenet classification with deep convolutional neural networks / A. Krizhevsky, I. Sutskever, G.E. Hinton // Advances in neural information processing systems. - 2012. - P. 1097–1105.
7. Visualizing and Understanding Convolutional Networks [Электронный ресурс] / Matthew D

Zeiler, Rob Fergus // arXiv, 2013. - Режим доступа: http://arxiv.org/abs/1311.2901.
8. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich. Going Deeper with Convolutions. Proceedings of IEEE Conference Computer Vision and Pattern Recognition (CVPR 2015). June 7-12. - 2015. - P. 1-12.
9. K. He, X. Zhang, S. Ren, J. Sun. Deep Residual Learning for Image Recognition. Microsoft Research. Proceedings of Computer Vision and Pattern Recognition Conference (CVPR 2015), Boston, June 8-10. - 2015. - P. 770-778.
10. Krizhevsky, A. Dropout: a simple way to prevent neural networks from overfitting / A. Krizhevsky, G. Hinton, N. Srivastava, I. Sutskever, R. Salakhutdinov // The Journal of Machine Learning Research, 2014, vol. 15, no.1. – P. 1929–1958.
11. Николенко С. Глубокое обучение. Погружение в мир нейронных сетей. / Николенко С., Кадурин А., Архангельская Е. – СПб.: Питер, 2018. – 480 с.: ил.
12. Recent Advances in Convolutional Neural Networks / J. Gu et al. // arXiv, 2015. [Электронный ресурс]. - Режим доступа: http://arxiv.org/abs/1512.07108
13. A Taxonomy of Deep Convolutional Neural Nets for Computer Vision [Электронный ресурс] / S. Srinivas et al. // arXiv, 2016. - Режим доступа: http://arxiv.org/abs/1601.06615

РАЗРАБОТКА ПРОГРАММНОГО КОМПЛЕКСА ДЛЯ РАСПОЗНАВАНИЯ ЧАСТИЦ СЫПУЧИХ ПРОДУКТОВ С ПОМОЩЬЮ СВЕРТОЧНЫХ НЕЙРОННЫХ СЕТЕЙ

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

Задача распознавания изображений востребована в таких отраслях как робототехника, медицина, химическая промышленность, аэрофотосъемка, автомобилестроение, информационный поиск, мониторинг и анализ визуальных данных, исследования искусственного интеллекта.

Интенсивные исследования в этой области имеют многолетнюю историю и связаны с работами Д. Хьюбела и Т. Визела, [1-3], Т. Кохонена [4], М. Турка и А. Петланда [5], Д. Хинтона [6,7], Я. ЛеКуна [8,9] и других. За последнее время существенный прогресс в распознавании изображений был достигнут с появлением методов снижения размерности [10], сверточных нейронных сетей [8,11].

В последнее десятилетие начали активно развиваться компании, занимающиеся фотосепарацией (сортировкой по цвету) [12]. Для развития и обеспечения конкурентоспособности компании-производители фотосепараторов следят за достижениями в области распознавания изображений, в особенности за сверточными нейронными сетями.

Перед применением любой новой технологии в производстве необходимо оценить преимущества, получаемые при ее использовании, и проблемы, возникающие при внедрении. Для оценки преимуществ и проблем применения сверточных нейронных сетей необходимо разработать программный комплекс, использующий их для распознавания частиц сыпучих продуктов. При проектировании и разработке программного комплекса нужно обратить внимание на то, что главными показателями оценки качества работы фотосепаратора являются точность и производительность сортировки, а также широта спектра обрабатываемых продуктов.

Сверточные нейронные сети (convolution neural networks, CNN) – это весьма широкий класс архитектур, основная идея которых состоит в том, чтобы переиспользовать одни и те же части нейронной сети для работы с разными маленькими, локальными участками входов [13]. Нейронная архитектура сверточной сети известна довольно давно, и для нее нашлось

множество различных применений, но основным приложением является обработка изображений.

Работа сверточной нейронной сети обычно интерпретируется как переход от конкретных особенностей изображения к более абстрактным деталям, и далее к ещё более абстрактным деталям вплоть до выделения понятий высокого уровня. При этом сеть самонастраивается и вырабатывает необходимую иерархию абстрактных признаков, фильтруя маловажные детали и выделяя существенное.

Фактически, «признаки», вырабатываемые сложной сетью, малопонятны и трудны для интерпретации настолько, что в практических системах не особенно рекомендуется пытаться понять содержания этих признаков или пытаться их «подправить», вместо этого рекомендуется усовершенствовать саму структуру и архитектуру сети, чтобы получить лучшие результаты. Так, игнорирование системой каких-то существенных явлений может говорить о том, что-либо не хватает данных для обучения, либо структура сети обладает недостатками, и система не может выработать эффективных признаков для данных явлений.

Преимущества использования сверточных нейронных сетей для распознавания частиц сыпучих продуктов:

- Выделение отличий между годными и негодными частицами конкретного продукта и его загрязнителями происходит при обучении сверточной нейронной сети. Данную способность можно применить для сортировки продуктов. При обучении сети будут подбираться эффективные параметры распознавания изображений, за счет этого можно улучшить качество сортировки фотосепаратора. Также это должно дать возможность фотосепарации продуктов, для которых сложно выделить признаки, отличающие годные частицы от засорителей, используя простые метрики объекта (ширина, высота, цвет и другие характеристики).

Проблемы, которые необходимо решить при использовании сверточных нейронных сетей для распознавания частиц сыпучих продуктов:

- Для обучения нейросети требуется большой объем размеченных данных. Для решения этой проблемы можно использовать синтетические изображения. Для их создания необходим программный комплекс, генерирующий размеченные для обучения данные на основе 3D-моделей продукта и его возможных загрязнителей. Для создания 3D-моделей можно использовать программные продукты, предназначенные для 3D-моделирования, например, Blender [14]. Создание качественных моделей при помощи 3D-редакторов - трудоемкая задача, более быстрым, но в тоже время и дорогим способом создания модели является моделирование с использованием 3D-сканера. После обучения модели на синтетических данных, необходимо провести ее дообучение на реальных данных;
- Сложность изменения существующих параметров сортировки определенного продукта. Продукт при хранении меняет цвет, форму и т.д. Изменения потребуют дообучение модели к новым особенностям продукта, а для этого нужны размеченные данные с соответствующими особенностями. В идеальном же случае модель должна обучаться все время работы фотосепаратора, т.е. производить обучение с подкреплением [13]. При таком методе обучения модель работает в окружении, о которой ничего не знает, но имеет возможность вносить в себя изменения. Для обеспечения связи между окружением и моделью вводится функция вознаграждения, которая сигнализирует модели о позитивных или негативных последствиях внесенных изменений. Сложность применения этого метода заключается в определении функции вознаграждения;
- Необходимость подбора топологии сверточной нейронной сети (LeNet [15], AlexNet [7], ZF Net [16], GoogLeNet [17], VGGNet [18], ResNet [19] и другие), при которой обученная и оптимизированная нейросеть сможет обеспечить высокую скорость и точность ответа при работе на GPU. Большинство параметров нейронной сети сосредоточено в сверточных слоях. Рассмотрим некоторые техники оптимизации сверточных слоев. Одной из часто

используемых техник является прореживание (pruning) нейронной сети. Основная идея прореживания заключается в том, что те связи (веса) между нейронами или сами нейроны сети, которые оказывают малое влияние на ошибку аппроксимации, можно исключить из модели без значительного ухудшения качества аппроксимации. Другая техника основана на идее квантизации весов, которая заключается в том, что веса могут принимать значения из заранее заданного набора чисел. Подробнее данные техники, а также их комбинирование описано в [20-23]. Для увеличения коэффициента сжатия описанные техники могут быть объединены с кодированием Хаффмана, как это описано в работе [20].

Список литературы

1. Hubel, D.H. Brain and visual perception / D.H. Hubel, T.N. Wiesel. - ISBN13, 2005. - P. 36–46.
2. Hubel, D.H. Eye, brain, and vision / D.H. Hubel. - New York: Scientific American Library, 1988. - P. 85–87.
3. Hubel, D.H. Receptive fields and functional architecture of monkey striate cortex / D.H. Hubel, T.N. Wiesel // The Journal of physiology. - 1968. - no. 195(1). - p. 215–243.
4. Kohonen, T. Self-organization and associative memory / T. Kohonen // Springer-Verlag Berlin Heidelberg New York. - 1988. - no. 8(1). - P. 13–27.
5. Turk M.A. Face recognition using eigenfaces / M. A. Turk, A. P. Pentland // Computer Vision and Pattern Recognition. - 1991. - no. Proceedings CVPR'91., IEEE Computer Society Conference. - P. 586–591.
6. Hinton G.E. Transforming auto-encoders / G.E. Hinton, A. Krizhevsky, S. D. Wang // Artificial Neural Networks and Machine Learning–ICANN 2011. - 2014. - P. 44–51.
7. Krizhevsky A. Imagenet classification with deep convolutional neural networks / A. Krizhevsky, I. Sutskever, G.E. Hinton // Advances in neural information processing systems. - 2012. - P. 1097–1105.
8. LeCun Y. Backpropagation applied to handwritten zip code recognition / Y. LeCun // Neural computation. - 1989. - no. 4. - P. 541–551.
9. LeCun Y. Comparison of learning algorithms for handwritten digit recognition / Y. LeCun // International conference on artificial neural networks. - 1995. - no. 60. - P. 111–115.
10. Hinton G.E. Reducing the dimensionality of data with neural networks / G.E. Hinton, R.R. Salakhutdinov // Science. - 2006. - no. 313(5786). - P. 504–507.
11. Szegedy C. Going deeper with convolutions / C. Szegedy // arXiv. - 2014. - no. 1409.4842.
12. Главная страница сайта ООО «СиСорт» [Электронный ресурс]. - Режим доступа: http://csort.ru/
13. Николенко С. Глубокое обучение. Погружение в мир нейронных сетей. / Николенко С., Кадурин А., Архангельская Е. - СПб.: Питер, 2018. – 480 с.: ил.
14. Home of the Blender project [Электронный ресурс]. - Режим доступа: https://www.blender.org/
15. Gradient-Based Learning Applied to Document Recognition / Y. LeCun et al. // Proc. IEEE, 1998, vol. 86, no. 11. – P. 2278-2324.
16. Visualizing and Understanding Convolutional Networks [Электронный ресурс]/ Matthew D Zeiler, Rob Fergus // arXiv, 2013. - Режим доступа: http://arxiv.org/abs/1311.2901.
17. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich. Going Deeper with Convolutions. Proceedings of IEEE Conference Computer Vision and Pattern Recognition (CVPR 2015). June 7-12. - 2015 - P. 1 - 12.

18. K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-scale Image Recognition. Proceedings of 3rd International Conference on Learning Representations (ICLR2015), Hilton San Diego Resort & Spa, May 7-9. - 2015. – P. 1 - 14.
19. K. He, X. Zhang, S. Ren, J. Sun. Deep Residual Learning for Image Recognition. Microsoft Research. Proceedings of Computer Vision and Pattern Recognition Conference (CVPR 2015), Boston, June 8-10. - 2015. - P. 770 - 778.
20. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding [Электронный ресурс] / Han Song, Mao Huizi, Dally William J. // arXiv, 2015. - Режим доступа: http://arxiv.org/abs/1510.00149
21. Learning both weights and connections for efficient neural network / Song Han, Jeff Pool, John Tran, William Dally // Advances in Neural Information Processing Systems. - 2015. - P. 1135–1143.
22. PerforatedCNNs: Acceleration through elimination of redundant convolutions / Mikhail Figurnov, Aizhan Ibraimova, Dmitry P Vetrov, Pushmeet Kohli // Advances in Neural Information Processing Systems. - 2016. - P. 947–955.
23. Automated Pruning for Deep Neural Network Compression [Электронный ресурс] / A. Rozza et al. // arXiv, 2017. - Режим доступа: http://arxiv.org/abs/1712.01721

КЛАССИФИКАЦИЯ ИЗОБРАЖЕНИЙ И АНАЛИЗ ТОНАЛЬНОСТИ ТЕКСТОВ С ПОМОЩЬЮ НЕЙРОННЫХ СЕТЕЙ

Киреков С.А. - студент, Крайванова В.А. - к.ф.-м.н., доцент
Алтайский Государственный Технический Университет (г. Барнаул)

В современном мире машинное обучение занимает занимает лидирующие позиции в области computer science. Отныне оно не является лишь предметом научных исследований: многие крупные компании применяют алгоритмы машинного обучения для продвижения своего бизнеса. Например, Google анализирует запросы пользователей и на их основании показывает рекламу тех товаров, которые могут их заинтересовать. Также машинное обучение (в частности, искусственные нейронные сети) применяются и в пользовательских приложениях. Среди новых разработок можно выделить “Алису” от Яндекса, которая может отвечать на вопросы, делать запросы в поиске Яндекса и даже играть в некоторые игры.

Целью данной работы является разработка программного комплекса для быстрой постановки экспериментов по классификации различных типов объектов с помощью современных нейронных сетей.

Сами классификаторы и определяемые классы хранятся в базе данных. На данный момент приложения способно классифицировать изображения и анализировать тональность поданного текста. Для проектирования и обучения нейронных сетей был использован язык Python и библиотеки Tensorflow и Keras. Tensorflow - библиотека от Google, Keras - фреймворк для упрощения работы с Tensorflow.

На рисунке 1 представлена архитектура базы данных, в которой хранится информация о доступных классификаторах.

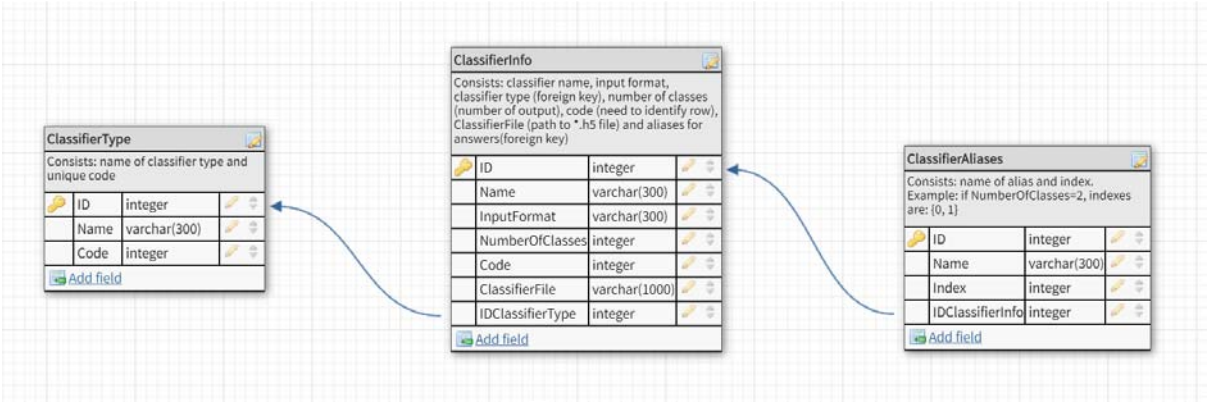


Рисунок 1 - Схема базы данных

В таблице ClassifierType представлены типы возможных классифицируемых объектов (в данном случае “Классификация изображений” и “Анализ тональности текста”). ClassifierInfo хранит информацию о конкретном классификаторе. В ClassifierAliases находится информация о классах, которые данный классификатор распознает. Такая схема позволяет легко добавлять новые типы классифицируемых объектов и новые классификаторы. Архитектура обученной сети, а также веса записываются файл с расширением .h5 с помощью интерфейса библиотеки Keras. Ссылка, где хранится файл на компьютере, записывается в таблицу ClassifierInfo в поле ClassifierFile.

Для классификации изображений была выбрана архитектура *сверточных нейронных сетей* [1]. Классификатор обучен на базе датасета CIFAR-10 (датасет разделен на обучающую и тестовую выборки: 50000 и 10000 изображений соответственно) [2]. На вход подается изображения размера 32x32. На выходе массив из десяти элементов, где единица в массиве, означает класс с соответствующим индексом. Архитектура нейронной сети представлена на рисунке 2.

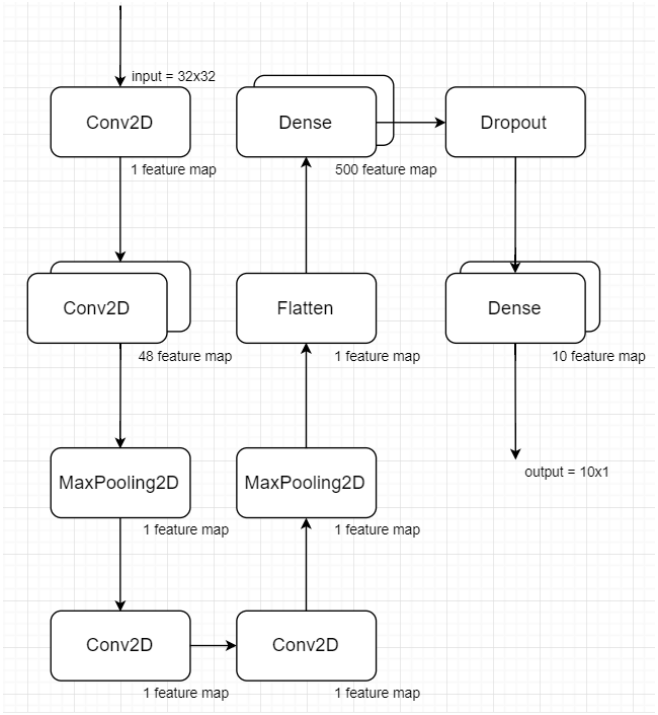


Рисунок 2 - Архитектура нейронной сети для классификации изображений

В сентимент-анализе (анализ тональности текста) решается задача бинарной классификации сообщений с точки зрения его тональности. То есть текст определяется как положительно либо отрицательно окрашенный.

Для определения тональности текста была выбрана рекуррентная LSTM (Long-Short-Term-Memory [3]) нейронная сеть. Обучение происходило на датасете [6] (выборка была разделена на обучающую и тестовую по 1200 и 500 текстов соответственно). Однако подавать текст напрямую на вход нейронной сети нельзя. Сначала нужно преобразовать его в вектор. Для этого можно использовать, например, word2vec [4] от Google. В моей работе мы применили Tokenizer из библиотеки keras, который дает каждому уникальному слову индекс, а затем представляет текст как последовательность этих индексов. На вход нейронной сети подается эта последовательность (массив), на выходе получается массив из двух элементов, где единица в первом означает принадлежность текста к первому классу (негативная окраска), а второй, соответственно, ко второму (положительная окраска). Архитектура нейронной сети представлена на рисунке 3:

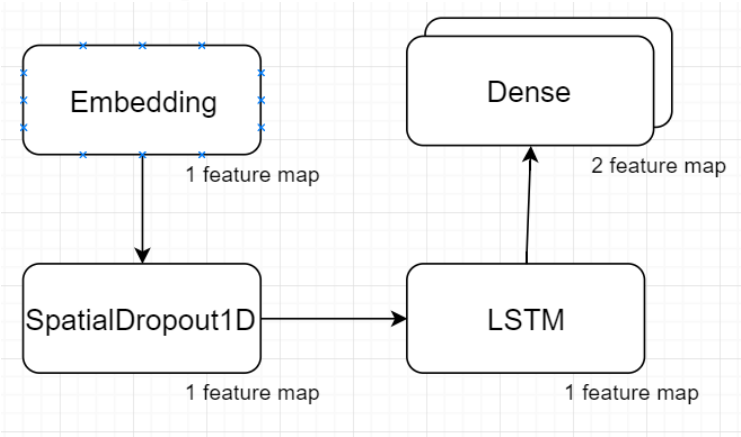


Рисунок 3 - Архитектура нейронной сети для анализа тональности текста

Рассмотрим результаты работы программы. Проверим точность предсказания для изображений. В качестве проверки возьмем 10000 изображений из тестовой выборки датасета CIFAR-10. Результаты представлены на рисунке 5:

```
Средняя величина ошибки (MSE):  0.0308614080233383
Средняя точность:  0.7809999975860119
```

Рисунок 5 - Результат классификации изображения

Как видим, точность классификации довольно высока (почти 80%). В качестве ошибки считается средняя квадратичная ошибка (MSE [5])

Проверим анализ тональности текста. В качестве датасета для проверки возьмем тестовую часть датасета [6] (500 текстов). Результаты представлены на рисунке 6

```
Средняя величина ошибки (Cross Entropy):  0.0012390605251911
Средняя точность:  0.7620000000215059
```

Рисунок 6 — Результат сентимент-анализа

Ошибка считается по методу перекрестной энтропии [7]. Точность равна ~76%.

Выводы

Как можно заметить, протестированные архитектуры нейронных сетей показывают результаты, сравнимые по точности с текущим состоянием области. У изображений они чуть

выше, что, скорее всего, связано с большей обучающей выборкой (50000 изображений против 1200 текстов).

Список литературы

1. Inside Deep Learning: Computer Vision With Convolutional Neural Networks [Электронный ресурс]. - Режим доступа: https://www.kdnuggets.com/2015/04/inside-deep-learning-computer-vision-convolutional-neural-networks.html
2. CIFAR-10 and CIFAR-100 datasets [Электронный ресурс]. - Режим доступа: https://www.cs.toronto.edu/~kriz/cifar.html
3. Long-Short-Term-Memory [Электронный ресурс]. -Режим доступа: http://www.bioinf.jku.at/publications/older/2604.pdf
4. Word2Vec, Doc2Vec & Glove: Neural Word Embeddings For Natural Language Processing [Электронный ресурс]. - Режим доступа: https://deeplearning4j.org/word2vec.html
5. Mean Squared Error [Электронный ресурс]. - Режим доступа: https://en.wikipedia.org/wiki/Mean_squared_error
6. Sentiment Analysis in Russian [Электронный ресурс]. - Режим доступа:https://www.kaggle.com/c/sentiment-analysis-in-russian/data
7. Cross Entropy: [Электронный ресурс]. - Режим доступа: https://en.wikipedia.org/wiki/Cross_entropy#Cross-entropy_minimization
8. Введение в машинное обучение [Электронный ресурс]. - Режим доступа: https://www.coursera.org/learn/vvedenie-mashinnoe-obuchenie
9. Саймон Хайкин. Нейронные Сети. Полный курс. - Издательство: Вильямс, 2016. - 1104 с.

ОБНАРУЖЕНИЕ И ОТСЛЕЖИВАНИЕ ТРАНСПОРТНЫХ СРЕДСТВ В ВИДЕОПОТОКЕ

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

Актуальность

В последнее время стали широко распространены системы компьютерного зрения, к которым в том числе относятся: автоматическая фиксация нарушений правил дорожного движения, анализ автомобильного трафика. Стремительно развивается сфера разработки беспилотных транспортных средств и вспомогательных систем для автомобилей, анализирующих дорожную обстановку. Одним из важнейших вопросов в данных системах является отслеживание других участников движения, в том числе транспортных средств, для дальнейшего анализа их поведения и принятия решений на основе этой информации.

Несмотря на использование в подобных системах большого набора различных датчиков, основным источником информации зачастую является видеопоток со стационарной или закрепленной на автомобиле камеры.

Постановка задачи

В задаче отслеживания транспортных средств можно выделить две отдельные подзадачи: обнаружение транспортных средств на отдельных кадрах видео и связь обнаруженных на различных кадрах областей, относящихся к одному автомобилю.

Интерес представляют наиболее быстрые подходы к решению этих задач, так как зачастую требуется обработка данных в реальном времени. Процент ошибок должен быть минимально возможным, а в некоторых случаях полностью исключен.

Рассмотрим подходы к решению каждой из поставленных задач и проанализируем результаты их применения при разработке системы отслеживания транспортных средств.

Обнаружение транспортных средств

Самым распространенным подходом определения движущихся объектов на видеопотоке со стационарных камер является вычитание фона (background subtraction) – техники, когда передний план отделяется от фона, который незначительно различается между кадрами, что позволяется несложно его определить по серии кадров, а затем отделить с последующих.

В рамках работы был опробован алгоритм BackgroundSubtractorMOG2, к достоинствам которого можно отнести возможность определения теней на изображении и хорошую адаптируемость к изменениям освещенности и погодных условий. На полученном переднем плане находятся контуры, которые являются интересующими нас объектами. Данный метод показал хорошие результаты работы, но в рамках разрабатываемой системы было решено использовать видеопоток с динамической камеры, поэтому от него пришлось отказаться.

Часто встречается подход, основанный на методе Виолы-Джонса (каскады Хаара), который наиболее часто применяется для распознавания лиц. К изображению применяется сканирующее окно различного размера, к каждому положению которого применяется классификатор, выносящий на основе значений признаков вердит, содержит ли выбранная область искомый объект. Классификация основана на поиске областей с перепадами яркости и состоит из нескольких каскадов для быстрого отбрасывания неподходящих областей.

Обучение классификатора требует большого количества времени и изображений, содержащих искомый объект. В работе был опробован обученный вариант из открытого репозитория [1], который обучен на 526 изображениях автомобилей сзади. Несмотря на достаточно неплохие результаты в большинстве случаев, данный классификатор давал большое количество ложноположительных срабатываний, в том числе повторяющихся между соседними кадрами, что значительно затрудняет фильтрацию. Также были замечены проблемы с обнаружением некоторых разновидностей автомобилей, например, микроавтобусов. Поэтому от этого подхода также было решено отказаться.

Есть решения, основанные на методе опорных векторов (SVM), когда каждый объект представляется p-мерным вектором признаков и относится только к одному из двух классов. Во время обучения находится оптимальная гиперплоскость размерности (p – 1), разделяющая эти два класса. Тогда классификация объекта основывается на том, по какую сторону от этой гиперплоскости лежит вектор, описывающий проверяемый объект.

В данных системах в качестве признаков используется гистограмма направленных градиентов и гистограммы всех цветовых каналов. Обученный классификатор также используется со сканирующим окном. В работе данный метод не использовался.

Вариантом, выбранным для применения в системе, является сверточная нейронная сеть YOLOv2 (You only look once) [2]. Особенностью, которой является то, что она применяется один раз к целому изображению. Это позволяет учитывать общий контекст изображения, а также быстро обрабатывать изображение при использовании вычислений на GPU.

Изображение размера 416x416 разбивается сеткой 13x13 на ячейки, каждая из которых отвечает за предсказание 5 прямоугольных областей, вероятно содержащих объект, и уровня уверенности в том, что эти области действительно содержат какой-либо объект. Для каждой найденной области определяется распределение вероятностей по всем возможным классам. Из полученных 845 областей отсекаются те, что имеют слишком низкие уровни уверенности.

В системе используется уже обученная модель, способная распознавать 20 классов объектов, среди которых есть автомобили, автобусы и мотоциклы. Изначально данная модель разработана для использования с фреймворком Darknet, но для удобства использования граф был сконвертирован в формат TensorFlow.

При использовании перечисленных выше методов среди результатов будут встречаться ложноположительные, а также один объект может содержаться в нескольких пересекающихся прямоугольниках, что требует фильтровать результаты. Учитывая применение в дальнейшем алгоритмов отслеживания, для улучшения качества работы в случае спорных ситуаций лучше откинуть верное срабатывание, чем оставить ложное.

Существуют подходы на основе тепловой карты, но было решено использовать алгоритм Non-Maximum Suppression, обрабатывающий области по убыванию уверенности и откидывающий пересекающиеся с уже выбранными больше заданного порога. Также объединяются в одну те, что незначительно отличаются. Дополнительно анализируются несколько предыдущих кадров для отсечения случайных ложных срабатываний.

Отслеживание объектов

Для улучшения результатов в работе используется комбинация нескольких подходов связи областей между кадрами, каждый из которых может использоваться независимо.

При обработке кадра для каждого из объектов будем искать точки, которые можно было бы отследить в дальнейшем. Для этого воспользуемся алгоритмом Shi-Tomasi поиска углов. При обработке следующего кадра с помощью алгоритма Лукаса-Канаде для вычисления оптического потока (векторов перемещения для точек изображения) найдем точки, в которые перешли выбранные на предыдущем и из областей выберем ту, которая содержит большую часть из этих точек, но не менее заданного порога, она будет относится к тому же объекту.

Метод оптического потока хорошо работает только между соседними кадрами, но часто возникает ситуация, когда детектор на каком-то кадре или их серии не может обнаружить объект и в таком случае не получится установить соответствие, когда он будет обнаружен снова. Воспользуемся фильтром Калмана [3], использующим динамическую модель системы и множество последовательных измерений для формирования оптимальной оценки состояния. Алгоритм состоит из двух повторяющихся фаз – предсказания состояния в следующий момент времени и корректировки предсказанного значения новой информацией.

Фильтруемая величина - центр области, содержащей отслеживаемый объект. В большинстве случаев изменение этой величины на кадрах, когда объект потерян детектором, будет осуществляться по тому же закону, что и до этого. Будем выполнять только корректировку, если получилось связать области с помощью оптического потока, а в случае потери объекта – предсказание состояния. Для потерянных объектов каждый кадр ищем область, центр которой расположен ближе всего к предсказанному и она содержит его, связываем их и выполняем корректировку. Если в течении определенного количества кадров объект не был повторно найден, то считаем его окончательно потерянным.

Недостатком использования фильтра Калмана является то, что он не может предсказать положение объекта при малом количестве выполненных корректировок, а также при резком изменении закона перемещения в момент длительной потери детектором. Для решения данной проблемы используется алгоритм отслеживания объектов MOSSE (Minimum Output Sum of Squared Error) – стабильный корреляционный фильтр, который может быть инициализирован на одном кадре видео, адаптирующийся к изменениям внешнего вида.

Если требуется предсказание положения, но количество корректировок фильтра Калмана меньше заданного, то выполняется попытка поиска объекта на данном кадре с использованием MOSSE на основе фрагмента последнего кадра, на котором объект был обнаружен детектором. Если он успешно найден, то перед предсказанием выполняется корректировка с использованием полученного положения. Дальше такие корректировки выполняются только в случае, если по предсказанному положению не была найдена область.

Результаты вычислительного эксперимента

Для анализа были взяты ролики с YouTube, снятые на видеорегистраторы. Полученные результаты отслеживания показали, что система в целом справляется с задачей приемлемо, но существуют следующие неразрешенные использованными подходами проблемы:

1. Нейронная сеть и последующая фильтрация дают не всегда корректные данные: посторонние объекты изредка распознаются как автомобили; в ситуациях, когда два автомобиля находятся рядом, пространство между ними распознается как автомобиль с большей вероятность, чем эти автомобили отдельно; автомобиль может быть распознан как несколько малопересекающихся областей вместо одной большой.
2. Если были распознаны два объекта с небольшим перекрытием и второй оказался потерян на последующих кадрах, то первый может оказаться содержащим две метки.
3. Объекту может быть переназначена метка, если он будет потерян при совершении резкого маневра и местоположение будет предсказывать только фильтра Калмана.
4. При перекрытии одного автомобиля другим прекращается отслеживание скрытого и возможно переназначение его метки видимому при его потере детектором.

Список литературы

1. Vehicle Detection by Haar Cascades with OpenCV [Электронный ресурс]. – Режим доступа: https://github.com/andrewssobral/vehicle_detection_haarcascades
2. Real-time object detection with YOLO [Электронный ресурс]. – Режим доступа: http://machinethink.net/blog/object-detection-with-yolo
3. Фильтр Калмана – Введение / Хабрахабр [Электронный ресурс]. – Режим доступа: https://habrahabr.ru/post/140274/

ВОПРОСЫ ОБЕСПЕЧЕНИЯ КАЧЕСТВА ПРОГРАММНЫХ ПРОДУКТОВ В IT-ПРОЕКТАХ СИСТЕМ ОРГАНИЗАЦИОННОГО УПРАВЛЕНИЯ

Ананьев Т.П., Васильева О.В., Данилин А.В. – студенты, Астахова А.В. – к.э.н., доцент
Алтайский государственный технический университет (г. Барнаул)

В современных условиях развития информационных технологий наблюдается достаточно высокая конкуренция. В связи с этим возрастают требования к качеству IT-проектов. В соответствии со стандартом ISO/IEC 25000:2014 [1], «качество программного обеспечения — способность программного продукта при заданных условиях удовлетворять установленным или предполагаемым потребностям». При этом модель качества программного продукта, в соответствии со стандартом ISO/IEC 25010:2011 (ГОСТ Р ИСО/МЭК 25010-2015) [2], включает следующие основные характеристики верхнего уровня:

- функциональная пригодность;
- уровень производительности;
- совместимость;
- удобство использования (юзабилити);
- надёжность;
- защищённость;
- сопровождаемость;
- переносимость (мобильность).

Очевидно, что характеристики верхнего уровня при разработке IT-проекта не должны быть проигнорированы, так как они напрямую влияют на содержание характеристик качества нижнего уровня. В данных тезисах рассмотрены такие характеристики модели качества, как: функциональная пригодность, уровень производительности, совместимость, удобство использования (юзабилити), сопровождаемость.

Для учета названных характеристик в проекте, прежде всего, требуется решить такую актуальную задачу, как разработка полного и подробного технического задания, адекватно отображающего автоматизируемые процессы и объекты предметной области.

Ниже обобщен опыт авторов данных тезисов по исследованию конкретной предметной области и разработке соответствующего технического задания на IT-проект. Проект имеет название: «Система автоматизации управления материально-техническим снабжением (МТС) студенческого городка с учётом требований потребителей».

Написание технического задания (ТЗ) – это один из первых этапов работы над проектом. ТЗ – это обязательный документ, определяющий цели, требования и основную исходную информацию, необходимую для разработки и внедрения IT-проекта. Рассмотрим основные результаты, полученные авторами данных тезисов в рамках ТЗ.

Первая рассматриваемая в ТЗ задач – задача формирование вербальной модели предметной области, качество описания которой напрямую влияет на функциональную пригодность и удобство использования программного продукта.

На основе вербальной модели выделяются функции, подлежащие автоматизации, и устанавливается их взаимодействие с другими информационными системами. Функциональная пригодность и удобство использования будущего IT-продукта во многом определяются адекватным отображением в предметной области содержания и технологии реализации функций и процедур, выполняемых сотрудниками реальной системы в рамках их должностных обязанностей. Другими словами, функциональная пригодность и юзабилити напрямую зависят от качества описания процессов предметной области.

Остановимся на фрагменте вербальной модели предметной области, который позволяет выделить как объекты, так и процессы, протекающие в ней. Материальное обеспечение в студенческом городке включает в себя достаточное количество предметов имущества: столы, стулья, компьютеры и т.д. Для того, чтобы планировать, контролировать и регулировать наличие и состояние объектов имущества, а также формировать отчётные документы, существует система управления материально-техническим снабжением. В системе управления задействованы такие «механизмы управления» как заведующий складом и заведующий хозяйством (пользователи информационной системы).

Для обеспечения функциональной пригодности, а также удобства работы пользователей с системой, важен учет распределения должностных обязанностей между ними. В этой связи вербальная модель области должна отображать закрепление должностных обязанностей. Приведем пример, кратко иллюстрирующий данное высказывание.

Планированием поставок объектов имущества занимается заведующий складом на основе своей должностной инструкции и уставов: ВУЗа и студенческого городка. К зав. складом поступают заявки на ресурсы от потребителей (студентов, аспирантов). Если при этом требуется замена используемого ранее имущества, то заведующий складом получает извещение о замене объекта имущества. На основе обработки этих данных он создает план закупки необходимых товаров и план материально-технического снабжения (МТС) на следующие периоды. В случае списания имущества заведующий складом получает от завхоза акт о списании. Если происходит перемещение объекта имущества между помещениями внутри подразделения, он получает накладную на внутреннее перемещение. При передаче имущества от одного подразделения другому заведующий складом получает акт приема-передачи имущества. На основе обработки всех полученных документов оперативного учета зав. складом вносит изменения в план МТС на период.

Другая важная характеристика ПО – удобство (использования программного продукта) эффективно реализуется в случае учета в IT-проекте технологии работы специалистов с конкретными управленческими документами, которые используются при реализации соответствующих управленческих процедур. Модель предметной области при её описании должна обязательно учитывать названную особенность. Ниже приведен фрагмент вербальной модели, иллюстрирующий данный тезис.

Составленный план закупки передается отделу закупки ВУЗа. План МТС на период и учет поставок используются заведующим хозяйством (завхозом) студенческого городка для анализа и регулирования материально-технического снабжения.

Оперативный учет поставок объектов имущества (на основе счета-фактуры и товарно-транспортных накладных) осуществляется завхозом и заведующим складом и включает следующие процедуры: инвентаризацию (с формированием ведомости инвентаризации), анализ результатов инвентаризации (с формированием сличительной ведомости), учет брака в поставках, учет выбытия, расчет остатков – с регистрацией и/или формированием соответствующих первичных документов. На основе информации из инвентарных карточек и актов о списании заведующий складом рассчитывает остаток имущества на складе, с учетом которого формируется оборотная ведомость, а в инвентарные карточки вносятся необходимые изменения.

Качество вербальной модели зависит не только от качественного отображения таких характеристик, как «функциональная пригодность» и «удобство использования», но и от такой характеристики, как «совместимость». Ниже приведен фрагмент предметной области, которая отображает взаимодействие субъектов системы управления, которое должно учитываться при информационной и алгоритмической увязке модулей, реализующих соответственно автоматизированные рабочие места (АРМ) завскладом и завхоза студгородка, а также связи с вышележащим в системе эшелоном управления.

Получая план на период МТС и инвентаризационную опись, заведующий хозяйством занимается регулированием в области МТС. Если в ходе инвентаризации выяснилось, что определенные объекты имущества были перемещены, завхоз создает акт приема-передачи имущества, либо накладную на внутреннее перемещение и передает документ заведующему складом для планирования МТС. На основе инвентарных карточек и инвентаризационной описи завхоз создает отчет “Анализ выполнения плана МТС” и передает его на верхний эшелон управления.

Таким образом, составленная вербальная модель в словесной форме описывает объекты и процессы, происходящие в организации до внедрения и начала использования автоматизации, содержит полное описание предметной области с учетом таких характеристик как «функциональная пригодность», «удобство использования» будущего проекта, «совместимость» с другими проектами, позволяя в итоге увидеть достаточно полную картину анализируемой системы.

Вербальная модель – основа для разработки формализованной модели процессов.

Следующая задача – моделирование процессов предметной области с использованием case-средств формализации. Решение этой задачи также нацелено на реализацию в ТЗ таких характеристик как «функциональная пригодность» и «совместимость». Построение модели процессов в формализованном виде позволяет, с одной стороны, более углубленно рассмотреть предметную область, с другой, – выделить основные информационные потоки и список автоматизируемых процедур для разработки требований к прикладному ПО.

Авторами данных тезисов использовалась нотация IDEF0 для характеристики описанной выше многоуровневой системы управления МТС студгородка. В первую очередь, была разработана контекстная диаграмма, которая описывает управление МТС снабжением студенческого городка с учетом требований пользователей, приведенная на рисунке 1. Далее следует декомпозиция, описывающая четыре основных функции, приведенная на рисунке 2. После следуют декомпозиция процесса «оперативный учет МТС», приведенная на рисунке 3. При выполнении декомпозиции глубже изучаются процессы предметной области, и таким образом повышается функциональная пригодность программного продукта.

Анализируя предметную область, выделяем исходные и результирующие данные, представленные документами. Так как обмен между системами происходит с использованием формализованных документов и этот обмен регулируется положением о подразделениях, то можно наладить взаимодействие с системами, с которыми связана разрабатываемая информационная система. Тем самым на данном этапе глубже прорабатывается вопрос совместимости продукта.

В рассматриваемом проекте на входе контекстной диаграммы имеем: извещение о замене объекта имущества; информацию инвентарных карточек; заявки на ресурсы; счет-фактуру; товарно-транспортную накладную; материальный поток объектов имущества.

В результате реализации данного укрупненного процесса формируются: план закупки товаров; отчет «Анализ выполнения плана МТС»; оборотная ведомость.

К нормативным правовым документам и нормативно-организационным актам в данной системе относятся: устав студенческого городка; устав ВУЗа; должностные инструкции завскладом и завхоза студенческого городка. При этом в качестве механизмов реализации процессов на основе нормативной информации и фактического состояния системы выступают специалисты: заведующий складом и заведующий хозяйством.

Следующей задачей является углубление информационной модели предметной области: разработка предварительных требований к структуре данных, в данном случае базе данных. Более детальная проработка этого вопроса на уровне ТЗ позволяет избежать грубых ошибок на более поздних этапах реализации IT-проекта. С точки зрения качества программных продуктов, данный этап влияет на эффективность таких характеристик как «функциональная пригодность» и «уровень производительности». Авторами данных тезисов спроектированы модели процессов предметной области, разработаны проектные предложения по ее концептуальной модели, в которой уточняющей состав классов объектов, определяются их атрибуты и взаимосвязи, разработаны требования к составу и структуре ПО. Концептуальная модель отображается в виде файлов (таблиц) реляционной базы данных, отображающих содержание первичных информационных потоков предметной области (потока нормативно-справочной информации; потока учетных показателей; потока формируемых плановых показателей). Представив информационную модель предметной области в виде структуры базы данных, изображенной на рисунке 4, разработчики IT-проекта при проектировании автоматизированного рабочего места (АРМ) получают возможность работать не только в терминах предметной области, но и с объектами модели.

Здесь следует подчеркнуть, что разработка грамотных требований к проекту базы данных влияет на дальнейший уровень производительности системы, так как надо использовать правильные типы данных, эффективную организацию индексов и грамотную структуру хранения.

На основе анализа предметной области и концептуальной модели составляются требования к АРМ, также являющиеся частью технического задания. Требования к АРМ влияют, во-первых, на удобство использования продукта конечными пользователями, во-вторых, участвуют в формировании функциональной пригодности, так как отражают аспекты, связанные напрямую с автоматизацией, которые нельзя было выделить на предыдущих этапах разработки ТЗ, отвечавших в основном за анализ протекающих процессов в уже имеющейся системе.

Ещё одной важной частью технического задания является сетевой график, который будет реализован в будущем и позволит эффективно организовать работы, входящие в жизненный цикл системы, частью которого является сопровождение. Тем самым заранее прорабатывается «сопровождаемость», как одна из характеристик качества продукта.

ТЗ влияет на многие факторы при разработке ПО, поэтому включает большое количество разделов. Здесь рассмотрены основные разделы, влияющие на качество ПО. При разработке ТЗ авторы данных тезисов старались максимально использовать графические материалы для наглядного и сжатого представления информации: одна диаграмма зачастую в состоянии заменить несколько страниц текста.

Из указанных выше характеристик модели качества не были рассмотрены: надежность; защищенность; переносимость. Они должны предусматриваться на стадии разработки проектов (технического и рабочего).

В заключении следует отметить, что разработанный проект в виде ТЗ может быть использован в качестве учебного примера при изучении студентами направления подготовки «Программная инженерия» некоторых учебных дисциплин, в том числе: «Основы экономики программной инженерии и управление проектами»; «Технология командной разработки ПО»; «Проектирование человеко-машинных интерфейсов», а также при выполнении по соответствующей тематике курсовых работ и ВКР.

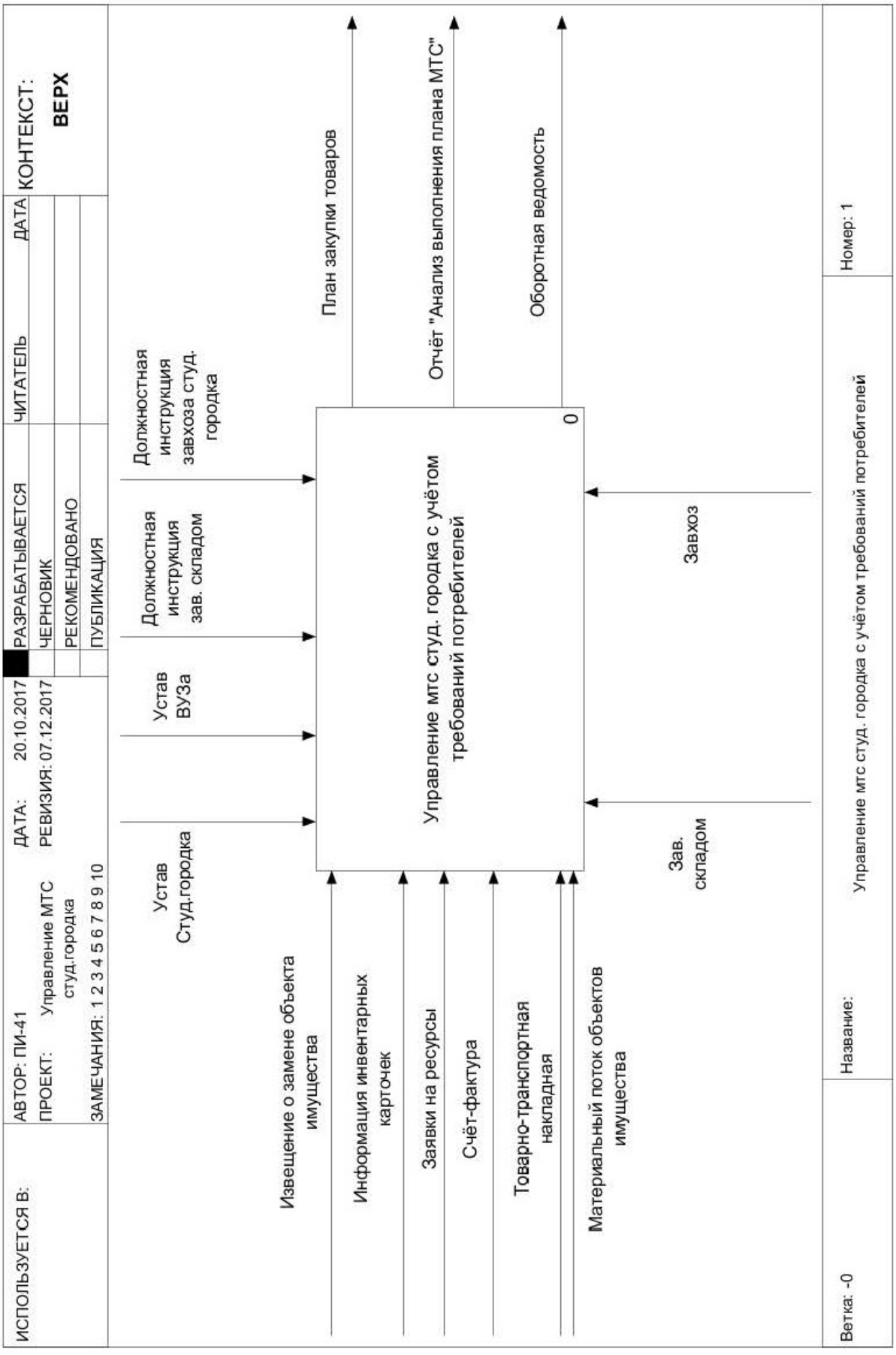


Рисунок 1 – Контекстная диаграмма

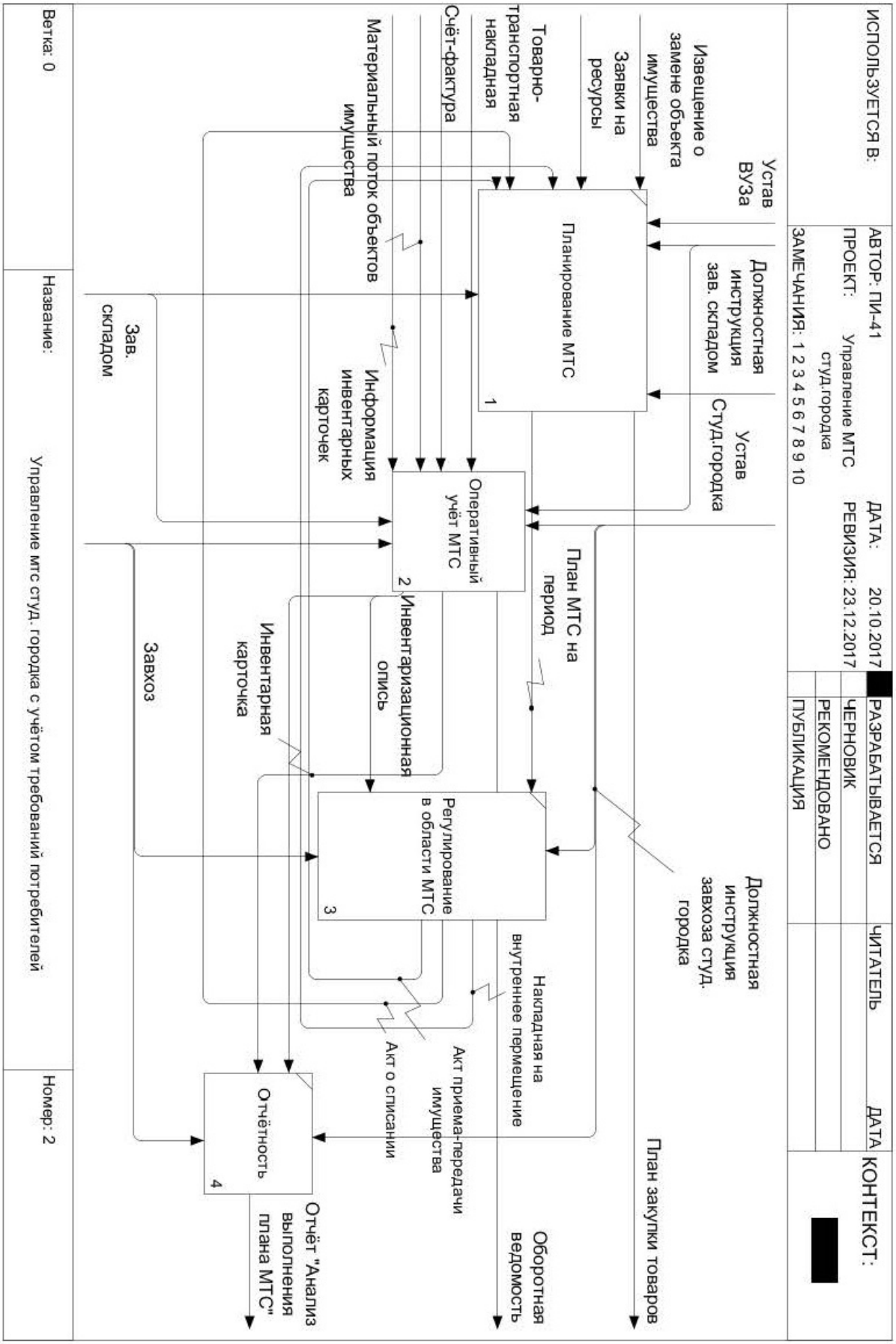


Рисунок 2 – Диаграмма декомпозиции

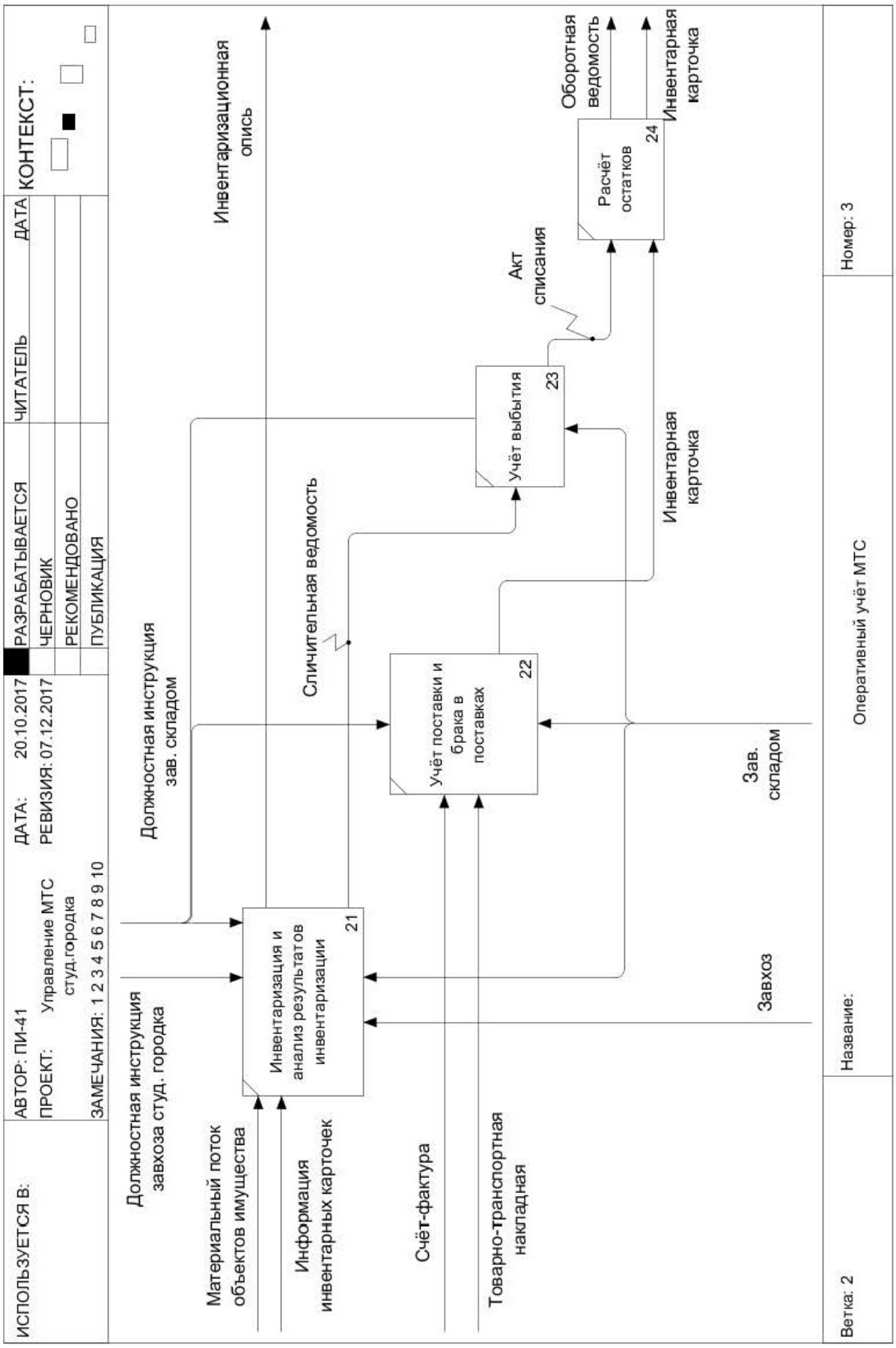


Рисунок 3 – Диаграмма второго уровня декомпозиции

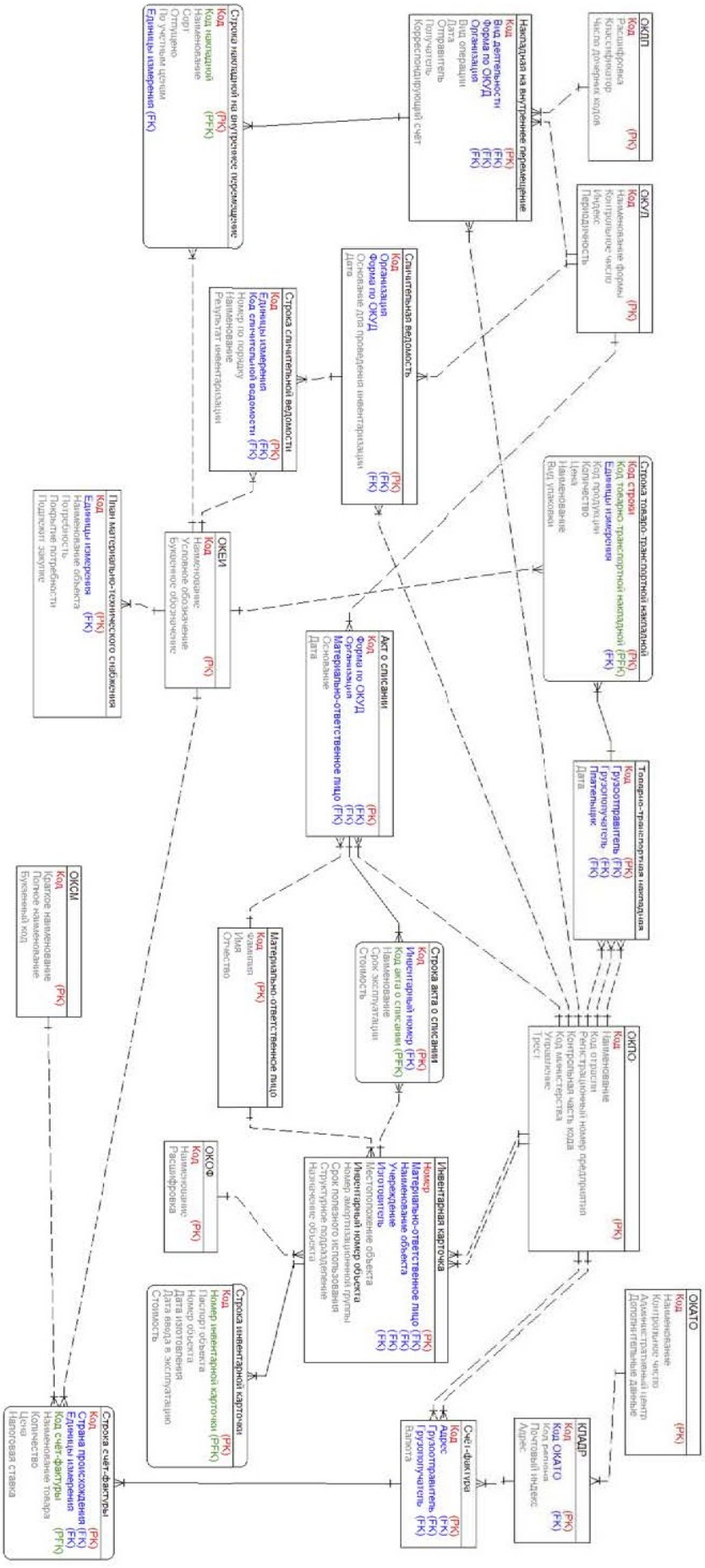


Рисунок 4 – Концептуальная модель

Список литературы

1. ISO/IEC 25000:2014(en) Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Guide to SQuaRE [Электронный ресурс]. – Режим доступа: https://www.iso.org/obp/ui/#iso:std:iso-iec:25000:ed-2:v1:en
2. ГОСТ Р ИСО/МЭК 25010-2015 Информационные технологии. Системная и программная инженерия. Требования и оценка качества систем и программного обеспечения (SQuaRE). Модели качества систем и программных продуктов [Электронный ресурс]. – Режим доступа: http://docs.cntd.ru/document/1200121069

ПРОЕКТИРОВАНИЕ СИСТЕМЫ ПРОВЕДЕНИЯ СОРЕВНОВАНИЙ ПО ПРОГРАММИРОВАНИЮ ИГРОВЫХ СТРАТЕГИЙ

Ананьев Т.П., Фаст А.С. – студенты, Крючкова Е.Н. – к.ф.-м.н., профессор
Алтайский государственный технический университет (г. Барнаул)

Помимо наиболее распространенных соревнований по программированию, в которых участникам необходимо решить несколько задач и решения проверяются на заранее подготовленных наборах входных и выходных данных, проводятся соревнования [1], в которых участникам необходимо придумать и реализовать на одном из языков программирования стратегию для игры с определенными правилами, в которой им предстоит действовать против игрового окружения и стратегий других участников. Данный вид соревнований представляет интерес из-за непредсказуемости результатов, отсутствия конкретного решения и азарта соревнований с другими стратегиями.

Одной из ключевых особенностей является наличие визуального пошагового представления действий участников на игровом поле, что позволяет оценивать корректность работы собственной стратегии и анализировать действия других участников.

Перед нами стоит задача спроектировать и реализовать систему, которая позволяла бы упростить процесс организации и проведения соревнований в данной форме, взяв на себя весь функционал, не отличающийся между различными играми. Взаимодействие с системой должно производиться через Web-интерфейс, а для реализации стратегий участникам должны быть доступны популярные языки программирования общего назначения.

Структура системы

Для достижения поставленной задачи была проработана структура данной системы, в которой основной акцент сделан на модульность, которая позволяет упростить разработку несколькими людьми, развертывание и взаимодействие частей в работающей системе. Также возможно дальнейшее расширение системы, например, для проведения классических соревнований по программированию на её основе.

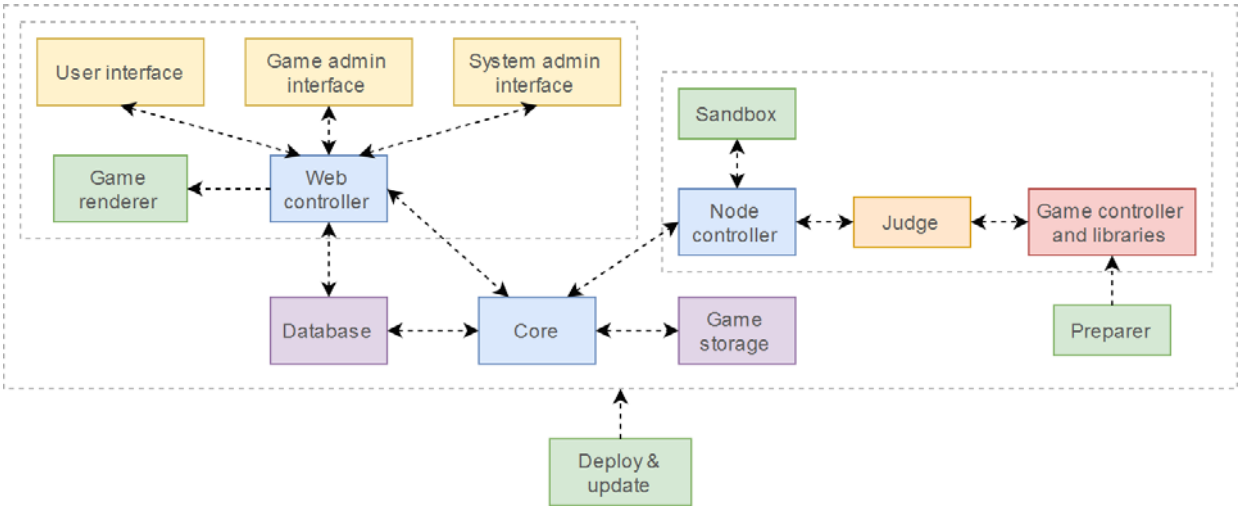


Рисунок 1 – Предлагаемая структура системы

Основным модулем системы является ядро *(Core)*, отвечающее за запуск системы и синхронизацию отдельных её элементов. Совместно с ядром в одном приложении работает контроллер Web-интерфейса *(Web controller)*, который отвечает за работу пользовательского интерфейса и взаимодействие пользователей с ядром платформы.

Пользовательский интерфейс можно условно разбить на три части в зависимости от роли пользователя:

- интерфейс посетителей сайта и зарегистрированных пользователей;
- интерфейс разработчика игры;
- интерфейс администратора системы.

В зависимости от роли пользователя в интерфейсе становятся доступны те или иные элементы управления. За определение роли пользователя отвечает Web-контроллер, также проверяющий права пользователей на выполнение определённых действий и пресекающий попытки несанкционированного доступа.

Также отдельной составной частью пользовательского интерфейса можно выделить визуализатор игр (Renderer) - он доступен и в интерфейсе пользователя, и в интерфейсах администраторов. Данный модуль отвечает за визуальное отображение пошагового выполнение игр.

Для хранения всех необходимых данных система использует два типа хранилищ. Для хранения информации о пользователях и всех связанных с соревнованиями данных (алгоритмы и настройки игр, истории и результаты запусков, решения участников) используется база данных (Database), что позволяет легко организовать резервное копирование данных и при необходимости перенос системы. Для хранения временных данных, используемых только в рамках одного соревнования для синхронизации запусков и определения результатов, предназначено игровое хранилище (Game storage), расположенное в оперативной памяти. Игровое хранилище представляет собой набор key-value таблиц.

Разработчик игры должен реализовать логику придуманной им игры на языке программирования Java и загрузить её в систему в виде скомпилированного jar-файла, называемого в дальнейшем game. В соответствии с технологией dependency injection [2] один из классов должен реализовывать определенный интерфейс, позволяющий game и модулю judge взаимодействовать друг с другом. Для данных, выходящих за рамки одного раунда, game должен реализовывать stateless логику, так как в различных раундах одного соревнования не гарантируется использование одного и того же экземпляра объекта, для этих данных он должен использовать игровое хранилище.

Интерфейс, который необходимо реализовать разработчику игры включает в себя:

- логику выбора набора участников следующего раунда игры и порядка их действий на каждом шаге;
- интерфейс взаимодействия между решениями участников и game, а через него с решениями других участников и игровым полем;
- логику изменения игрового поля с течением времени;
- логику отображения процесса игры, используя инструменты модуля Renderer.

Решения участников взаимодействует с системой и game через автоматически добавляемую при их компиляции прослойку, обеспечивающую участнику максимальную простоту чтения и записи необходимых данных. Данная прослойка генерируется для всех допустимых языков программирования автоматически модулем Preparer при загрузке game в систему на основе предоставленного автором интерфейса взаимодействия.

Тестирование решений участников может быть распределено между одним или несколькими тестирующими узлами, которые могут быть расположены на удаленных машинах. За управление работой такого узла отвечает модуль Node controller, который подключается к ядру и взаимодействует с хранилищами через него, что позволяет организовать верификацию узлов и защитить хранилища от несанкционированного доступа. Для ограничения действий участников на каждом узле развернута песочница (Sandbox), в которой запускаются скомпилированные решения. За запуск решений отвечает модуль Judge, который взаимодействует с game и отвечает за запуск решений участников в песочнице.

Модульная структура проекта, применение автоматической системы сборки gradle, а также использование системы контроля версий в процессе разработки, позволят осуществлять простое и быстрое развертывание актуальной версии системы и её обновление, в том числе в полностью автоматическом режиме.

Заключение

Таким образом, в статье были предложены архитектура и требования к системе проведения соревнований по программированию в формате игровых стратегий, позволяющие максимально упростить организацию подобного вида соревнований. При проектировании заложены гибкость, масштабируемость и универсальность, расширяющие область применения.

Список литературы

1. Competetive Programming: Towards a New Era of eSports? [Электронный ресурс]. – Режим доступа: https://www.codingame.com/blog/competitive-programming-towards-new-era/
2. Inversion of Control Containers and the Dependency Injection pattern [Электронный ресурс]. – Режим доступа: https://www.martinfowler.com/articles/injection.html

АРХИТЕКТУРА ДИНАМИЧЕСКИ КОНФИГУРИРУЕМЫХ СЦЕНАРИЕВ, ИСПОЛНЯЕМЫХ В ИЗОЛИРОВАННОМ ОКРУЖЕНИИ

Фаст А.С., Ложкина Д.Д. – студенты
Алтайский государственный технический университет (г. Барнаул)

Актуальность

Современные, распределенные программно-аппаратные комплексы отличаются повышенной сложностью из-за большого числа компонентов. И требуют специализированных инструментов для решения задач сопровождения.

Одним из распространённых типов таких инструментов является консоль администратора, позволяющая разрабатывать скрипты на интерпретируемых языках программирования для модификации данных системы и прямого вызова прикладной логики.

Платформа поддержки развития бизнеса (ППРБ) - инструмент для создания бизнес-приложений, в разработке которой применяется новейшие технологии распределенных вычислений в памяти и работы приложений с большими объемами данных в реальном времени – In Memory Data Grid [1].

Для решения инцидентов и устранения их последствий у администраторов сопровождения ППРБ возникает необходимость выполнения следующих задач:

- Модификация бизнес-данных:
  - Удаление дублируемых записей;
  - Модификация данных при обновлении модели данных;
  - Исправление ошибок в данных (опечатки, неверных формат данных);
  - Корректировка справочников.
- Операции над моделью хранения данных:
  - Перестройка индексов;
  - Валидация индексов;
  - Валидация метаданных;
  - Анализ колокации данных (распределение данных по узлам).
- Вызов прикладной логики:
  - Повторный вызов операции после восстановления работоспособности;
  - Проверка интеграционных взаимодействий;
  - Изолированное тестирование компонентов.

Подобные операции на текущий момент невозможны либо требуют приостановки доступа к сервису из-за перезапуска узлов приложения, что критично для выполнения соглашения об уровне обслуживания системы (SLA) [2] и, в некоторых случаях, недопустимо. Также требуются большие затраты на разработку отдельного инструмента под каждый конкретный случай.

Проблема

Необходимость перекомпиляции приложения и сложные процедуры развертывания - одни из ключевых проблем сопровождения решений на Java, что приводит к длительному времени устранения инцидентов в случае некорректной работы одного или нескольких компонентов платформы.

Отсутствие универсального механизма выполнения динамически конфигурируемых сценариев (скриптов) без полномасштабного развертывания является серьезной проблемой для сопровождения платформы ПРБ. Реализация подобного инструмента является критичным требованием инженеров сопровождения к разрабатываемой платформе.

Предлагаемое средство для решения задачи

Предлагаемый сервис призван устранить недостатки существующих решений путем предоставления инженерам сопровождения технического сервиса для решения задач по модификации данных и выполнения бизнес-логики приложений путем исполнения сценариев в виде скриптов на интерпретируемом языке без необходимости перезапуска приложения.

Сервис может поддерживать любой совместимый с JVM язык, в настоящий момент используется Groovy [3].

Архитектура сервиса

Сервис состоит из трёх модулей и базы данных (Рисунок 1).

- Модуль runner отвечает за запуск сценариев в отдельных потоках, создание для них изолированных окружений и загрузку классов (может быть развернут в нескольких экземплярах, для одновременной работы с несколькими разнородными хранилищами данных);
- Модуль hot-fix-tool-ui-static – пользовательский интерфейс;
- Модуль hot-fix-tool-ui отвечает за связь пользовательского интерфейса с базой данных и модулем runner (контроллер из паттерна MVC).
- База данных HFT хранит скрипты, объединенные в группы скриптов, и jar-файлы, из которых были загружены классы для каждого скрипта.

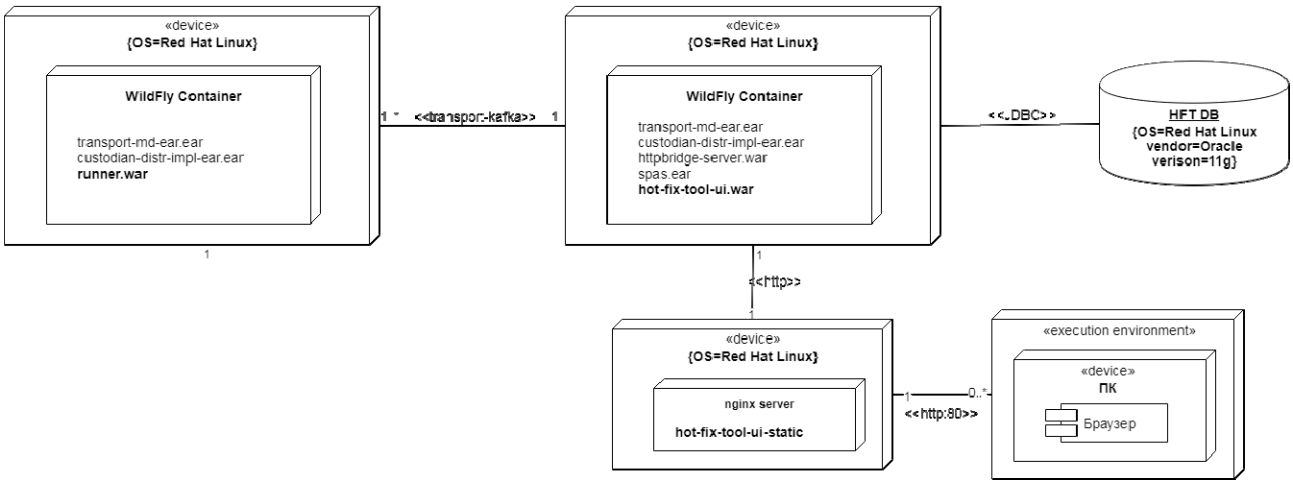


Рисунок 1 - Модель развертывания

В качестве сервера приложений используется WildFly, как хорошо зарекомендовавшее себя и основное решение для всех компонентов платформы ПРБ.

Разработка и выполнение сценариев

Для обеспечения комфортной, в том числе и многопользовательской работы, сервис реализует хранение созданных сценариев в базе данных. Скрипты объединяются пользователями в группы, внутри групп обеспечивается уникальность имени скрипта. При работе с сервисом пользователь может как создать новый скрипт, так и выбрать любой сценарий из ранее сохраненных и работать с ним.

В интерфейсе пользователю предоставляется возможность ввода имени и исходного кода скрипта, а также возможность загрузить jar-файлы с классами компонентов, необходимых для работы сценария.

В процессе выполнения сценария у разработчика может возникнуть необходимость вывода какой-либо информации. Например, сообщения для отслеживания прогресса работы скрипта, значения каких-либо переменных, состояние объектов и т.д. Сервис предоставляет такую возможность. Разработчик может использовать логгер с различными уровнями логгирования (Error, Info, Warning) для отслеживания прогресса выполнения скрипта и вывода необходимых сообщений. Например, с помощью вызова функции логгера с уровнем info в нужном месте программы, можно увидеть в пользовательском интерфейсе сообщение с введенный текстом “Start modify data”, что будет означать, что скрипт начал модифицировать данные (Рисунок 2).

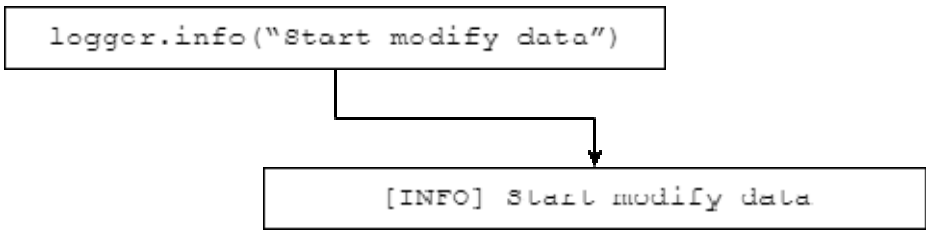


Рисунок 2 - Пример использования логгера

Для комфортной работы пользователя данный логгер работает по той же схеме, что и известные фреймворки (Log4J, Java Logging API, SLF4J).

При запуске сценария на выполнение его исходный код отправляется на сервер, где перед запуском скрипта происходит проверка всех классов, указанных в блоке импортирования. В случае невозможности загрузить какие-либо классы, их имена будут выведены пользователю, и запуск сценария не будет произведен.

Также разработчик может указать места, в которых возможно произвести безопасную остановку скрипта. При достижении каждого такого места будет произведена проверка, отправил ли пользователь запрос на остановку, и если он такой запрос отправил, то скрипт будет остановлен. Использование точек возможной остановки скрипта обусловлено тем, что принудительная остановка скрипта в произвольном месте может нарушить целостность данных, с которыми в данный момент работал сценарий.

Например, скрипт

```
logger.info(“start modify first block of data”)
… // код модификации первого блока данных
interruptPoint() // точка возможной остановки
logger.info(“start modify second block of data”)
… // код модификации второго блока данных
```

может быть остановлен в точке возможной остановки, если во время модификации первого блока данных пользователь отправит запрос на остановку скрипта. В таком случае пользователю будет выведен следующий лог:

```
[INFO] start modify first block of data
… // лог, который выводит первый блок кода
[WARN] interrupted – сообщение о том, что скрипт был остановлен.
```

Возможность параллельного запуска

При проведении работ по устранению инцидентов может потребоваться запуск нескольких скриптов одновременно. Сервис предоставляет такую возможность, для этого пользовательский интерфейс организован с использованием вкладок, а запуск сценариев и их обработка происходит в отдельных потоках. Для управления потоками используется пул потоков [4], который позволяет эффективно использовать ранее созданные потоки, без необходимости создавать новые, но с возможностью расширения пула при нехватке. Для запуска каждого сценария выделяется два потока: первый отвечает за выполнение скрипта, а второй необходим для отслеживания состояния процесса выполнения скрипта.

Так как разработчик может использовать логгер в исходном коде скрипта, после вызова логгера в процессе исполнения необходимо вывести соответствующее сообщение в пользовательском интерфейсе. Для решения данной задачи оба потока содержат ссылку на одну и ту же потокобезопасную очередь. Логгер, встроенный в сценарий, организует

добавление сообщений в очередь, а второй поток периодически опрашивает ее на предмет появления новых сообщений. UI-контроллер при поступлении сообщений отправляет их непосредственно в пользовательский интерфейс (Рисунок 3).

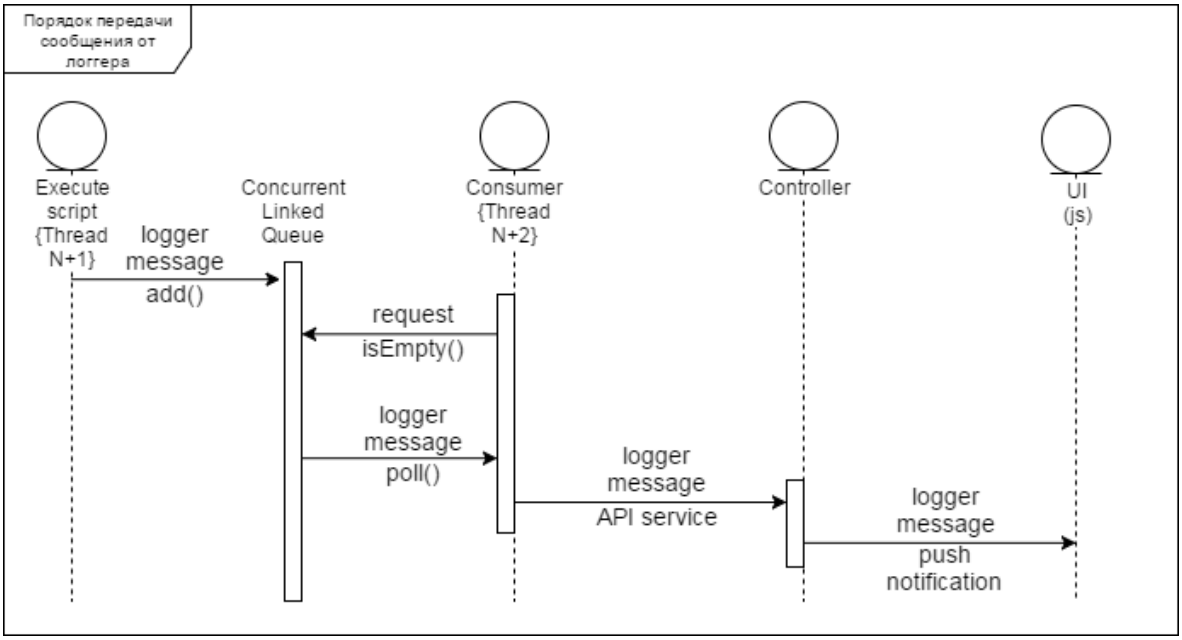


Рисунок 3 - Порядок передачи сообщения от логгера в пользовательский интерфейс

Изолированное окружение

Сценарий может содержать логику, для которой требуются сторонние классы, однако по умолчанию возможно использование только контекста приложения. Одной из основных особенностей платформы Java является модель динамической загрузки классов, которая позволяет загружать исполняемый код в JRE, не перезагружая основное приложение.

Реализация пользовательского динамического загрузчика классов (Dynamic Class Loader) позволяет загружать классы в контекст приложения из выбранного пользователем jar-файла, а выполнение каждого скрипта с использованием разных загрузчиков и создает изолированное окружение, а значит позволяет использовать стороннюю логику, классы и интерфейсы для каждой отдельной задачи.

В Java SE имеет место основанная на делегировании иерархия загрузчиков классов, которая в общем случае выглядит следующим образом:

1) Bootstrap (базовый)
2) Extensions (загрузчик расширений)
3) Application (системный загрузчик)
4) Пользовательский (если существует)

Каждый загрузчик классов (кроме Bootstrap) имеет ссылку на родителя, и в большинстве случаев запрашивает его определить указанный класс перед тем, как попробовать загрузить его самостоятельно.

Так как в данном приложении для каждого скрипта определен пользовательский загрузчик с собственным кэшем загруженных им классов, процесс делегирования выглядит следующим образом (Рисунок 4)

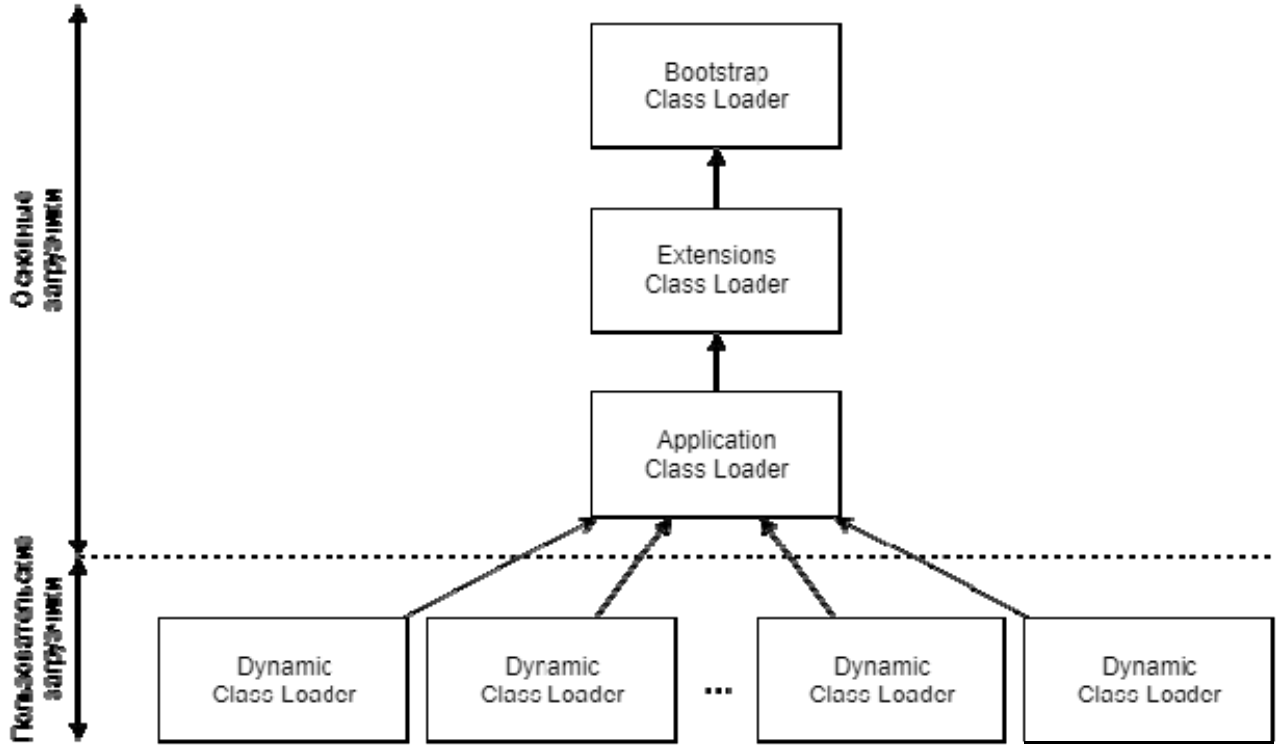


Рисунок 4 - Процесс делегирования

С точки зрения JVM уникальный идентификатор класса образует пара, состоящая из полного имени класса и загрузчика. То есть один и тот же загрузчик не может определить два класса с одинаковыми именами. Однако, учитывая, что каждый загрузчик имеет свое пространство имен для создаваемых классов, для повторной загрузки класса достаточно определить два загрузчика на одном уровне иерархии. Подобные дублированные классы будут рассматриваться JVM как различные, так как они определены разными загрузчиками.

Аналогичное решение используется для возможности обновить версию уже загруженного класса в рамках одной задачи. Для этого загрузчик «пересоздает» сам себя, то есть создается новый объект класса DynamicClassLoader, который инициализируется данными предыдущего загрузчика (кэш классов и справочник jar-файлов). После требующий обновления класс загружается из файла уже новым загрузчиком, почему и распознается JVM, как другой класс (Рисунок 5).

Для оптимизации работы приложения при загрузке jar-файла сразу происходит его сканирование и запоминание классов, которые он содержит. Загрузчик организован таким образом, что для каждого используемого пользователем класса сохраняет информацию о его полном имени, файле, из которого он может быть загружен и записи в этом файле с этим классом. В том случае, когда в файле содержится класс, полное имя которого совпадает с уже присутствующим в JVM классе, необходимо пересоздать загрузчик, что и происходит после окончания сканирования всех jar-файлов.

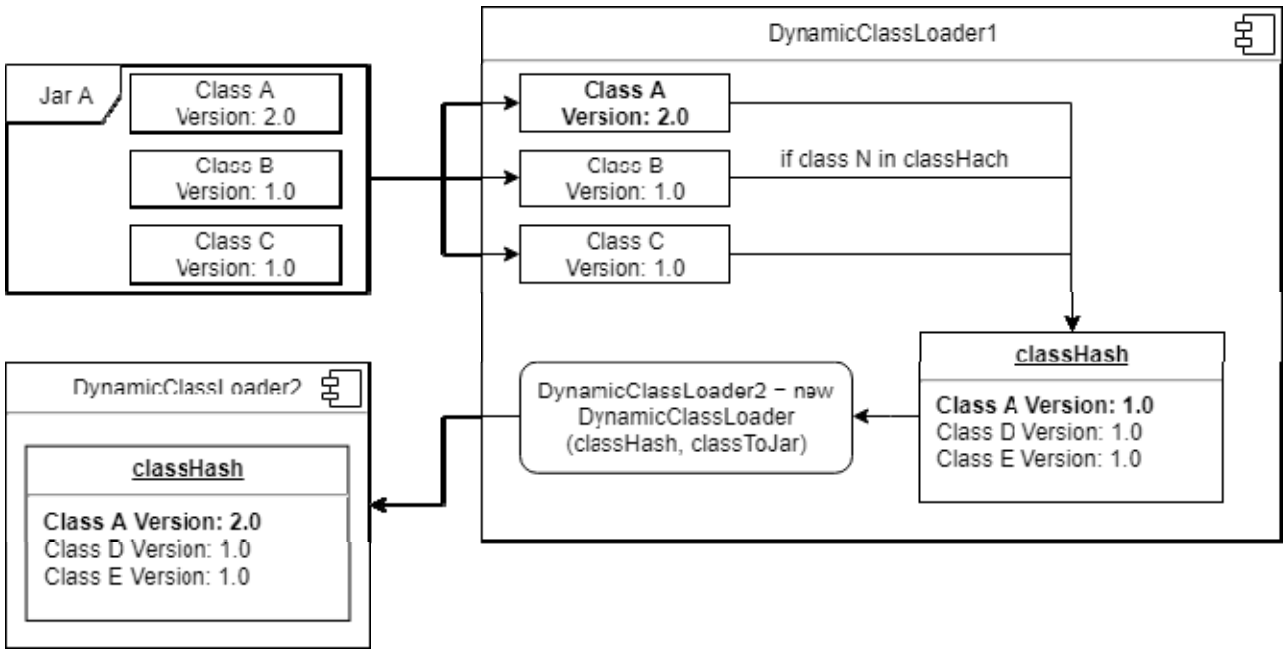


Рисунок 5 - Обновление версии класса

Во время исполнения текста скрипта при обращении к какому-либо классу происходит проверка, присутствует ли он в кэше динамического загрузчика. Если его там нет, значит, данный класс еще не был загружен. Тогда происходит проверка, есть ли класс с таким именем в одном из jar-файлов, и, в случае успеха, текущий загрузчик загружает его и запоминает в собственный кэш, иначе задача по распознаванию этого класса делегируется родительскому загрузчику.

Заключение

Данный сервис позволяет администраторам сопровождения платформы разрабатывать и запускать скрипты для решения инцидентов и устранения их последствий. Скрипты могут быть запущены параллельно и независимо друг от друга. Время устранения инцидентов для определенных классов ошибок уменьшилось с нескольких дней до нескольких минут. Инвестиции в инфраструктуру упрощают внедрение и эксплуатацию платформы в будущем.

Список литературы

1. Платформа поддержки развития бизнеса [Электронный ресурс]. - Режим доступа: http://sbertech.ru/component/k2/item/22.html
2. Gerard Blokdijk. The Service Level Agreement SLA Guide - SLA book, Templates for Service Level Management and Service Level Agreement Forms. Fast and Easy Way to Write your SLA. - Lightning Source. – 2008. – 204 с.
3. A multi-faceted language for the Java platform [Электронный ресурс]. - Режим доступа: http://groovy-lang.org/
4. Шилтд, Г. Java 8. Полное руководство; 9-е изд.: Пер. с англ. - М.: ООО "И.Д. Вильямс", 2015. - 1376 с.

РЕАЛИЗАЦИЯ ПРОГРАММЫ ДЛЯ АУТЕНТИФИКАЦИИ ПОЛЬЗОВАТЕЛЯ ПО ГОЛОСУ

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

В последнее время активно развиваются системы аутентификации человека, связанные с различными биометрическими характеристиками человека. Это связано в частности с активным ростом популярности мобильных устройств, в которые производители внедряют разные виды аутентификации пользователя. В связи с этим распознавание человека по голосу является актуальным и, хоть и не является абсолютно точным, может быть использована в совокупности с другими методами.

В работах по распознаванию голоса наиболее популярен метод кепстрального преобразования спектра речевых сигналов. Схема метода такова: на интервале времени в 10 — 40 мс вычисляется текущий спектр мощности, а затем применяется обратное преобразование Фурье от логарифма этого спектра (кепстр) и находятся коэффициенты, характеризующие данный отрывок речи.

В данной работе был реализован алгоритм распознавания диктора на основе вычисления мел-кепстральных коэффициентов по известным записям голоса и последующего их сравнения с распознаваемой записью.

Мел – психофизическая единица высоты звука, характеризующая особенности восприятия звука человеческим ухом. Соотношение между частотой звука и высотой, воспринимаемой человеком, примерно описывается логарифмической зависимостью.

$$M(f) = 1125\ln(1 + f/700)$$

Рисунок 1 – Переход от Герц в Мел

$$M^{-1}(m) = 700(\exp(m/1125) - 1)$$

Рисунок 2 – Переход от Мел к Герц

Общая схема работы программы такова:

- Для всех дикторов рассчитать мел-кепстральные коэффициенты на основе образцов записей голоса;
- Рассчитать набор коэффициентов для распознаваемой записи и вычислить вероятность принадлежности голоса каждому диктору.

Мел-кепстральные коэффициенты рассчитываются независимо для каждого короткого отрывка – в программе реализована возможность задания длины данного отрывка в диапазоне от 20 до 128 мс. Притом разбиение записи на отрывки – фреймы – происходит с половинным наложением.

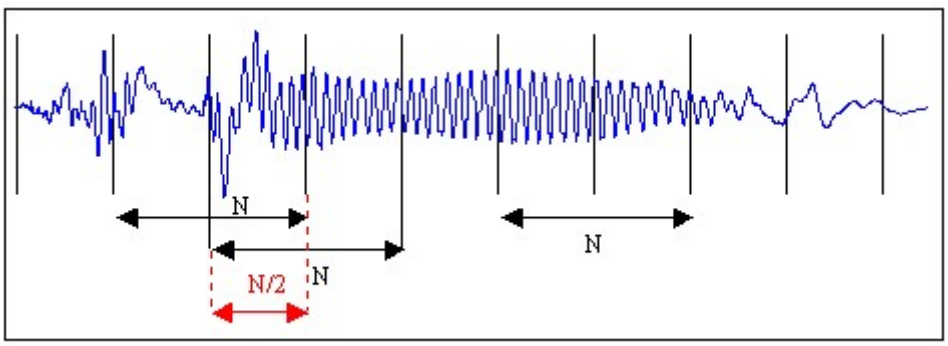


Рисунок 3 – «Нарезка» звука на фреймы

Набор рассчитанных коэффициентов для всех фреймов и является характеристикой, по которой определяется вероятность принадлежности голоса тому или иному диктору.

Опишем вкратце схему расчета мел-кепстральных коэффициентов для конкретного фрейма.

1. Предварительно обработать запись – выполнить нормализацию, очистить от шума;
2. Умножим значения на весовую функцию – «окно», это необходимо для устранения нежелательных эффектов при расчете преобразования Фурье, которое ориентированно на бесконечно повторяющийся сигнал. В данной работе используется окно Хэмминга;
3. Выполним дискретное преобразование Фурье;
4. Рассчитаем периодограмму - оценку спектральной плотности мощности;

$$P_i(k) = \frac{1}{N}\left|S_i(k)\right|^2.$$

Рисунок 4 – Расчет периодограммы

5. Рассчитаем набор мел-фильтров;
   a. Зададим количество фильтров равным 26 (используют значения от 20 до 40). Обрабатываемый диапазон в Герц зададим от 300 Гц до 8кГц. Обе границы переведем в Мел и полученный диапазон в Мел равномерно разобьём на отрезки, добавив между границами ровно 26 точек (получим 27 равных отрезков);
   b. Каждое полученное значение переведем обратно в Герц, обозначим их как h(i) и получим индексы в периодограмме, соответствующие этим частотам. Для этого необходимо знать количество элементов, участвовавших в преобразовании Фурье (N) и частоту дискретизации (SR).

$$F(i) = \left[(N+1) * \frac{h(i)}{SR} \right].$$

Рисунок 5 – Переход к индексам периодограммы

   c. Мел-фильтры представляют собой набор треугольных окон, которые на краях равны 0, в средней точке – 1.

$$H_m(k) = \begin{cases} 0, & k < f(m-1) \\ \dfrac{k - f(m-1)}{f(m) - f(m-1)}, & f(m-1) \le k \le f(m) \\ \dfrac{f(m+1) - k}{f(m+1) - f(m)}, & f(m) \le k \le f(m+1) \\ 0, & k > f(m+1) \end{cases}$$

Рисунок 6 – Мел-фильтры

   d. Для расчета m-го фильтра необходимо вычислить следующую сумму:

$$f(m) = \sum_{n=f(m-1)}^{f(m+1)} (P(k) * H_m(k))$$

Рисунок 7 – Расчет m-го фильтра

e. Теперь полученные значения необходимо прологарифмировать. После логарифмирования становится возможным отсечение фреймов, в которых записана тишина – значения логарифмов будут большими. В данной работе реализовано игнорирование фрейма, если максимальное значение фильтра превышает 10^4;
f. После этого шага нужно оставить только фильтры с 2 по 13, так как остальные фильтры будут нести мало информации о говорящем;
g. Последним пунктом алгоритма является дискретное косинусное преобразование второго типа. Оно заключается в умножении вектора фильтров на матрицу преобразования.

$$\text{DCT-2}_n = \left[\cos\left(k\left(l+\tfrac{1}{2}\right)\tfrac{\pi}{n}\right)\right]_{0\le k,l\le n}.$$

Рисунок 8 – Матрица дискретного косинусного преобразования

6. Полученный вектор из 12 значений и будет вектором Мел-кепстральных коэффициентов.

Рассчитав таким образов набор коэффициентов для каждого фрейма, получаем набор векторов, характеризующий конкретного диктора. Но так как разбиение выполнялось на небольшие интервалы, то и векторов будет много. Очевидно, что их надо каким-либо образов классифицировать.

В данной работе реализована кластеризация векторов методом k-средних, в котором выделяются 6 кластеров для каждого диктора – это обеспечивает достаточно высокую скорость работы алгоритма и приемлемую точность распознавания. Алгоритм заключается в следующем:

1. Выделить некоторое количество центров из исходного набора случайным образом;
2. Для каждой точки, определить кластеру какого центра она принадлежит – какой из центров ближайший;
3. В каждом кластере пересчитать центр – как среднее арифметическое всех векторов, в него входящих;
4. Повторять пункты 2-3 пока уменьшается суммарное квадратичное отклонение точек кластеров до центров этих кластеров, но не более заданного числа раз.

После проведения кластеризации для векторов, характеризующих данного диктора, становится возможным расчет вероятности принадлежности какой-либо записи этому диктору. Для этого применяется следующий алгоритм:

1. Рассчитать набор мел-кепстральных коэффициентов для распознаваемой записи, используя те же параметры, что были использованы при расчете коэффициентов для записей данного диктора;
2. Для каждого вектора рассчитать отклонение от центров кластера, обозначим как X. Среднее отклонение всех векторов, характеризующих диктора, от центров кластера обозначим как Y. Тогда вероятность принадлежности фрейма диктору рассчитывается как

$$Z = 2 * X / Y,$$

$$P = \min\left(1, \frac{4}{X+3}\right).$$

Рисунок 9 – Расчет вероятности принадлежности фрейма диктору

3. Считается, что запись может принадлежать данному диктору если рассчитанная вероятность не менее 0.7.

Реализованная программа позволяет собрать набор записей для нескольких дикторов, производить расчет мел-кепстральных коэффициентов, кластеризацию и, на основе этих данных, распознавание говорящего. Возможен ввод данных с микрофона, и загрузка записей из файла.

Данный алгоритм позволяет производить довольно точную идентификацию говорящего, но конечно же не исключает вероятность ошибки.

СИСТЕМА КОМПЬЮТЕРНОГО ЗРЕНИЯ ДЛЯ АВТОМАТИЧЕСКОГО АНАЛИЗА И КЛАССИФИКАЦИИ МИКРОСТРУКТУРНОГО ИЗОБРАЖЕНИЯ

Данилин А.В. – студент, Крючкова А.В. – к.ф.-м.н., профессор
Алтайский государственный технический университет (г. Барнаул)

Введение

Микроструктура – это мелкомасштабная структура материала, которую можно обнаружить под микроскопом как минимум с более чем 25-кратным увеличением. Микроструктурные изображения играют важную роль в различных областях современной науки в таких как медицина и материаловедение. За прошедшее столетие медики добились больших успехов в приобретении, анализе и сравнении микроструктурных изображений. Значительная часть усилий была направлена на глубокое понимание конкретных систем материалов или классов.

Если известен каталог возможных микроструктурных признаков или их легко перечислить, можно использовать методы анализа цифровых изображений для сравнивнения микроструктур. Но если набор признаков или представляющих интерес особенностей неизвестен, когда микроструктуры отличаются незначительными параметрами, проблема анализа становится сложной, а выбор оптимальной схемы сегментации зависит от предварительного знания особенностей анализируемой микроструктуры, причем для многих материалов в данный момент не существует адекватной схемы сегментации. Это ограничивает общую применимость методов микроструктурного анализа, которые работают на сегментированных изображениях.

Предлагаемое решение

Цель проекта - разработка нового общего метода для поиска полезных характеристик и отношений внутри и между микроструктурными изображениями без каких-либо предположений о том, какие функции могут присутствовать. Исследования в области компьютерного зрения, особенно функции распознавания текстуры изображения заложили основу для такого подхода. Распознавание текстуры изображения относится к задаче количественного определения пространственного распределения интенсивности изображения для идентификации отдельных областей. Обычно используемые функции включают фильтр Габора, вейвлет-преобразования, характерные сегменты изображения и локальные дескрипторы изображения. Наличие огромного объёма данных цифровой микроструктуры [1-3], накопленного за последние два десятилетия, облегчают реализацию.

Описание используемого метода

В этой работе применяется представление изображения в виде Bag of Words (BoW), которое обычно используется при распознаванию объектов и сцен [4]. Модель представляет изображение как набор визуальных слов без учета их взаимного расположения и взаимных связей. При ее использовании семантическая близость двух изображений как двух наборов визуальных слов оценивается по статистике - количеству совпадающих слов. Это означает, что два изображения, в которых мало общих слов или вообще нет, считаются неблизкими. Таким образом, гистограмма виуальных слов моделирует класс изображений как распределение вероятности по набору визуально отличительных признаков. Это представление изображения можно быстро вычислить для произвольных и несегментированных изображений, что дает количественное представление микроструктуры.

Обычно организация BoW разделена на этапы: обнаружение и характеристика локальных черт изображения, извлечение визуального словаря (как правило, посредством кластерного анализа по подмножеству обнаруженных признаков) и построение пакета представлений визуальных черт, которые могут быть дополнительно проанализированы с использованием различных подходов к компьютерному обучению.

Функция обнаружения ключевых точек

Чтобы локализовать отличительные микроструктурные функции, которые будут полезны для характеристики, мы используем ключевые точки. Используемый оператор процентных точек обнаруживает редкий набор областей изображения, которые содержат сложную структуру градиента изображения. Использовалась OpenCV – популярная библиотека компьютерного зрения с открытым исходным кодом.

Многие подходы компьютерного зрения представляют собой визуальную структуру функций ключевых точек посредством шаблона или распределения локальных значений градиента изображения. В данной работе используется Scale-Invariant Feature Transform (SIFT), реализованный OpenCV [5-7]. SIFT является обычно используемым дескриптором ключевой точки, который предлагает вращательную инвариантность, используя опорный кадр, который нормализуется по отношению к характерному масштабу каждой отдельной функции ключевой точки. Характерный масштаб каждой функции ключевой точки вычисляется во время этапа обнаружения ключевой точки.

Извлечение визуального словаря

В BoW рассматривается изображение как набор локальных шаблонов градиента изображения, каждый из которых представлен высокоразмерным вектором. Полученное пространство объектов квантуется в визуальный словарь, заданного пользователем размера k, чтобы обеспечить быстрое совпадение и применение методов поиска информации. Визуальный словарь получается путем выполнения кластерного анализа на случайном подмножестве дескрипторов ключевых точек, извлеченных из репрезентативного набора образов. Результатом будет словарь, состоящий из заданного k числа функций. Один визуальный словарь извлекается из набора изображений, содержащих весь спектр микроструктурных изменений, относящихся к задаче. Таким образом, для изучения систем связанных материалов (то есть сплавов) примеры каждой представляющей интерес системы материалов должны быть включены в набор тренировок, используемый для построения визуального словаря.

Построение гистограмм соответствия

BoW для изображения представляет собой нормированную гистограмму, измеряющую частоту каждого визуального слова на изображении. Каждая функция ключевой точки считается экземпляром визуального слова, соответствующего ближайшему центру кластера в

визуальном словаре. Таким образом, гистограмма изображения является отпечатком микроструктуры, которой она соответствует. Чем меньше разница между гистограммами изображений (следовательно, чем больше общих черт), тем больше подобия между микроструктурами.

Особенности сравнения

Существует множество металлов и сплавов. В случае добавления некоторой примеси в сплав его микроструктурное изображение может разительно измениться. В то же время некоторые сплавы могут быть не так очевидно различимы для программы в связи со схожестью дескрипторов ключевых точек.

Рассмотрим несколько сплавов меди, они представлены на рисунках 1, 2, 3 и 4.

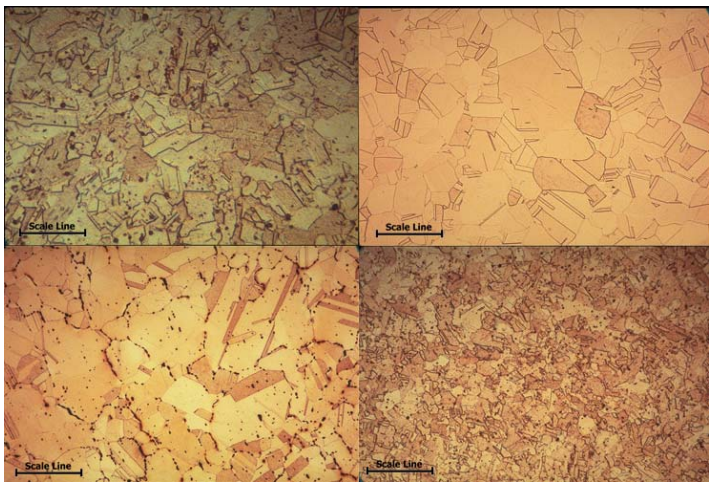


Рисунок 1 – Сплавы меди

А вот некоторые микроструктурные изображения высокопрочного чугуна и ковкого чугуна не так легко различить.

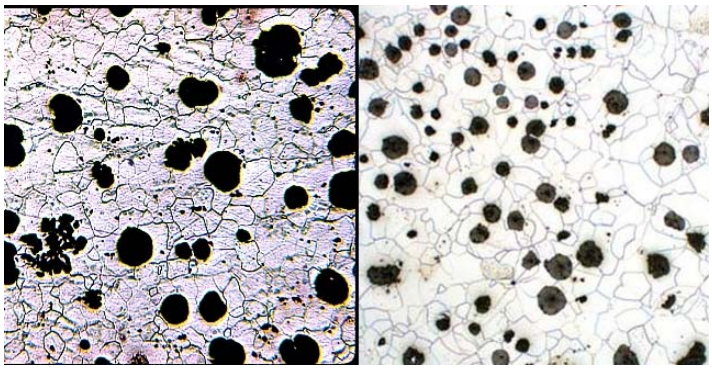


Рисунок 2 – Высокопрочный чугун

Список литературы

1. DocMe [Электронный ресурс]. – Режим доступа: http://www.docme.ru/download/1146207 / Атлас металлов и сплавов.
2. Металл-экспертиза [Электронный ресурс]. – Режим доступа: http://metall-expertiza.ru/articles/349921 / Сборник микроструктур металлов и сплавов
3. eKNIGI [Электронный ресурс]. - Режим доступа: https://eknigi.org/nauka_i_ucheba/79036-atlas-makro-i-mikrostruktur-metallov-i-splavov.html / Атлас макро- и микроструктур металлов и сплавов

4. Gabriella Csurka, Christopher R. Dance, Lixin Fan, Jutta Willamowski, Cédric Bray: Visual Categorization with Bags of Keypoints, 2013.
5. Прохоренок В. OpenCV и Java обработка изображений и компьютерное зрение. - Издательство: БХВ-Петербург, 2018.
6. CodeProject [Электронный ресурс]. - Режим доступа: https://www.codeproject.com/Articles/619039/Bag-of-Features-Descriptor-on-SIFT-Features-with-O / Bag-of-Features Descriptor on SIFT Features with OpenCV (BoF-SIFT)
7. Docs OpenCV [Электронный ресурс]. - Режим доступа: https://docs.opencv.org/3.3.0/da/df5/tutorial_py_sift_intro.html / Introduction to SIFT (Scale-Invariant Feature Transform)

ПРОБЛЕМА ВЫДЕЛЕНИЯ ЖИВЫХ ОБЪЕКТОВ НА ИЗОБРАЖЕНИЯХ

Бабушкин И.В., Данилин А.В. – студенты, Крючкова Е.Н. – к.ф.-м.н., профессор
Алтайский государственный технический университет (г. Барнаул)

Актуальность

Раньше природа диктовала человечеству правила жизни. Теперь её влияние значительно меньше, а человек сам влияет на неё, нанося зачастую ей заметный ущерб. В результате антропоморфного фактора многие виды животных находятся под угрозой исчезновения и занесены в красную книгу. Одним из таких редких видов является амурский тигр (далее просто тигр), сегодня численность тигров колеблется около 550 особей.

Для учёта и контроля популяции редких животных в местах обитания тигров устанавливаются специальные камеры-ловушки, которые срабатывают на движущиеся объекты. Камеры предоставляют серии снимков одной или нескольких особей тигров с одного ракурса. Данные с камер ловушек собирают раз в несколько месяцев, после этого производится опознавание тигров. Дополниельно для этой цели можно мспользовать снимки, сделанные фотографами на территории заповедника.

В данной работе предлагается система, предназначенная для автоматизации локализации тигра на снимке. В работе рассматриваются статические и динамические методы. Статические методы можно использовать только на серии снимков с одного ракурса, а динамические методы могут работать с отдельными фотографиями. В данной статье мы рассмотрим три метода:

1. Поиска фона
2. Использования Mask R-CNN
3. Использования YOLO

Поиск фона на серии снимков

Серия снимков - это снимки сделанные с определенным интервалом с одной фотоловушки. У серии снимков общий фон, что можно использовать для отделения объектов съемки [1, 2]. Если цветовые значения пикселов на двух или более снимках совпадает, этот пиксель представляет фон, в противном случае на одной из фотографий этот пиксель принадлежит движущемуся объекту. Пусть a *и* b – две фотографии серии, тогда с учетом порогового значение p для отсечения шума для каждого пиклеля вычислим маску

$$res_{ij} = \begin{cases} |a_{ij} - b_{ij}|, & |a_{ij} - b_{ij}| > p \\ 0, & |a_{ij} - b_{ij}| \le p \end{cases}.$$

Теперь у нас в выделение попадают все движущиеся объекты. Для того чтобы избавиться от лишнего, нужно найти все разности между снимками серии. Итоговую маску *l*-ого снимка серии размера *n* будем вычислять как среднеарифметическое разностей снимков межу этим снимком и остальными:

$$res_{ijl} = \sum_{l \neq k} \frac{b_{ijl} - b_{ijk}}{n-1}.$$

Далее обработаем полученную маску следующим образом. Разобьем всю маску на небольшие вертикальные колонки. Будем опустим верхнюю границу до тех пор пока среднее значение небольшой окрестности не достигнет заданного порога. С нижней границей поступим так же. Фрагмент изображение между верхней и нижней границей и есть нужный объект.

Пример простого попиксельного сравнения приведен на рисунке 1. Маска, получения путем вычисления среднеарифметического разностей снимков, представлена на рисунке 2. Пример отделения фона представлен на рисунке 1.

Рисунок 1 – Результат отделения фона

Mask R-CNN

R-CNN (Regional Convolutional Neural Networks) – это сверточная нейронная сеть основанная на регионах [3]. Цель R-CNN состоит в том, чтобы взять изображение и правильно определить (помещая в рамки), где находятся основные объекты на изображении. Помещенная в рамки часть изображения называется регионом. Для каждого такого региона строится карта особенностей, тем самым определяется отличительные черты данного региона.

R-CNN создает эти рамки, используя процесс, называемый выборочный поиск. На высоком уровне для идентификации объектов выборочный поиск просматривает изображение через окна разных размеров, и для каждого размера пытается сгруппировать соседние пиксели по текстуре, цвету и интенсивности.

Существует четыре версии R-CNN: оригинальный R-CNN, Fast R-CNN и Faster R-CNN, отличающиеся быстродействием, а так же использующая рамки вместе с масками Mask R-CNN.

Mask R-CNN основана на Faster R-CNN, её главной отличительной чертой является решение задачи пиксельной сегментации изображения, то есть определение принадлежности конкретного пикселя изображения тому или иному объекту. Данная версия была получена дополнением Faster R-CNN ветвью использующей FCN (полную свёрточную нейронную

сеть) для генерации маски на основе карты особенностей, полученной на предыдущем шаге. Пример работы с исползованием Mask R-CNN представлен на рисунке 2.

YOLO

YOLO - система обнаружения объектов в реальном времени [4]. Данная система применяет нейронную сеть ко всему изображению. Сеть делит изображение на регионы и предсказывает рамки и вероятности для каждого региона. С использованием многоколоночной классификации каждая область, помещенная в рамки, предсказывает какие объекты и с какой вероятностью могут содержаться в этих рамках.

В нашем случае необходимо распознавать тигров, что данная нейросеть сама по себе делать не умеет. Очень часто тигр распознаётся как зебра, кошка, лошадь или медведь. Это происходит по той причине, что используемая нами нейросеть натренирована на этих животных, и они схожи по используемым классификаторам с тиграми. Главной причиной схожести зебр и тигров являются их полоски, нейросеть считает их схожими. Вероятность распознавания тигра как зебры значительно выше в случае, когда тигр повернулся боком и полоски хороши видны.

Рисунок 2 – Выделение с использованием Mask R-CNN

Рисунок 3 – Выделение с использованием YOLO

Структура данных

В данной системе важно хранить оригинальное изображение и части, которые мы определили. После отделения фона мы получаем только те части снимка на которых должны остаться тигры исключительно. Назовем отдельные независимые области, попавшие в выделение ***частями*** фотографии. Так как при разных методах вырезания фона могут получится отличные друг от друга части фотографии, то нужно хранить информацию о том, каким методом была получена данная часть фотографии. Создание таблицы методов нужно как для структуризации данных, так и для дальнейшего расширениея функционала. В данный момент мы используем три метода, и благодаря спроектированной структуре мы сможем добавить новый метод.

Каждая из загруженных фотографий привязана к сессии загрузки посредством коллекции. Коллекции введены для удобной систематизации загруженных фотографий. За одну сессию пользователь может загрузить сразу несколько фотографий, и они все будут проанализированы независимо друг от друга. Для каждой сессии хранится информация о пользователе, загрузившем фотографии.

Заключение

Проведенные эксперименты показали, что при решении данной зпдпчи каждый из описанных выше методов имеет свои преимущества и недостатки.

Метод обработки серии снимков со статической камеры может исполняться только на ЦП (CPU). В первую очередь, данный метод выделяется моментальной инициализацией и своим высоким быстродействием, более одного кадра в секунду, при этом возможно провести оптимизации, повышающие скорость работы. Наилучшим образом данный метод показал себя с серией снимков, снятых на одну статическую камеру с интервалом не более нескольких часов..

Метод Mask R-CNN является самым ресурсоемким, все расчеты производятся только на дискретных видеокартах (GPU). Метод работает достаточно долго. Один снимок может обрабатываться более одной секунды. Нейросеть может достаточно точно выделять объекты на фотографиях.

Метод с использованием системы YOLO предъявляет меньшие требования предъявляемые к аппаратному обеспечению по сравнению с R-CNN, может исполняться как на GPU, так и на CPU. Обработка фотографий на GPU осуществляется со скоростью около 12 - 15 кадров в секунду, но скорость значительно падает при использовании CPU. Кроме того, минусом метода является невозможность отделения фона, так лишь выделяются области,, где есть объект.

В роли GPU использовалась Nvidia GTX 970, в роли CPU - Intel Core i7 3770.

Список литературы

1. Background Subtraction [Электронный ресурс]. – Режим доступа: https://docs.opencv.org/3.1.0/db/d5c/tutorial_py_bg_subtraction.html
2. How to Use Background Subtraction Methods [Электронный ресурс]. – Режим доступа: https://docs.opencv.org/3.1.0/d1/dc5/tutorial_background_subtraction.html
3. A Brief History of CNNs in Image Segmentation [Электронный ресурс]. – Режим доступа: https://blog.athelas.com/a-brief-history-of-cnns-in-image-segmentation-from-r-cnn-to-mask-r-cnn-34ea83205de4 / A Brief History of CNNs in Image Segmentation
4. YOLO: Real-Time Object Detection [Электронный ресурс]. – Режим доступа: https://pjreddie.com/darknet/yolo / YOLO: Real-Time Object Detection

СОЗДАНИЯ ФРЕЙМВОРКА ДЛЯ РАБОТЫ С СЕРВИСАМИ РАСПОЗНАВАНИЯ РЕЧИ

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

Пару десятилетий назад писатели фантасты только мечтали о возможности взаимодействовать с компьютером при помощи голосовых команд. Сейчас развитие технологий предоставляет возможность распознавать голос и переводить его в текстовый вид, понятный для компьютера. Существуют множество решений, позволяющих трансформировать устную речь в письменный вид.

Программы с возможностью распознавания голоса обычно привязаны к одному сервису. В случае перевода программы на другой сервис, возникает необходимость вносить большое количество правок в программу. Внесение в стабильно работающий код большого количества правок всегда связано с риском получения большого числа ошибок в исполняемом коде программы. Все эти действия требуют большого количества времени, которое тратит разработчик ПО. Поэтому данную систему можно вынести в отдельный модуль, это решение

позволит облегчить перевод программы с одного сервиса на другой. Поэтому было решено написать модуль, который позволит использовать разные сервисы и легко менять их.

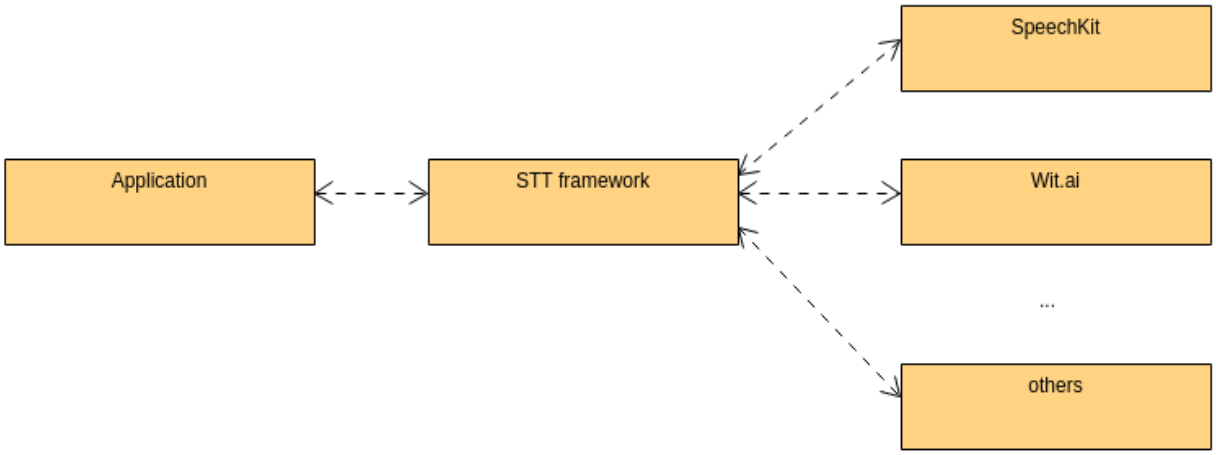


Рисунок 1 - Структура модуля

Для каждого сервиса пишется адаптер, все адаптеры реализуют единый интерфейс, с которым и происходит работа. Адаптеры в свою очередь помешаются в пул объектов. Когда приходит запрос на распознавание, из пула объектов выбирается сервис по определенным правилам. Правила выбора можно также с легкостью менять. Разработчику не придется писать свои адаптеры, так как адаптеры для популярных сервисов уже написаны. Разработчику всего лишь нужно вызвать функцию- конструктор, передав ей параметры в зависимости от сервиса. Мы можем объявить, что тот или иной сервис используется в бесплатном режиме с определенными ограничениями, которые могут учитываться при выборе адаптера из пула объектов.

Реализованы следующие правила выбора из пула:

- Выбор сервиса, у которого осталось больше запросов чем у других.
- Выбор сервиса, у которого стоимость запроса минимальна.
- Выбор сервиса, который может обработать аудио запись определенной длины.
- Выбор сервиса согласно приоритетам, установленным в ручную.

Заключение

В результате реализован модуль для языка Java, позволяющий легко подключать один или несколько популярных сервисов распознавания речи и работать с ними по единому образцу. Фреймворк позволяет распределять запросы на разные сервисы в зависимости от их загруженности, сервис распознающий речь изменяется одной командой.

Список литературы

1. Безлимитное распознавание речи. [Электронный ресурс]. - Режим доступа: https://habrahabr.ru/post/346334/
2. Распознавание речи с помощью CMU Sphinx [Электронный ресурс]. - Режим доступа: https://habrahabr.ru/post/267539/
3. Распознавание речи [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D0%BF%D0%BE%D0%B7%D0%BD%D0%B0%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D1%80%D0%B5%D1%87%D0%B8

4. Mohri M., Pereira F., & Riley M. Speech recognition with weighted finite-state transducers. In Springer Handbook of Speech Processing. - Springer Berlin Heidelberg, 2008. - P. 559-584.
5. Hinton G., Deng L., Yu D., Dahl G. E., Mohamed A.R., Jaitly N. et al. . Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. // Signal Processing Magazine, IEEE. - 2012. - 29(6). - P. 82-97.
6. Jurafsky D., Martin J.H. Speech and language processing, 2nd edition. - Prentice Hall, 2008.

РАЗРАБОТКА ВИЗУАЛЬНОЙ НОВЕЛЛЫ

Деулина А.Д. – студент
Алтайский государственный технический университет (г. Барнаул)

Сегодня многие мечтают донести свои мысли и идеи до других людей. Написание рассказов и больших книг, рисование комиксов или анимаций, сочинение музыкальных или танцевальных композиций – всё это помогает им выразить те идеи, что терзают сознание, не дают покоя, пока не выйдут на свет. Однако при объединении всех таких усилий получится намного более полная и точная передача мыслей. Ведь зритель получит не только историю, но и её графическое представление, вкупе с настраивающей на нужный лад музыкой.

Один из путей подобного объединения – создание визуальной новеллы. Визуальный роман (яп. ビジュアルノベルбидзюарунобэру, от англ. visualnovel) - жанр компьютерных игр, подвид текстового квеста, в котором зрителю демонстрируется история при помощи вывода на экран текста, статичных (либо анимированных) изображений, а также звукового и/или музыкального сопровождения. Нередко используются и вставки полноценных видеороликов. Степень интерактивности в таких играх обычно низка, и от зрителя лишь изредка требуется сделать определённый выбор, в частности — выбрать вариант ответа в диалоге. Персонажи этих игр обычно выполнены в стиле аниме, который, как и визуальные романы, возник в Японии[1]. Для облегчения работы над подобной новеллой обычно используется движок Ren'Py. Ren'Py (от ren и Python) - это бесплатный, свободный и открытый движок для создания как некоммерческих, так и коммерческих визуальных романов (графических квестов с диалоговой системой) в 2D-графике [2]. Разработка визуальной новеллы на этом движке сама по себе - не самая сложная задача. Главное - обладать навыками писателя, чтобы грамотно преподнести свою историю. Если же человек хочет самостоятельно разработать новеллу от начала до конца, он должен быть ещё и сценаристом, художником, музыкантом.

Особой популярностью сейчас пользуются игры, в которых персонажи "ломают четвёртую стену" - обращаются напрямую к игроку, минуя игровые ограничения. Также интерес вызывают так называемые "глитчи" - программно предусмотренные искажения графических и/или звуковых файлов. Идея воплотить нечто подобное в жанре визуальной новеллы пришлась мне по вкусу.

Простейшие новеллы не требуют особых навыков программирования - движок весьма дружелюбен. Но для решения такой задачи, как моя, без программирования не обойтись. Чтобы игра обращалась не к персонажу, а к тому, кто за ним стоит, необходимо считать информацию из профиля пользователя, запустившего игру: дату и время, имя компьютера, и другие параметры среды. Код для таких изменений записывается на языке Python в соответствующих скрипт-файлах. Неспецифические изменения изображений (глитчи) - увеличение или уменьшение изображений, изменение цвета, движение по экрану, - также

требуют написание программного кода. Для достижения подобных результатов используется специальная библиотека классов.

На данный момент создана игра, в игре реализованы программные изменения изображений в ходе определённых выборов в истории и некоторые элементы поломанной "четвёртой стены". При дальнейшей разработке планируется добавить больше эффектов поломанной "четвёртой стены". Стоит также добавить, что для созданной работы был самостоятельно написан текст сценария и код программы, а также нарисованы изображения персонажей.

Рисунок 1 - Вид главного меню

Рисунок 2 - Вид изменённого изображения

Список литературы

1. Визуальный роман [Электронный ресурс] // Wikipedia. – Режим доступа: https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%B7%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%80%D0%BE%D0%BC%D0%B0%D0%BD

2. Ren'Py [Электронный ресурс] // Wikipedia. – Режим доступа: https://ru.wikipedia.org/wiki/Ren%27Py

АРХИТЕКТУРА СОВРЕМЕННЫХ SPA ПРИЛОЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ БИБЛИОТЕК REACT И REDUX НА ПРИМЕР КОМПОНЕНТА ПРОГРЕССА ПОЛЬЗОВАТЕЛЯ

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

В настоящее время растет популярность одностраничных веб-приложений (SPA), которые по своему функционалу приближаются к полноценным десктопным, но имеют преимущества в виде доступности с любого мобильного устройства. С ростом функционала растет и сложность пользовательского интерфейса (UI) приложения, все сложнее становится поддерживать UI согласованным с состоянием приложения, облегчить это призваны современные библиотеки и фреймворки. Рассмотрим архитектуру SPA-приложений на примере использования библиотек React и Redux.

React - это библиотека JavaScript, разработанная Facebook для создания пользовательского интерфейса на базе парадигмы реактивного программирования. В React весь пользовательский интерфейс описывается декларативно и представляется с помощью компонентов. Каждый компонент имеет некоторые входные параметры (props) и может иметь свое внутреннее состояние (state). Компонент описывает отображение параметров и состояния на UI. При любом изменении параметров или состояния запускается механизм рендеринга. Так как операции перерисовки объектной модели документа (DOM), формируемой браузером, достаточно дорогостоящие, React, получая на вход приложение в виде дерева компонентов, для каждого компонента запускает механизм рендеринга, которые строит сначала виртуальное DOM-дерево, после этого сравнивает реальный DOM в браузере и виртуальным DOM и точечно применяет изменения к реальному DOM, что позволяет экономить на дорогостоящих операциях изменениях с реальным DOM деревом[1].

Рассмотрим данный механизм на примере реализации компонента, отображающего прогресс пользователя. На рисунке 2 показан исходный код компонента. Главной функцией в любом компоненте, является функция render, она описывает отображение входных параметров и состояния компонента на UI. Компонент прогресса принимает текущий уровень пользователя, кол-во очков, необходимое для получения следующего уровня, и текущее кол-во очков пользователя. На рисунке 1 изображен внешний вид компонента. Так как React использует подход реактивного программирования, то, как только изменятся входные параметры, компонент обновится и UI станет согласованным с общим состоянием приложения.

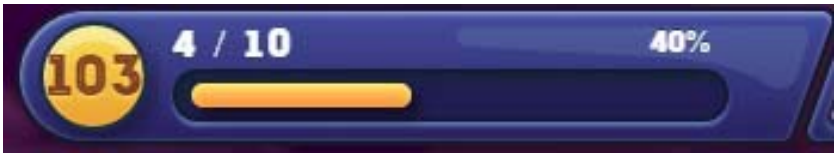


Рисунок 1 – Внешний вид компонента «Прогресс пользователя»

React отвечает только за визуализацию компонентов, поэтому с ним обычно используют библиотеку, которая отвечает за хранение и управление глобальным состоянием

приложения. Redux – самая популярная библиотека для управления состоянием, используемая совместно с React. Redux построена на функциональном подходе, и использует три объекта. Состояние (state) описывает состояние всего приложение в виде обычного Javascript объекта. Действие (action) - простой Javascript объект, который описывает изменение, вносимое в состояние. Создатель действий (action creator) - функция, результатом которой является действие или несколько действий, которые будут применены к состоянию. Редьюсер (reducer) – функция, которая применяет действия ко всему состоянию или к части состояния. Redux использует функциональный подход и требует, чтобы редьюсер был «чистой» функцией, то есть редьюсер должен возвращать новое состояние вместо изменения предыдущего и результат функции должен зависеть только от входных данных. Это облегчает сравнения состояния, так как можно не сравнивать все дерево состояния, а только проверить, изменилась ли ссылка по сравнению с прошлым состоянием. Все асинхронные взаимодействия должны совершаться только в создателях действий (action creators) [2].

```
export class HeaderLeftPanel extends React.Component<HeaderLeftPanelProps, { isNewLevel: boolean }>{
    constructor(props) {
        super(props);

        this.state = { isNewLevel: false }
    }

    componentWillReceiveProps(props: HeaderLeftPanelProps) {
        if (this.props.level && props.level > this.props.level) {
            this.setState({ isNewLevel: true });
        }
    }

    resetNewLevelFlag = () => {
        if (this.state.isNewLevel) {
            this.setState({ isNewLevel: false });
        }
    }

    render() {
        let nextLevelProgress = this.props.levelPoints / this.props.pointsToNextLevel;
        if (!nextLevelProgress || nextLevelProgress == NaN)
            nextLevelProgress = 0.01
        const progressBarMaxWidth = 230;

        return (
            <div >
                <div onClick={this.resetNewLevelFlag} className={`level ${this.state.isNewLevel ? "new-level" : ""}`} id="level">
                    {this.props.level}
                </div>
                <div onClick={this.resetNewLevelFlag} className={`new-level-msg-wrapper ${this.state.isNewLevel ? "visible" : ""}`}>
                    <div className="new-level-msg">{LocalizeText("HeaderLeftPanel.ReachedNewLevel")}</div>
                </div>
                <div className="level-points" id="level-points">{this.props.levelPoints} / {this.props.pointsToNextLevel}</div>
                <div className="level-progress" id="level-progress" style={{ width: progressBarMaxWidth * nextLevelProgress }}></div>
                <div className="level-progress-background"></div>
                <div className="level-percentage" id="level-percentage">{(nextLevelProgress * 100).toFixed(0) + "%"}</div>
            </div>);
    }
}
```

Рисунок 2 – Исходный код компонента «Прогресс пользователя»

Для отображения прогресса пользователя, необходимо реализовать получение текущего прогресса пользователя с сервера и сохранение его в состояние. Для этого необходима реализация редьюсера (сохранение полученных данных) и создателя действий (для запроса к серверу). На рисунках 3,4 показан пример асинхронного создателя действий для организации взаимодействия с сервером, а на рисунке 5 приведен пример редьсера. При обращении к вспомогательной функции userObjectProcessRequest в глобальном состоянии установится признак начала запроса и на UI отобразится значок загрузки, после завершения запроса, флаг сбросится и данные, пришедшие с сервера попадут в редьюсер, который вернет новый объект, созданный на основе пришедших с сервера данных.

```
export const getUser = (doneCallback?: (param: UserData) => void) =>
    (dispatch, getState: () => AppStore) => {
        const state = getState();

        const getUserRequest = () =>
            configureGetFetch(state.clientConfig.urls.serverUrl + state.clientConfig.urls.getUserInfo, state.socialNetworkData);

        return userObjectProcessRequest(getUserRequest, dispatch)
            .then(doneCallback)
    }
```

Рисунок 3 – Пример асинхронного создателя действий

```
export function processRequest<T>(requestFunction: () => Promise<Response>, dispatch: Dispatch<any>, successAction: (t: T) => any) {
    dispatch(requestStarts());

    return requestFunction()
        .then(response => {
            return response.json()
        })
        .then((json: T | ErrorResponse) => {
            if ((<ErrorResponse>json).errorMessage) {
                dispatch(requestFailed((<ErrorResponse>json).errorMessage));
                throw new Error((<ErrorResponse>json).errorMessage)
            }
            else {
                dispatch(successAction((<T>json)));
                return <T>json;
            }
        })
        .catch(error => {
            dispatch(requestFailed(error.message));
            throw error;
        });
```

Рисунок 4 – Вспомогательная функция, организующая асинхронное обращение к серверу

```
import { UserData } from '../StateTypes';
import { ActionTypesEnum, UserObjectRequestSucceeded, } from '../constants';

const UserDefaultState: UserData = {
    id: "",
    balance: 0,
    level: 0,
    levelPoints: 0,
    pointsToNextLevel: 0,
    isVip: false,
    ticketId: null,
    isDailyBonusAvailable: false,
    lengthOfSeriesDailyBonus: 0,
    spinsCount: 0,
    nextFreeSpinDate: ""
}

export default function userDataReducer(state = UserDefaultState, action: UserObjectRequestSucceeded): UserData {
    switch (action.type) {
        case ActionTypesEnum.USER_OBJECT_REQUEST_SUCCEEDED:
            return action.result
        default:
            return state
    }
}
```

Рисунок 5 – Пример редьюсера

Redux спроектирован для использования отдельно от React, поэтому для их совместного использования нужно использовать пакет React-Redux. Также для облегчения взаимодействия и построения расширяемой архитектуры предполагается разделение на контейнерные и презентационные компоненты. Презентационные компоненты описываются с помощью React. Внутреннее состояние реализуется в Redux, поэтому в компонентах React

желательно не использовать внутреннее состояние, только входные параметры (props), которые описывают UI и предоставляют функции обратного вызова для взаимодействия с пользователем. Контейнерные компоненты чаще всего сгенерированы с использованием вспомогательных функций, их роль – отображение части глобального состояния всего приложения на входные данные для компонента (props), а также внесение изменения в глобальное состояние с помощью обратных вызовов функции dispatch (которая вносит изменения в глобальное состояние).[3] Такая архитектура позволит переиспользовать презентационные компоненты, т.к. они отделены и независимы от глобального состояния, а также позволит перезапускать механизм рендеринга точечно, только для тех компонентов, которые подписаны на нужные части глобального состояния. На рисунке 6 представлена схема презентационных, контейнерных компонентов и глобального состояния приложения.

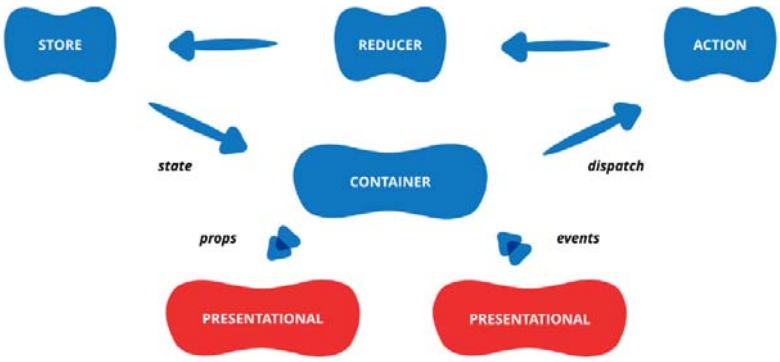


Рисунок 6 – Взаимодействие презентационных и контейнерных компонентов

В рассматриваемом примере требуется отобразить элемент глобального состояния, хранящий объект пользователя для входных данных для компонента, отображающего прогресс пользователя. Для этого используется функция connect, которая генерирует контейнерный компонент на основе двух функций mapStateToProps и mapDispatchToProps. MapStateToProps описывает, как отображается глобальное состояние приложения на входные данные для заданного компонента, а mapDispatchToProps используется для отображения на входные данные компонента вспомогательной функции dispatch, которая пересылает действий в редьюсеры. На рисунке 7 показано использование функции connect для генерации контейнерного компонента HeaderLeftPanelContainer на основе презентационного компонента HeaderLeftPanel. Теперь осталось разместить контейнер в нужном месте дерева компонентов, и при обновлении глобального состояния приложения, он получит новые данные и вызовет механизм рендеринга у презентационного компонента. В результате UI и глобальное состояние приложения останутся согласованными.

```
import { connect } from "react-redux";
import { AppStore } from "../StateTypes";
import { HeaderLeftPanel, HeaderLeftPanelProps } from "../components/HeaderLeftPanel";

const mapStateToProps = (state: AppStore) => ({
    level: state.userData.level,
    levelPoints: state.userData.levelPoints,
    pointsToNextLevel: state.userData.pointsToNextLevel
})

export const HeaderLeftPanelContainer = connect<HeaderLeftPanelProps>(mapStateToProps)(HeaderLeftPanel);
```

Рисунок 7 – Пример контейнерного компонента

Представленная архитектура позволяет разделить логику и взаимодействие компонентов на слои. Презентационные компоненты не содержат логики, контейнерные компоненты содержат обращения к создателям действий, которые в свою очередь содержат в себе обращения к серверу. Использование парадигмы функционального программирования облегчает тестирование, и отделяет все побочные эффекты. Построение UI с помощью композиции позволяет создавать переиспользуемые компоненты.

Список литературы

1. React documentation [Электронный ресурс]. – Режим доступа: https://reactjs.org/docs/
2. Redux documentation [Электронный ресурс]. – Режим доступа: https://redux.js.org/
3. Presentational and Container Components [Электронный ресурс]. – Режим доступа: https://medium.com/@dan_abramov/smart-and-dumb-components-7ca2f9a7c7d0

СИСТЕМА АВТОМАТИЗАЦИИ РЕГРЕССИОННОГО ТЕСТИРОВАНИЯ

Бочарова Е.С. – магистрант
Алтайский государственный технический университет (г. Барнаул)

Актуальность

На сегодняшний день мало кто сомневается в целесообразности проведения процесса тестирования разрабатываемых программных продуктов. Программные продукты на рынке становятся все более и более сложными как технически, так и функционально.

Как известно, наибольшая доля времени от тестирование приходится на регрессионное тестирование. Тестирование финансовых систем требует проводить много сложных расчетов при проверке функционала. В такой ситуации увеличивается риск пропустить ошибку. Поэтому встает вопрос об автоматизации регрессионного тестирования. Автоматизация поможет сократить рутинную работу персонала и свести к минимуму человеческий фактор.

В ходе автоматизации выполняются следующие задачи:

- анализ объектов автоматизации;
- выбор средств автоматизации тестирования;
- определение автоматизируемых компонент;
- разработка архитектурного решения;
- разработка и отладка скриптов;
- проведение регрессионного тестирования;
- обучение специалистов проектных команд работе со средствами автоматизации, а также практикам эксплуатации разработанной инфраструктуры тестирования. [3, 4].

Тема автоматизации довольно популярна и количество инструментов для ее реализации быстро растет, но подобрать оптимальный набор инструментов, который бы отвечал всем требованиям и нуждам компании и при этом не приводил к лишним затратам, зачастую оказывается не простой задачей.

Предлагаемое решение

В ходе выполнения работы, предполагается выполнить все перечисленные выше этапы и построить комплексную систему автоматизации тестирования на проекте, а так же покрыть автотестами один из сервисов компании.

Рассмотрим ряд существующих инструментов и методов автоматизации:

1. Unified Functional Testing (UFT) – предоставляет полный набор функций для тестирования API, веб-сервисов, а также для тестирования графического интерфейса десктопных, мобильных и веб-приложений на всех существующих платформах. UFT использует Visual Basic Scripting Edition, который может пригодиться для записи информации о выполненном тестировании, а также для управления объектами. UFT интегрирован с Mercury Business Process Testing и Mercury Quality Center. Инструмент поддерживает CI с помощью интеграции с инструментами CI, такими как Jenkins, что является огромным плюсом.

2. IBM Rational Functional Tester – это платформа для управляемого данными тестирования функциональности и регрессии ПО. Она поддерживает возможность тестирования широкого спектра приложений, написанных на различных языках программирования, таких как: .Net, Java, SAP, Flex и Ajax. RFT использует Visual Basic .Net и Java в качестве языков сценариев. RFT имеет уникальную функцию – Storyboard testing.

3. TestComplete – является эффективным инструментом для тестирования десктопных, мобильных и веб-приложений. TestComplete поддерживает различные языки сценариев, такие как: JavaScript, VBScript, Python и C ++ Script. TestComplete обладает схожей с UTF функцией распознавания объектов GUI. Данный инструмент также интегрируется с Jenkins в течение CI-процесса.

4. Tricentis Tosca – это модельно-ориентированный инструмент автоматизации тестирования, который предоставляет довольно широкий набор функций для непрерывного тестирования, включая тестирование с последующим выведением данных, их анализом и интеграцией для поддержки гибких методологий программирования и DevOps-методологий.

5. Robot Framework — это фреймворк для автоматического тестирования с открытым исходным кодом, в котором реализуется подход тестирования на основе ключевых слов для приемочного тестирования и разработки через приемочное тестирование (ATDD). Robot Framework предоставляет возможность решения различных задач автоматизации тестирования. Инженеры-тестировщики могут использовать Robot Framework в качестве фреймворка для автоматического тестирования не только веб-приложений, но для приложений под Android и iOS [5, 6].

Для уменьшения стоимости реализации, упрощения эксплуатации и дальнейшей поддержки автоматизации тестирования, предполагается использовать часть инструментов, которые уже активно применяются в организации.

Реализацию проекта планируется осуществлять с помощью следующего набора инструментов:

- язык разработки – C#;
- среда разработки – Visual Studio 2017;
- система контроля версий Git;
- система непрерывной интеграции – Jenkins;
- фреймворк для сборки и запуска тестов – NUnit;
- система отчетности Allure Framework.

Выбор представленных выше инструментов обусловлен следущим:

1. API для всех решений, разрабатываемых на проекте, написано на C++ и C#, поскольку C# является более простым в изучении, есть возможность обучить тестировщиков в более короткие сроки;

2. Visual Studio 2017, Git и Jenkins на текущий момент активно используются в компании, к тому же Jenkins имеет множество плагинов, в том числе для автоматизации тестов, а так же дает возможность автоматически запускать тесты, устанавливать новые сборки приложений на тестовое окружение, осуществлять проверку code style и пр.; [2]

3. NUnit – интегрирован с C#, имеет понятную структуру тестов и множество полезных встроенных функций;

4. Allure Framework – отчеты отображают информацию в понятном виде и для тестировщиков, и для менеджеров проекта. К ним можно прикладывать скриншоты, логи и любые другие файлы. Allure позволяет разбивать сложные тесты на шаги, включать информацию о параметрах тестов и тестового окружения. Тесты в Allure можно привязывать к тикетам в трекере. Отчеты генерируются с помощью консольной утилиты или плагинов для систем непрерывной интеграции и сборки проекта. В репозитории Allure есть готовые плагины и адаптеры для Jenkins, MSTest, Maven, PyTest, Spock, NUnit, Codeception, Cucumber, Atlassian Bamboo, ScalaTest и других популярных решений для разных языков программирования [1].

Построение процесса автоматизации будет состоять из нескольких обязательных этапов:

- анализ и изучение тестируемого функционала;
- написание сценария тестирования (тест-кейсы);
- написание авто тестов для покрытия функционала выбранного приложения;
- коммит написанного кода в систему управления версиями Git;
- запуск авто тестов в Jenkins;
- после окончания процесса проверки происходит построение отчета в Allure.

Данный подход, как и любой другой, требует постоянной поддержки тестов в актуальном состоянии. Обычно в код тестов приходится вносить изменения после любой доработки тестируемого функционала.

Стоит отметить, что запуск и анализ результатов тестов сможет выполнить любой сотрудник, даже тот который не имеет знаний в области программирования.

Основной процесс тестирования смоделирован на рисунке 1.

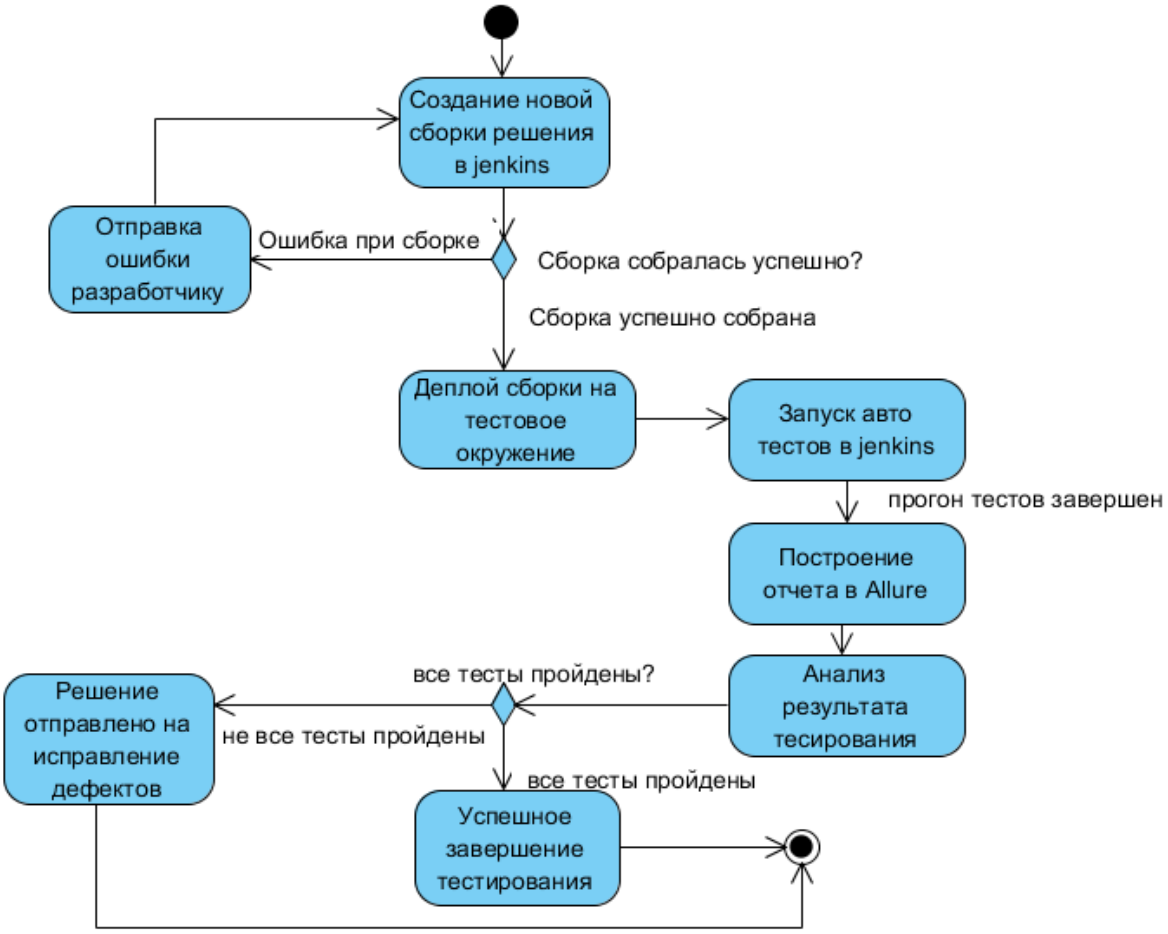


Рисунок 1 – Процесс тестирования функционала

По итогу реализации данного проекта, компания получает отлаженную систему автоматизации тестирования приложений и сервисов, тем самым будет сокращено время сотрудников, затрачиваемое на проведение регрессионного тестирования. Отчет о прохождении тестов будет доступен и понятен даже менеджеру проекта. Поскольку в компании уже используется большая часть инструментов выбранных для автоматизации тестирования, данную модель будет легко применять на практике.

Список литературы

1. Allure Test Report [Электронный ресурс] // allure.qatools.ru. – Режим доступа: http://allure.qatools.ru.
2. Jenkins User Documentation [Электронный ресурс] // jenkins.io. – Режим доступа: https://jenkins.io/doc/
3. Автоматизация регрессионного тестирования [Электронный ресурс] // Прогрессор. – Режим доступа: http://prgssr.ru/development/avtomatizaciya-regressionnogo-testirovaniya-css-2016.html
4. Автоматизация регрессионного тестирования [Электронный ресурс] // quadrium.ru. – Режим доступа: http://www.quadrium.ru/solutions/m-testing/m-t-r
5. Лучшие практики автоматизации тестирования обеспечения [Электронный ресурс] // GetBug. – Режим доступа: http://getbug.ru/luchshie-praktiki-avtomatizatsii-testirovaniya/

РЕАЛИЗАЦИЯ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ РАНЖИРОВАНИЯ ВЕБ-САЙТОВ ПО ВЕБОМЕТРИЧЕСТИМ ХАРАКТЕРИТСИКАМ

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

Обладание качественным представительством в сети Интернет является основой успешного продвижения бизнеса в сети. Для оценки эффективности работ по оптимизации контента сайтов необходимы формальные критерии, по которым можно сравнивать сайты друг с другом. Вебометрика — раздел информатики, в рамках которого исследуются количественные аспекты конструирования и использования информационных ресурсов, структур и технологий применительно к Всемирной паутине. Вебометрические характеристики являются такими формальными критериями, на основе которых можно транжировать веб-сайты.

Обзор существующих решений

Из существующих решений самым известным является ранжирование по методологии компании Cybermetrics Lab: для сайта вычисляется четыре характеристики [1]:

1. Visibility — количество ссылок с других сайтов (RankV)
2. Size — общее количество страниц сайта (RankS)
3. Rich files - количество полноценных текстовых файлов (RankV)
4. Scholar — количество размещенных на сайте статей и их цитирований (RankSc)

Перечисленные метрики вычисляются путём запроса к нескольким поисковым машинам. Далее, вычисляется вебометрический ранг WR, по которому и идёт ранжирование:

Метод Cybermetrics Lab не использует российские поисковые машины, поэтому в России разработаны собственные проекты по созданию вебометрических рейтингов. В большинстве случаев данные проекты дополняют список используемых поисковых машин Яндексом [2] [3] и Bing [4], а также изменяют коэффициенты при индикаторах при вычислении вебометрического ранга.

Поиски более совершенных методов ранжирования идут до сих пор. Составление системы метрик-индикаторов, позволяющей наиболее объективно отранжировать сайты является очень непростой задачей.

Предлагаемое решение

В данной работе предлагается реализация приложения, позволяющего по указанным адресам сайтов вычислять их вебометрические характеристики. Приложение скачивает контент указанного сайта до определенной глубины, вычисляет вебометрические характеристики, позволяет легко добавлять новые алгоритмы вычисления вебометрических характеристик. Для всех обработанных сайтов приложение визуализирует их характеристики и дает возможность сортировки сайтов по взвешенной сумме всех вычисленных характеристик. Главное окно системы представлено на рисунке 1.

Разработанная система ранжирования веб-сайтов по вебометрическим характеристикам состоит из нескольких модулей:

1. Модуль информации о сайте — позволяет для заданного сайта сохранять его файлы в файловой системе, а также его произвольные числовые характеристики.
2. Модуль управления очередью задач — позволяет хранить и упорядочивать работы в порядке поступления.
3. Модуль управления демонами — запускает фоновые задачи, выполняющие различные функции и получающие задания на работу из очереди задач. Позволяет запускать или останавливать экземпляры одного демона для регулирования количества одновременно обрабатываемых задач.
4. Краулер сайтов — демон, который получает из очереди задач адрес сайта и рекурсивно скачивает контент сайта по указанному адресу. Сохраняет страницы сайта через модуль информации о сайте и добавляет в очередь задач задания на их анализ.
5. Анализатор сайтов — демон, который получает из очереди задач имя сайта и через модуль информации о сайте получает его содержимое и вычисляет значения вебометрических индикаторов.
6. Web-api — представляет программный интерфейс, через который с помощью http можно получить доступ к функциям системы.
7. Веб-фронтенд — позволяет посредством web-api получить наглядный интерфейс к написанной системе и наглядно просмотреть все происходящие в ней процессы.

Рисунок 1 - Скриншот главной страницы фронтенда

При разработке системы были применены следующие паттерны программирования:

1. Factory

Экземпляры демонов создаются простой фабрикой, которая создаёт их на основе переданного типа. При добавлении новых типов демонов изменяться будет только код фабрики, учитывающий все нюансы инициализации конкретного типа демонов. Код потребителей же меняться не будет.

2. Builder

Используется для формирования конкретного элемента очереди. Он позволяет создавать элементы нужного типа и задавать им дополнительные свойства

3. Facade

Модуль web-api, предоставляющий HTTP интерфейс для работы с системой, является фасадом данного продукта. Именно через него происходит взаимодействие внешних пользователей с нашей системой. Данный паттерн позволяет менять внутреннюю архитектуру системы, не изменяя контракта для потребителей.

4. Mediator

В представленной системе демоны скачивания и анализа ничего друг о друге не знают. Они имеют информацию лишь о модуле очередей. В данной архитектуре можно очень легко добавлять новые типы обработчиков, так как если есть некоторые зависимости между ними, они будут решаться в одном модуле очередей и код остальных демонов не будет затронут.

5. Abstract Factory

Для создания экземпляров анализаторов используется абстрактная фабрика. Демон получает список всех фабрик анализаторов. И далее создаёт их экземпляры

6. Chain of responsibility (Middleware)

Все созданные анализаторы компонуется в цепочку. После чего демон передаёт им информацию об очередном анализируемом файле, которая проходит по цепочке анализаторов, вычисляющих по ней свои метрики. Организация анализаторов в цепочку необходима для того, чтобы не читать один файл несколько раз.

7. Template Method

Был создан абстрактный класс TaskDaemon реализующий общую логику управления. Наследникам остаётся лишь определить, какие именно задания будут получаться из очереди и что делать при получении задания. Реализация данного шаблонного метода позволяет легче создавать новых демонов, не копируя инфраструктурный код при каждой модификации системы.

Дальнейшие возможности расширения системы:

Так как для разных модулей данной системы вычислительная нагрузка не всегда совпадает, то держать данные модули в одном процессе не всегда правильно. Данные модули можно разнести по разным микросервисам, перейдя на микросервисную архитектуру.

Данная архитектура позволит масштабировать каждую часть системы индивидуально, в зависимости от нагрузки на неё, регулируя количество её запущенных реплик. Также микросервисы усиливают степень зацепления и уменьшают связанность в виду того, что для внешней системы модуль предоставляет строго определенное (чаще всего REST) API.

Сервис очередей при этом можно сделать адаптером для существующего промышленного продукта по управлению очередями. Например, RabbitMQ или NSQ.

Сервис демонов при улучшении можно модифицировать так, чтобы отдельные демоны были не задачами на пуле потоков, а полноценными процессами. При этом появятся механизмы для полноценного разграничения ресурсов для процессов. Также сами анализаторы индикаторов можно сделать загружаемыми библиотеками. При такой организации подхода, для появления нового индикатора в системе достаточно будет перезапустить демонов-анализаторов, без перезапуска всей основной системы.

Список литературы

1. Печников А.А., Илюкевич О.Г. Рейтинг официальных web-сайтов университетов России и Финляндии: сравнительный анализ // Информационные ресурсы России. - Москва: Российское энергетическое агентство Министерства энергетики Российской Федерации. - 2008. - № 3. - С. 25-28. - ISSN 0204-3653.
2. Мазалов В.В., Печников А.А. О рейтинге официальных сайтов научных учреждений Северо-Запада России // Управление большими системами: сборник трудов. - 2009. - № 24. - С. 130-146.
3. Методика 2013 г. [Электронный ресурс]. ИПМИ КарНЦ РАН. - Режим доступа: http://webometrics-net.ru/section.php?id=30
4. Методика 2013 г. [Электронный ресурс]. Институт вычислительных технологий СО РАН. - Режим доступа: http://w.ict.nsc.ru/ranking/

РАЗРАБОТКА БИБЛИОТЕКИ ДЛЯ УПРАВЛЕНИЯ ФОНОВЫМИ ЗАДАЧАМИ НА ОСНОВЕ КОНЕЧНЫХ АВТОМАТОВ

Колпаков А.С. – студент, Старолетов С.М. – к.ф.-м.н., доцент,
Дашкевич И.А. – ведущий инженер-программист*
Алтайский государственный технический университет (г. Барнаул)
*АО "ПФ "СКБ Контур" (г. Екатеринбург)

Актуальность

В разработке при написании кода в рамках решения бизнес-задач в организации, в которой осуществлялось прохождение практики, часто встречаются задачи по долгой обработке некоторых данных от пользователя. Обработка обычно происходит в специальных фоновых процессах — демонах. При этом существуют несколько моделей обработки полученных данных:

- Однопоточная — вся обработка данных идёт в одном потоке конкретного демона. Преимуществом этого подхода является возможность написания любой логики обработки на языке программирования привычным способом, без использования всяких дополнительных средств. Минусами являются низкая скорость из-за низкого уровня параллелизма (если во входных данных есть большой массив данных, которые можно обработать независимо друг от друга, то он обрабатывается однопоточно) и низкая отказоустойчивость — в случае краха процесса демона теряется вся проделанная работа, а обеспечение сохранения промежуточного состояния и восстановление из него трудно и для каждого обработчика должно делаться индивидуально.
- Fork-join модель — обработчик данных может запускать некоторое количество дочерних обработчиков и обрабатывать их результаты. Данная модель универсальна и может выразить любой многопоточный процесс. Она позволяет обрабатывать массивы данных параллельно и имеет явно видимые точки (перед порождением дочерних обработчиков, каждый дочерний обработчик сохраняет свой результат, перед обработкой всех результатов), в которых можно сохранять состояние обработки, обеспечивая точки, с которых можно продолжить обработку в случае отказа узла вычислений. Минусом является трудность выражения в данной модели цепочек вычислений — приходится выражать их через порождение одного дочернего процесса, в результате чего растет вглубь дерево процессов.
- MapReduce — модель распределенных вычислений, состоящая из двух шагов. Map — предварительная обработка входных данных. Reduce — агрегация обработанных данных. Является частным случаем Fork-Join, но благодаря специализированности может предоставлять более удобные и оптимизированные под задачу средства.
- Автоматная модель — в демоне явно задекларированы состояния автомата и их обработчики, что позволяет выражать сложные графы (машины состояний с переходами) обработки данных.

В данный момент в компании имеется библиотека для написания демонов (программ, выполняющих фоновые задачи) на основе модели fork-join. Однако оказалось, что большая часть задач требует реализации автоматной модели обработки. Например, такая модель необходима при печати файлов электронной отчетности в pdf: для реализации печати

необходимо выполнить последовательность шагов, каждый из которых занимает определенное время, причём после каждого шага следующий может выбираться в зависимости от результата предыдущего. Такая логика обработки хорошо укладывается в понятие конечного автомата.

Обзор существующих решений

Из известных подходов существует фреймворк Hangfire, реализующий обработку фоновых задач для приложений на платформе .NET. Однако в нём нет возможности явно задавать состояния, есть только возможность запустить следующий обработчик из другого обработчика [1]. Но такая функциональность есть вообще в любой библиотеке и она не удобна.

Ещё из альтернатив можно отметить фреймворк Windows Workflow Foundation, который позволяет проектировать автоматы в специальном дизайнере [2]. Но этот дизайнер доступен только под Visual Studio, а многие разработчики используют альтернативную IDE для разработки под .NET, например, Jetbrains Rider. Также он не имеет встроенной поддержки очереди задач, через которую принимаются задачи на обработку.

И общим минусом двух вышеприведенных альтернатив является обеспечение персистентности. Windows Workflow Foundation и Hangfire сохраняют состояние только в определенный набор баз данных [3]. Однако в нашем проекте используется написанная для своих нужд очередь задач, а для хранения информации используется также собственная документоориентированная БД, что принуждает к написанию адаптеров для БД и очереди задач в случае использования данных проектов.

Постановка задачи

Таким образом, имеется необходимость в реализации библиотеки работы с фоновыми задачами, в основе которой стоит конечный автомат со следующими возможностями:

- Возможность задавать обработчики для каждого состояния автомата.
- Возможность переходить из одного состояния в другое в коде обработчика.
- Возможность для взаимных вызовов со старыми fork-join обработчиками (обработчик модели fork-join может в качестве подзадачи вызвать автоматный обработчик и наоборот).

Иногда для выполнения работы демону нужно обратиться к внешним сервисам, которые могут долго обрабатывать задание. Эти сервисы имеют определенный интерфейс взаимодействия, позволяющий поставить задачу на выполнение, получать статус выполнения задачи (ждет выполнения, выполняется, выполнена, завершена с ошибкой) и получать результат.

- Необходимо реализовать обобщение для интерфейсов внешних обработчиков для обеспечения прозрачной интеграции внешних сервисов в качестве подзадачи (поллинг) для модели fork-join и автоматной модели.

Текущее состояние работы

В существующей библиотеке имеется возможность декларации обработчиков задач с Main методом, в котором можно порождать подзадачи и Join методами для обработки результатов каждого форка. Для каждого обработчика сохраняется входные данные, его состояние (список дочерних подзадач, статус выполнения) и его результат.

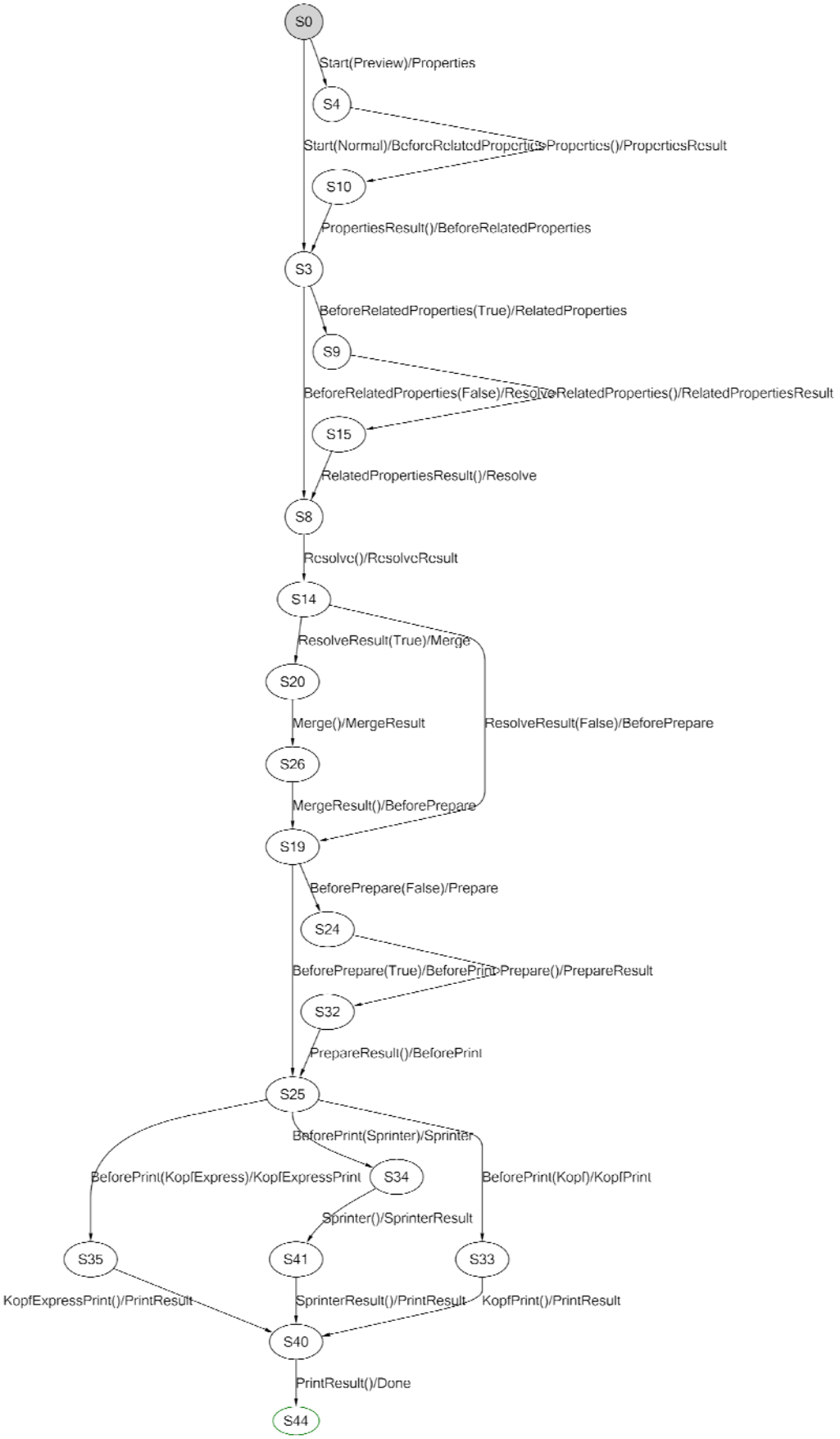


Рисунок 1 - Пример автомата поведения печати документа

План работы

В данный момент реализуется дописывание существующей библиотеки следующими функциями:

- Возможностью иметь в задаче несколько Main методов — для каждого состояния свой обработчик.
- Реализация методов инициации смены состояния.
- Сохранение в информацию о задаче текущего состояния автомата.
- Поллинг внешних сервисов с гибкой настройкой интервалов запросов.

При выполнении данной работы планируется перевести сервис печати на использование автоматной модели, что позволит сильно упростить код обработчиков и повысит надежность.

Так как при разработке применяется автоматный подход, то для тестирования может применяться тестирование по модели (Model Based Testing, MBT) и программное средство SpecExplorer для автоматической генерации модульных тестов. В настоящие время создана модель системы (обозревание модели представлено на рисунке 1), по которой сгенерированы тесты.

В дальнейших планах есть перевод остальных демонов, для которых это необходимо, на новую модель работы.

Список литературы

1. Hangfire.io Documentation [Электронный ресурс]. - Режим доступа: https://www.hangfire.io/features.html
2. Developing Applications with the Workflow Designer [Электронный ресурс]. - Режим доступа: https://msdn.microsoft.com/en-us/library/dd489396.aspx
3. Windows Workflow Persistence Services [Электронный ресурс]. - Режим доступа: https://msdn.microsoft.com/en-us/library/ms734764(v=vs.90).aspx
4. Spec Explorer [Электронный ресурс]. - Режим доступа: https://msdn.microsoft.com/en-us/library/ee620411.aspx
5. Основные технологии и методы тестирования [Электронный ресурс]. // webhamster.ru. – Режим доступа: https://webhamster.ru/mytetrashare/index/mtb0/1410498780s5p8df5nan

АНАЛИЗ И ОПТИМИЗАЦИЯ АЛГОРИТМА ПАРАЛЛЕЛЬНЫХ ЦЕПОЧЕК ДЛЯ РЕАЛИЗАЦИИ КОРНЕВОЙ РЕДУКЦИИ НА РАСПРЕДЕЛЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ

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

Введение

Параллельные алгоритмы и программы для распределенных вычислительных систем (ВС) преимущественно разрабатываются в модели передачи сообщений (message-passing), в рамках которой ветви программы (процессы) взаимодействуют посредством передачи сообщений по каналам межмашинных связей. Среди основных схем обменов значительное место по частоте использования и приходящемуся на них суммарному времени выполнения занимают коллективные операции обменов информацией (групповые, глобальные, collective communications) [1,2]. Такие операции делятся на корневые и некорневые. В них участвуют все или подмножество всех ветвей параллельной программы (в стандарте MPI такие подмножества образуют группы процессов коммуникаторов). К корневым (rooted) относятся

трансляционная передача (“один–всем”, one-to-all) и коллективный прием (“все–одному”, all-to-one). Примерами некорневого обмена (unrooted) служат обмены типа “каждый–всем” (all-to-all). Для широкого класса параллельных алгоритмов время выполнения коллективных операций является критически важным и определяет их масштабируемость [3].

В данной работе рассматривается реализация алгоритма k параллельных цепочек [4], который характеризуется линейной зависимостью времени выполнения от числа процессов.

Сокращение времени информационного обмена

Работы, ориентированные на сокращение времени информационных обменов в распределенных ВС, ведутся по двум направлениям:

1) создание специализированных коммуникационных сетей, ориентированных на минимизацию времени реализации основных схем информационных обменов (Cray Gemini/Aries, IBM PERCS, Fujitsu Tofu, TH Express-2, МВС-Экспресс, Ангара);
2) разработка масштабируемых алгоритмов реализации коллективных операций на базе примитивов двухсторонних обменов (send/recv, в отрыве от конкретной коммуникационной технологии) [5,6].

В данной работе приводится пример реализации по второму подходу: оптимизация алгоритмов корневой редукции на базе примитивов двусторонних обменов.

Корневая редукция (reduction, reduce) — это коммуникационно-вычислительная коллективная операция, которая широко используется при разработке параллельных алгоритмов и программ для ВС с распределенной памятью [7]. Данная операция реализует комбинирование операндов в памяти параллельных ветвей программы при помощи заданной бинарной ассоциативной и, возможно, коммутативной операции.

Для реализации корневой редукции на распределенных ВС разработано значительное количество алгоритмов [6,7]. В их основе лежит логическая организация процессов программы в графы (деревья) различных видов и параллельное вычисление частичных результатов операции, которые передаются между ветвями посредством двусторонних обменов сообщениями (send/recv). Наибольшее распространение получили алгоритмы биномиального дерева (binomial tree), Р. Рабенсейфнера (R. Rabenseifner) [7], параллельных цепочек (k-chain), бинарного и плоского деревьев (binary, flat/linear tree).

Метод k параллельных цепочек (k-chain algorithm)

В центре внимания данной работы — алгоритм k параллельных цепочек (k-chain algorithm), который характеризуется линейной зависимостью времени выполнения от числа процессов. Вопрос о выборе значения k является открытым и, как правило, решается эмпирическим путем [4]. В данной работе используется построенное в модели параллельных вычислений LogP аналитическое выражение [6] времени выполнения алгоритма как функция от числа процессов и количества цепочек. Это выражение позволило найти оптимальное значение числа k параллельных цепочек, при котором алгоритм характеризуется минимальным в модели LogP временем выполнения.

На основе найденного значения k в данной работе в стандарте MPI реализован алгоритм с оптимальным числом параллельных цепочек [7]. Для сокращения времени ожидания корневым процессом результатов частичных редукций существует алгоритм с адаптивным числом параллельных цепочек. Зависимость времени выполнения созданных алгоритмов с оптимальным адаптивным числом цепочек от числа процессов имеет порядок роста $O(\sqrt{P})$, что эффективнее по сравнению с линейным $O(P)$ временем выполнения исходного алгоритма с фиксированным числом цепочек.

Модель LogP — это математическая модель ВС с распределенной памятью, в которой параллельные процессы взаимодействуют посредством двусторонних обменов короткими

сообщениями фиксированного размера [12]. Модель не учитывает структуру коммуникационной сети ВС и отражает лишь показатели производительности каналов связи между процессорами. В модели LogP время t передачи сообщения от одного процессора другому выражается как t = 2o+L.

Алгоритм k параллельных цепочек организует процессы в k цепочек (конвейеров), которые передают свои результаты корневому процессу с номером root, где k — это фиксированный параметр алгоритма. Например, в библиотеке Open MPI по умолчанию значение k = 4 (подсистема коллективных операций tuned) [8]. Если число процессов P − 1 не кратно значению k, то остаток (P −1)%k процессов распределяется по первым (P −1)%k цепочкам, в каждую добавляется по одному процессу. На рис. 2 первые две цепочки получили по одному дополнительному процессу.

Алгоритм применим только в том случае, если из P − 1 процессов можно организовать k непустых цепочек. При k = 1 алгоритм параллельных цепочек сводится к конвейерному алгоритму (pipeline reduce), а при k = P − 1 — к алгоритму плоского дерева (flat/liner tree). Будем называть цепочку короткой, если она содержит (P − 1)/k процессов. В противном случае назовем цепочку длинной.

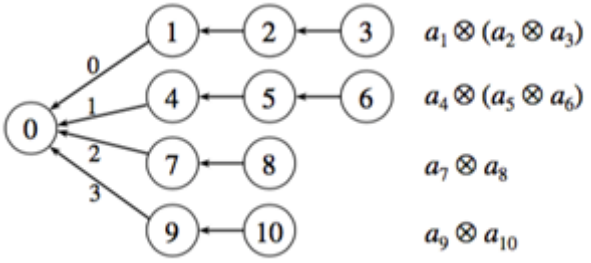


Рисунок 2 - Дерево обменов алгоритма параллельных цепочек: P = 11, k = 4, root = 0
(стрелками показаны направления передачи сообщений,
справа результаты редукций в цепочках)

Корневой процесс выполняет k приемов результатов цепочек и столько же локальных редукций. Сначала выполняется прием результатов от длинных цепочек, а затем от коротких. Некорневые процессы по своему номеру определяют свое положение в цепочке — номера следующего и предшествующего процессов.

В ходе работы выбран и реализован следующий вариант оптимизации - выбор оптимального числа цепочек.

Найдем число k цепочек, при котором алгоритм характеризуется наименьшим в модели LogP временем выполнения. Запишем время алгоритма параллельных цепочек как функцию от параметра k:

$$t(k)=(P-1)/k+1(2o+L+m\gamma)+(k-2)\max\{o+m\gamma,g\}+o+m\gamma.$$

Здесь P — количество процессоров в системе,

o — промежуток времени (overhead), в течение которого процессор занят передачей или приемом сообщения (в течение этого времени процессор не может выполнять другие операции);

g — минимальный интервал времени (gap) между последовательными передачами или приемами сообщений (1/g — пропускная способность канала связи, доступная процессору);

Выводы

Алгоритм реализован в стандарте MPI в качестве экспериментальной проверки метода Для каждого значения k рассчитывалось среднее время выполнения алгоритма по результатам 10 запусков. Рекомендуемое в [6] значение k = 7 обеспечивает время выполнения алгоритма, близкое к оптимальному.

Список литературы

1. Хорошевский В.Г. Распределенные вычислительные системы с программируемой структурой // Вестник Сиб- ГУТИ. -2010. - т. 10 - No 2. - С. 3–41.
2. Hoefler T., Moor D. Energy, memory, and runtime tradeoffs for implementing collective communication operations // Journal of Supercomputing Frontiers and Innovations. - 2014. vol.1. – N. 2. - P. 58–75.
3. Balaji P., Buntinas D., Goodell D., Gropp W., Hoefler T., Kumar S., Lusk E., Thakur R., Träff J. MPI on millions of cores // Parallel Processing Letters. - 2011. – 21. – N. 1. - P. 45–60.
4. Alverson R., Roweth D., Kaplan L. The Gemini System Interconnect // Proc. 18th IEEE Symposium on High Performance Interconnects. Washington, DC: IEEE Press. - 2010. - P. 83–87.
5. Chen D., Eisley N.A., Heidelberger P., Senger R., et al. The IBM Blue Gene/Q interconnection network and message unit // Proc. 2011 International Conference for High Performance Computing, Networking, Storage and Analysis. New York: ACM Press. - 2011. doi 10.1145/2063384.2063419.
6. Culler D., Karp R., Patterson D., et al. LogP: towards a realistic model of parallel computation // ACM SIGPLAN Notices. - 1993. – Vol.28. – N. 7. - P. 1–12.
7. Курносов М.Г. Анализ и оптимизация алгоритма параллельных цепочек для реализации корневой редукции на распределенных вычислительных системах // Вычислительные методы и программирование. - 2016. - Т. 17. - С. 218–327.
8. Worsch T., Reussner R., Augustin W. On benchmarking collective MPI operations // Lecture Notes in Computer Science. Vol. 2474. Heidelberg: Springer. - 2002. - P. 271–279.

АНАЛИЗ ЦЕЛЕСООБРАЗНОСТИ ПРИМЕНЕНИЯ ТЕХНОЛОГИИ БЛОКЧЕЙН ДЛЯ РАЗРАБОТКИ СИСТЕМЫ ПОДТВЕРЖДЕНИЯ ПОДЛИННОСТИ НОРМАТИВНЫХ ДОКУМЕНТОВ

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

Подделка документов, изготовление или сбыт поддельных документов весьма актуальная проблема для современного уголовного права. Согласно данным, представленными порталом правовой статистики МВД Российской Федерации, количество преступлений, предусмотренных статьей 327 Уголовного кодекса Российской Федерации, с каждым годов увеличивается, о чем говорят доклады о статистики преступлений из [1]. В век компьютерных технологий появляются новые, все более эффективные способы подделки официальных документов, штампов, печатей, бланков, государственных наград и их внедрения в оборот. В соответствии с Российским законодательством официальные документы выступают в качестве предметов преступных посягательств, а также орудий или средств совершения преступления. При этом использование поддельных официальных документов в качестве орудий или средств совершения опасных преступлении не всегда

отражено в содержании ряда статей УК РФ. К ним относятся, например, присвоение и растрата вверенного имущества, мошенничество, пересечение Государственной границы Российской Федерации без действительных документов, незаконная банковская деятельность, мошенничество в сфере кредитования. Данная проблема является актуальной на данный момент, а с развитием технологий будет становиться еще более актуальнее.

В данной статье предлагается оценка подхода по решению данной проблемы с использованием технологии блокчейн.

Технология Блокчейн

Блокче́йн (англ. blockchain или block chain) — выстроенная по определённым правилам непрерывная последовательная цепочка блоков (связный список), содержащих информацию. Чаще всего копии цепочек блоков хранятся на множестве разных компьютеров независимо друг от друга. В [2] приводится простое и понятное описание технологии. Говоря простым языком, блокчейн – это журнал с фактами, реплицируемый на несколько компьютеров, объединенных в сеть равноправных узлов (P2P). Фактами может быть что угодно, от денежных операций и до подписания контента. Члены сети — анонимные лица, называемые узлами. Все коммуникации внутри сети используют криптографию, чтобы надежно идентифицировать отправителя и получателя. Когда узел хочет добавить факт в журнал, в сети формируется консенсус, чтобы определить, где этот факт должен появиться в журнале; этот консенсус называется блоком. Распределенная природа баз данных Blockchain делает взлом хакерами практически невозможным, поскольку для этого им нужно одновременно получить доступ к копиям базы данных на всех компьютерах в сети. Технология также позволяет обезопасить личные данные, так как процесс хеширования необратимый. Если даже оригинальный документ или транзакция будут в дальнейшем изменены, то в результате они получат другую цифровую подпись, что сигнализирует о несоответствии в системе [3].

На текущий день технология блокчейн активно развивается. Появляются готовые решения от крупных поставщиком программных продуктов в сфере IT, таких как, например, IBM или Microsoft.

Использование технологии блокчейн в рамках решения проблемы

Перейдем к непосредственному решению поставленной в начале данной статьи проблеме. Итак, имеется распределенная база данных, доступ к которой возможно получить из любой точки мира с помощью глобальной сети интернет. В качестве узлов пиринговой сети выступают ведомства или организации, которые выдают какие-либо нормативные документы государственного образца. Транзакция подтверждает факт выдачи такого документа конкретному лицу. Отметим преимущества такой системы:

1. Транзакцию невозможно подделать, что упростит проверку выдачи конкретного документа частному лицу.
2. Обычно блокчейн используется для обеспечения анонимности, но в данной ситуации напротив позволит выявить автора транзакции в сети, что позволит контролирующим органам быстро реагировать на нарушение закона и вычислять правонарушителей.
3. Все что нужно для работы такой системы - это пользовательский интерфейс, доступ к сети и пространство для хранения копии распределенной базы данных. Не нужны большие вычислительные мощности, что сократит затраты на интеграцию и поддержку.
4. Предлагаемую пиринговую сеть вполне возможно сделать приватной, что исключить нежелательный доступ к ней извне.

Как и любая технология, блокчейн имеет и свои недостатки. Отметим некоторые из них и укажем на возможные решения:

P2P-сетям, как и прочим распределенным системам, приходится решать очень сложную проблему информатики: разрешение конфликтов, или согласование. Реляционные базы данных предлагают ссылочную целостность, но такой особенности нет в распределенной системе. Если два несовместимых факта прибывают в одно и тоже время, система должна иметь правила для определения того, какой факт считать правильным. В P2P сетях, два факта отправленные примерно в одно время могут прибыть в разном порядке в удаленные узлы. Тогда как всей сети согласовать какой же факт пришел первым? Чтобы гарантировать целостность в P2P сети, вам нужен способ согласования порядка фактов. Вам нужна система консенсуса.

Алгоритмы консенсуса для распределенных систем это очень активное поле для исследований. Для решения данной проблемы в рамках представленной системы предлагается алгоритм delegated proof-of-stake. DPoS (Delegated proof-of-stake) — это алгоритм достижения консенсуса в децентрализованой среде, который является альтернативой консенсусам PoW (Bitcoin proof-of-work) и PoS (Peercoin или NXT proof-of-stake). По данным [4], DPoS был разработан в 2014 году в рамках проекта Graphene и впервые был задействован в проекте Bitshares, позже в проекте Steemit.

Принцип работы алгоритма DPoS

Принцип работы DPos наиболее хорошо и полно описан в [5]. Если коротко сформулировать основной принцип работы DPoS, он будет выглядеть таким образом: разделение голосующих и валидирующих участников. В итоге, участники сети, которые имеют право голоса в системе не являются при этом валидаторами транзакций. Таким образом одно подмножество участников выбирает другое подмножество, которое в свою очередь будет формировать блоки. Условия, в которых работает данный алгоритм консенсуса, отличаются от условий в которых работает PoW и PoS. А именно, валидаторам необходимо раскрыть свои личности, что скорее даже плюс в рамках предлагаемого ПО, и заявить о готовности бесперебойно поддерживать работу полноценного узла сети, своевременно выполнять верификацию транзакций и формировать новые блоки. Консенсус на основе модифицированного proof-of-stake работает по такому правилу, что каждый пользователь по желанию может выставить свою кандидатуру на пост верифицирующей рабочей станции (узла валидатора). Потом среди всех пользователей проводится голосование за кандидатов, где вес каждого голоса определяется суммой активов голосующего. По результатам голосования выбирается N (натуральное число, которое выбирает комьюнити, обычно 20-50) кандидатов, которые получают право формировать новые блоки транзакций. Правила протокола гарантируют корректное принятие решений, если большая часть активов, принимающих участие в голосовании контролируется честными пользователями.

Выбранные в результате голосования валидаторы перемешиваются псевдослучайным образом, образуя очередь. Перемешивание выполняется специальным алгоритмом, так что предсказать очередь заранее невозможно, но она получается одинаковая для всех честных участников сети. Далее выделяется период времени, за который каждый из валидаторов должен сформировать один блок соответственно очереди. Причем, каждому валидатору в этом периоде выделяется строго ограниченный интервал времени (обычно 1 секунда). Либо валидатор успевает проверить новые транзакции и сформировать новый блок на основании предыдущего, либо эту работу сделает уже следующий в очереди валидатор. После завершения периода времени валидаторы снова перемешиваются и образуют новую очередь. Логично, что в качестве валидаторов могут быть представлены контролирующие законодательство органы.

Другие объективные проблемы, которые стоит отметить одновременно - это скорость проведения транзакции и стремительный рост размера репликации базы данных, ведь храниться она должна на всех узлах. С данной проблемой отлично справляется уже

отмеченный ранее проект Graphene. Протокол Graphene – это новый способ снижения пропускной способности за счет использования Bloom фильтров и IBLTs.

Протокол Graphene

Из статьи [6] известно, что проект протокола «Графен» осуществляется совместными усилиями ученых университета Массачусетса, а также известного биткойн-разработчика Гэвина Андрессена. По словам Кена Кода, который занимается разработкой деловых моделей, в тестовом режиме Graphene показал 10682 транзакции в секунду. Также создатели «Графена» считают, что данный протокол более эффективно обрабатывает и распространяет новые блоки – значительно лучше и быстрее, чем Compact Blocks и Xtreme Thinblocks. «Графен» создает блоки небольшого размера: для сравнения, размер блока в для Xtreme Thinblocks составляет 17.5 кб, а у «Графена» эта цифра составляет 2,6 кб. Проект имеет открытый исходный код и MIT лицензию, что позволяет использовать его в рамках реализации предлагаемой системы.

Заключение

Приведем схему работы системы с точки зрения бизнес-процессов (рисунок 1).

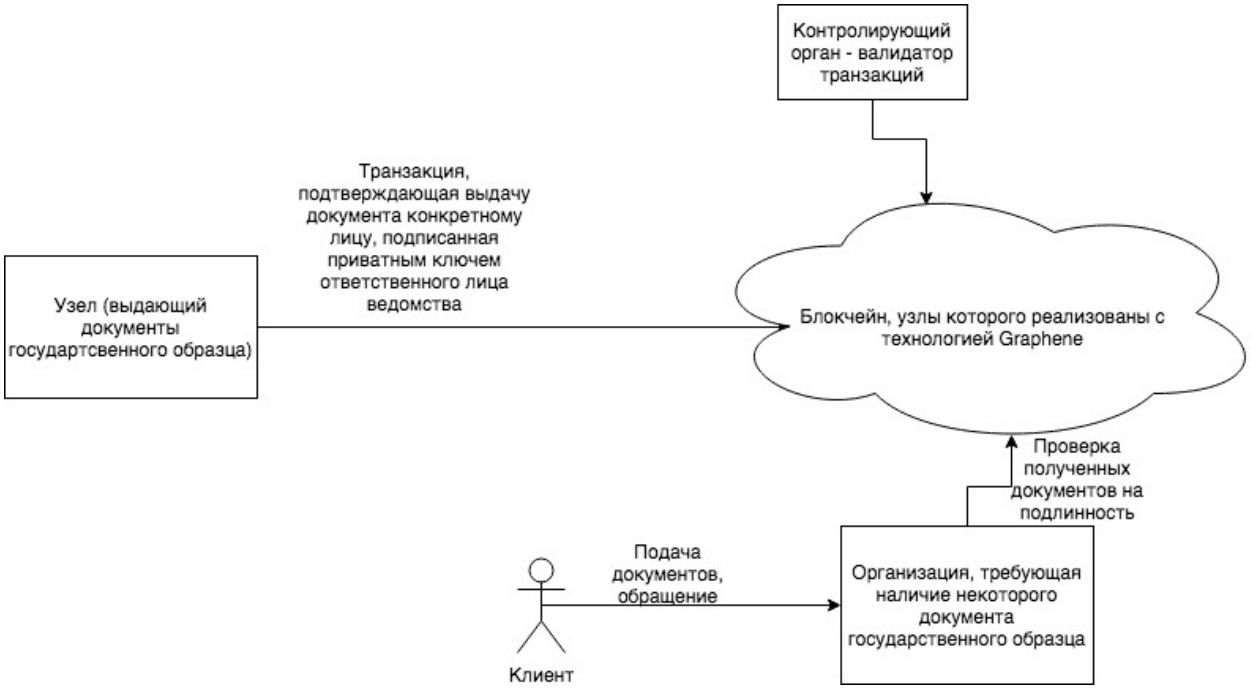


Рисунок 1 - предлагаемое решение

Итоговая система позволит решить проблемы с подделкой документов на нескольких уровнях: невозможно будет отправить транзакцию, не имея доступа к системе, а в случае обнаружения мошенничества со стороны компаний, выдающих документы государственного образца по записанной транзакции поиск виновников будет упрощен. Запись транзакций в блокчейн будет процедурой довольно прозрачной для пользователя, благодаря интерфейсам взаимодействия. Вся история транзакций доступна как любому участнику пиринговой сети, так и организациям использующим систему в рамках проверки полученного документа на подлинность. Для работы системы не требуется специфического оборудования и больших вычислительных мощностей.

Список литературы

1. Сайт МВД РФ [Электронный ресурс]. – Режим доступа: http://mvd.ru
2. The Blockchain Explained to Web Developers, Part 1: The Theory [Электронный ресурс]. – Режим доступа: https://marmelab.com/blog/2016/04/28/blockchain-for-web-developers-the-theory.html
3. Просто и доступно о Blockchain. Что это и как работает [Электронный ресурс]. – Режим доступа : https://golos.io/ru--golos/@aleco/prosto-i-dostupno-o-blockchain-chto-eto-i-kak-rabotaet
4. Proof-of-stake [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/Proof-of-stake
5. Описание работы Delegated Proof-of-stake [Электронный ресурс]. – Режим доступа : https://distributedlab.com/blog/ru/description-of-delegated-proof-of-stake
6. PinarOzisik A. Graphene: A New Protocol for Block Propagation Using Set Reconciliation / PinarOzisik. A. и др. // European Symposium on Research in Computer Security International Workshop on Data Privacy Management Cryptocurrencies and Blockchain Technology. – 2017. – 1. – P. 423–430.

РЕАЛИЗАЦИЯ ПОТОКОБЕЗОПАСНЫХ СТРУКТУР ДАННЫХ НА ОСНОВЕ МЕТОДА ДЕЛЕГИРОВАНИЯ ВЫПОЛНЕНИЯ КРИТИЧЕСКИХ СЕКЦИЙ ВЫДЕЛЕННЫМ ПРОЦЕССОРНЫМ ЯДРАМ

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

Введение

Современные многоядерные вычислительные системы повсеместно используются для решения задач различной сложности. Эффективность использования таких систем при выполнении на них многопоточных программ зависит от реализации в параллельных программах потокобезопасных структур данных [1] С увеличением количества процессорных ядер в многопроцессорных вычислительных системах остро ставится задача обеспечения масштабируемого доступа к параллельным структурам данных. Данная работа посвящена реализация потокобезопасных структур данных на основе метода делегирования выполнения критических секций выделенным процессорным ядрам. Для анализа эффективности полученной потокобезопасной структуры были проведены эксперименты с аналогичной структурой данных на основе обычных блокировках (lock).

Реализация потокобезопасного стека на основе алгоритма RCL

Remote Core Locking [2] или RCL предназначен для улучшения производительности мультипроцессорных приложений посредством выполнения критических секций на одном или нескольких отдельных процессорных ядрах. RCL позволяет повысить производительность многопоточных программ за счет уменьшения накладных расходов на выполнение критических секций. В результате сокращается конкурентный доступ и повышается локальность обращений к памяти, что позволяет уменьшить количество промахов по кэшу. Метод RCL может быть использован для оптимизации существующих многопоточных программ без их значительного изменения. В основе данного подхода лежит наблюдение, что большинство приложений не используют все существующие ядра

процессора, и, следовательно, можно выделить ядра, которые не задействованы при выполнении той или иной программы.

Метод RCL основан на замене выполнения инструкций критических секций в параллельных потоках, выполняющих основной код, на удаленный вызов процедур для выполнения этих инструкций на выделенном процессорном ядре (рисунок 1).

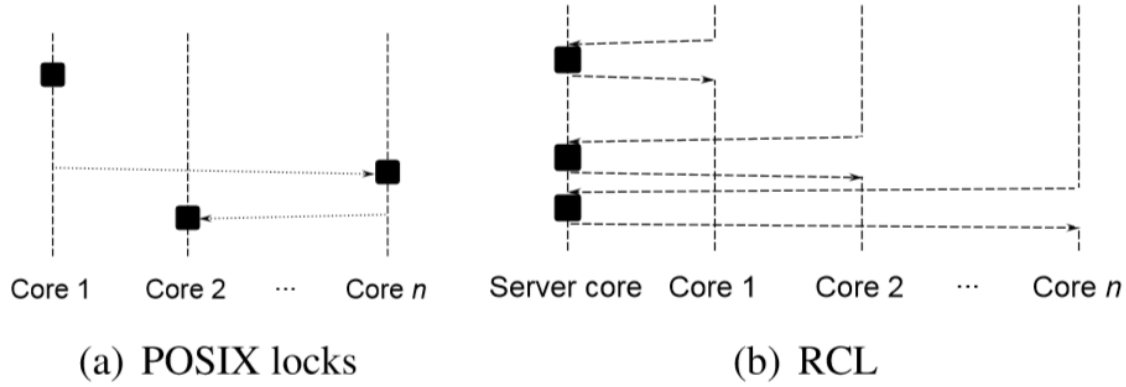


Рисунок 1 – Сравнение обычных POSIX блокировок и блокировок RCL

Для реализации функции удаленного вызова процедур потоки взаимодействуют через массив запросов (рисунок .2).

Рисунок 2 – Массив запросов в RCL.

Результаты экспериментов

Для сравнения эффективности работы RCL в рамках данной работы были реализованы две программы: первая программа реализует использование стека на обычных блокировках (с использованием библиотеки boost[3] для передачи mutex'a между процессорами), вторая - эмитирует работу алгоритма RCL. Для удобства запуска программы сразу на нескольких процессах был использован MPI [4].

Были проведены эксперименты на различном кол-ве процессов и на различном количестве исходных данных (n = 1000, 10000 и 100000 элементов), для которых каждый процесс выполнял операции ***Push*** и ***Pop*** со стеком.

На рисунке 3 приведён результат эксперимента при n=100000 (по оси x - количество процессов, по оси y - количество обращений к данным).

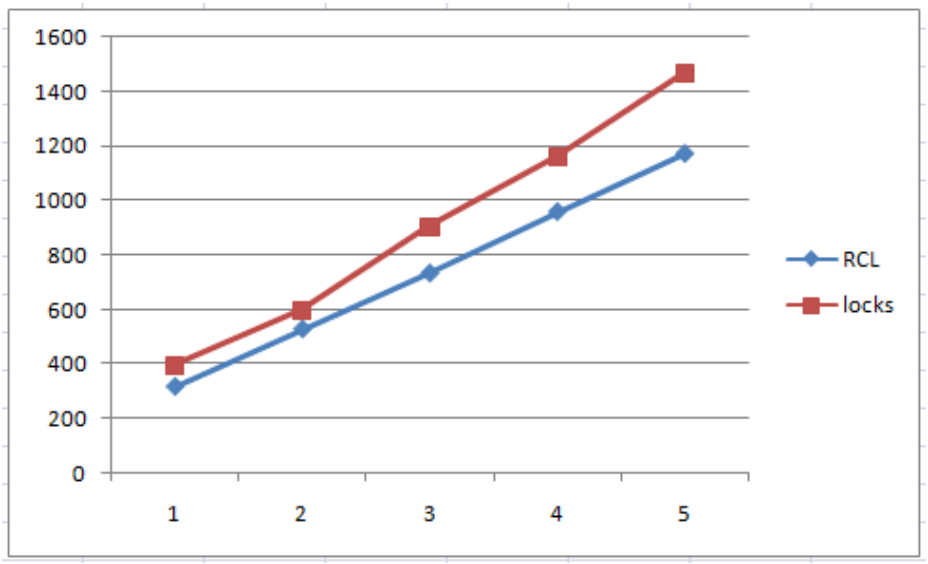


Рисунок 3 – Результаты экспериментов – время работы алгоритма на основе RCL и на основе синхронизирующмх блокировок

График демонстрирует эффективность работы алгоритма RCL, причём с увеличением рабочих процессов разница во времени между RCL и обычными блокировками начинает заметно расти.

Выводы

Метод RCL позволяет наиболее эффективно повысить масштабируемость потокобезопасного стека по сравнению с другими аналогичными реализациями. Оптимизация достигается преимущественно за счет того, что элементы структур данных (вершина стека) находятся в кэш-памяти одного процессора, на котором функционирует RCL-сервер, что позволяет минимизировать количество промахов по кэшу при выполнении операций в параллельных потоках.

Список литературы

1. Fatourou P., Kallimanis N. D. Revisiting the combining synchronization technique //ACM SIGPLAN Notices. ACM. - 2012. - Т. 47. - № 8. - P. 257-266.
2. Lozi J.P. et al. Remote Core Locking: Migrating Critical-Section Execution to Improve the Performance of Multithreaded Applications *ft* USENIX Annual Technical Conference. - 2012.
3. Boost C++ Libraries [Электронный ресурс]. – Режим доступа: http://www.boost.org/
4. Peter Pacheco Parallel Programming with MPI. 1st Edition. - Imprint: Morgan Kaufmann, 1996. – 500 p.

ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ В ЗАДАЧАХ РАСПОЗНАВАНИЯ РЕЧИ

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

Системы распознавания голоса – это вычислительные системы, которые могут определять речь говорящего из общего аудио потока. Эта технология связана с технологией распознавания речи, которая преобразует произнесенные слова в цифровые текстовые сигналы, путем проведения процесса распознавания речи машинами. Обе эти технологии

используются параллельно: с одной стороны для идентификации голоса конкретного пользователя с другой стороны для идентификация голосовых команд посредством распознавания речи. Распознавание голоса используется в биометрических целях безопасности, чтобы определить голос конкретного человека.

Существующие проблемы

Невозможность подавления внешних шумов

Несмотря на технический прогресс в сфере распознавания голоса, шумы продолжает оставаться одной из основных проблем на мировом рынке распознавания голоса. Кроме того, голосовая биометрия отличается особенной чувствительностью по сравнению с другими видами биометрии. Приложения распознавания голоса, голосовой биометрии и распознавания речи оказываются очень чувствительными к шуму окружающей среды. В результате, любое шумовое нарушение препятствует точности распознавания. Также нарушается автоматизированный ответ на голосовую команду. Неспособность подавить окружающий шум является единственным фактором, который не дает системам распознавания голоса достичь высоких результатов.

Проблемы с точностью распознавания

На мировом рынке распознавания голоса единой проблемой является невысокие показатели точности распознавания, не смотря на то, что в настоящее время системы распознавания голоса способны распознавать различные языки и определять подлинность голоса. Так как система включает в себя сложный процесс согласования баз данных с произносимыми командами и интегрированной технологией распознавания речи и голосовой верификации, даже незначительная ошибка в любой часть процесса может привести к неверному результату. Погрешность в распознавании речи является одним из основных ограничений в приложениях распознавания голоса. Однако некоторые производители начали разработку систем с очень низким уровнем погрешности в распознавании голоса. Они разработали системы с менее чем 4% неточных результатов (например, измерения голосовой биометрии неверно идентифицируют и отвергают голос человека, у которого есть доступ).

Общий алгоритм распознавания речи

Общий алгоритм распознавания связной речи:

- Исходный сигнал
- Начальная фильтрация и усиление полезного сигнала
- Выделение отдельных слов
- Распознавание слова
- Распознавание речи
- Реакция на распознанный сигнал

Этапы распознавания:

1. Обработка речи начинается с оценки качества речевого сигнала. На этом этапе определяется уровень помех и искажений.
2. Результат оценки поступает в модуль акустической адаптации, который управляет модулем расчета параметров речи, необходимых для распознавания.
3. В сигнале выделяются участки, содержащие речь, и происходит оценка параметров речи. Происходит выделение фонетических и просодических вероятностных характеристик для синтаксического, семантического и прагматического анализа. (Оценка информации о части речи, форме слова и статистические связи между словами.)
4. Далее параметры речи поступают в основной блок системы распознавания —

декодер. Это компонент, который сопоставляет входной речевой поток с информацией, хранящейся в акустических и языковых моделях, и определяет наиболее вероятную последовательность слов, которая и является конечным результатом распознавания.

Применение скрытых Марковских моделей для распознавания речи

Скрытой Марковской моделью (СММ) называется модель состоящая из N состояний, в каждом из которых некоторая система может принимать одно из M значений какого-либо параметра. Вероятности переходов между состояниями задается матрицей вероятностей A={aij}, где aij – вероятность перехода из i-го в j-е состояние. Вероятности выпадения каждого из M значений параметра в каждом из N состояний задается вектором B={bj(k)}, где bj(k) – вероятность выпадения k-го значения параметра в j-м состоянии. Вероятность наступления начального состояния задается вектором $\pi=\{\pi i\}$, где πi – вероятность того, что в начальный момент система окажется в i-м состоянии. Таким образом, скрытой Марковской моделью называется тройка $\lambda=\{A,B,\pi\}$.

Использование скрытых Марковских моделей для распознавания речи основано на двух приближениях:

1. Речь может быть разбита на фрагменты, соответствующие состояниям в СММ, параметры речи в пределах каждого фрагмента считаются постоянными.
2. Вероятность каждого фрагмента зависит только от текущего состояния системы и не зависит от предыдущих состояний.

Модель называется «скрытой», так как нас, как правило, не интересует конкретная последовательность состояний, в которой пребывает система. Мы либо подаем на вход системы последовательности типа O={o1,o2,…oi} - где каждое oi – значение параметра (одно из M), принимаемое в i-й момент времени, а на выходе ожидаем модель $\lambda=\{A,B,\pi\}$ с максимальной вероятностью генерирующую такую последовательность, - либо наоборот подаем на вход параметры модели и генерируем порождаемую ей последовательность. И в том и другом случае система выступает как “черный ящик”, в котором скрыты действительные состояния системы, а связанная с ней модель заслуживает названия скрытой.

Для осуществления распознавания на основе скрытых моделей Маркова необходимо построить кодовую книгу, содержащую множество эталонных наборов для характерных признаков речи (например, коэффициентов линейного предсказания, распределения энергии по частотам и т.д.). Для этого записываются эталонные речевые фрагменты, разбиваются на элементарные составляющие (отрезки речи, в течении которых можно считать параметры речевого сигнала постоянными) и для каждого из них вычисляются значения характерных признаков. Одной элементарной составляющей будет соответствовать один набор признаков из множества наборов признаков словаря. Фрагмент речи разбивается на отрезки, в течении которых параметры речи можно считать постоянными. Для каждого отрезка вычисляются характерные признаки и подбирается запись кодовой книги с наиболее подходящими характеристиками. Номера этих записей и образуют последовательность наблюдений O={o1,o2,…oi} для модели Маркова. Каждому слову словаря соответствует одна такая последовательность. Далее A – матрица вероятностей переходов из одного минимального отрезка речи (номера записи кодовой книги) в другой минимальный отрезок речи (номер записи кодовой книги). B – вероятности выпадения в каждом состоянии конкретного номера кодовой книги.

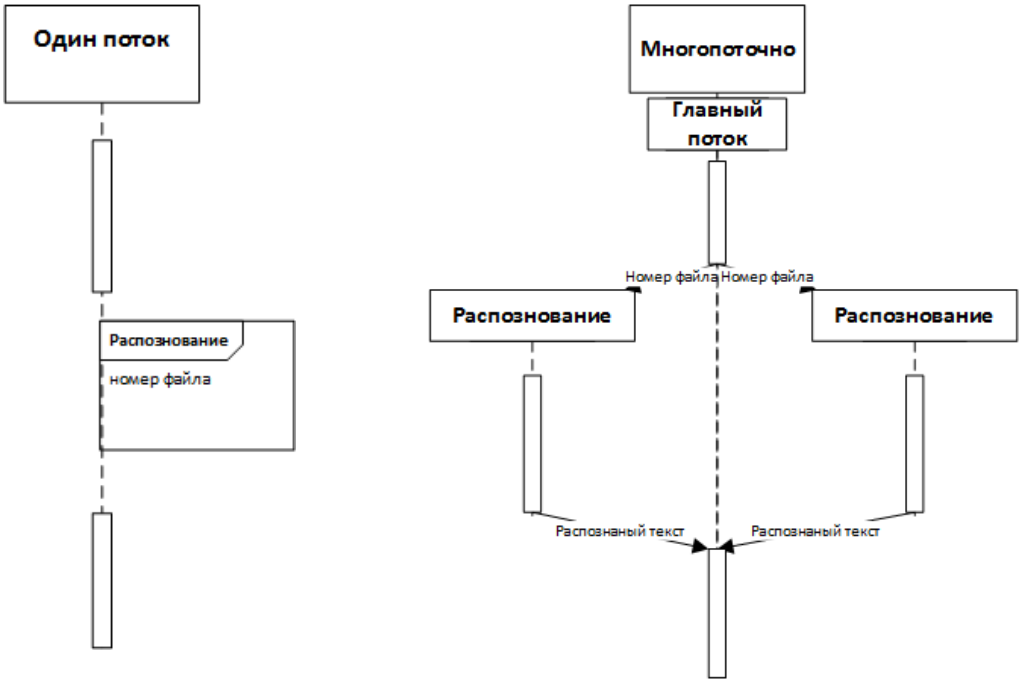
На этапе настройки моделей Маркова мы применяем алгоритм Баума - Уэлча для имеющегося словаря и сопоставления каждому из его слов матрицы A и B. При распознавании мы разбиваем речь на отрезки, для каждого вычисляем набор номеров кодовой страницы и применяем алгоритм прямого или обратного хода для вычисления вероятности соответствия данного звукового фрагмента определенному слову словаря. Если

вероятность превышает некоторое пороговое значение – слово считается распознанным.

Предпосылки для использования параллельных вычислений

- Большой массив однородных входных данных;
- Возможность выделения равнозначных участков распознавания (отдельных слов, отдельных звуков);
- Высокая сложность функций -примитивов (FFT, cepstral analysis, другие обобщенные характеристики участка речи);
- Независимость примитивных преобразований данных;
- Сложная база данных звуковых примитивов.

Ввиду данных предпосылок разумным решение будет использовать распараллеливание по данным. Изначальный аудиофайл бьется по промежуткам тишины. (Подбираются эмпирически исходя из аудио файла). И для каждого файла запускается распознавание по грамматике на основе скрытых Марковских моделей.



Выводы

Данный алгоритм параллельной обработки звуковой информации позволяет значительно сократить время распознавания. Его можно применять при обработке больших массивов данных для выделения конкретных слов.

Список литературы

1. Фланаган Дж.Л. Анализ, синтез и восприятие речи / пер. с англ. А. А. Пирогова. - М.: Связь, 1968. - 397 с.
2. Кузнецов В., Отт А. Автоматический синтез речи. - Таллинн: Валгус, 1989. - 135 с.
3. Вишнякова О.А., Лавров Д.Н. Применение преобразования Гильберта-хуанга к задаче сегментации речи // Математические структуры и моделирование. – 2011. - вып. 24. – С. 12-18
4. Чекмарев А. Речевые технологии – проблемы и перспективы. // Компьютерра. – 1997. - №49. - С. 26-43.

РАЗРАБОТКА БЕНЧМАРКОВ ДЛЯ АНАЛИЗА КОММУНИКАЦИОННЫХ ФУНКЦИЙ СТАНДАРТА MPI

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

Актуальность работы

Современные кластерные системы и суперкомпьютеры, прежде всего, работают с MPI [1] как с основным средством, на котором реализуются параллельные программы. В первую очередь MPI ориентирован на системы с распределенной памятью, то есть когда затраты на передачу данных велики. Но чтобы использовать MPI оптимально, имеется необходимость в бенчмарке коммуникационных функций MPI.

Тестирование кластеров необходимо проводить по многим объективным причинам. На мой взгляд, основной причиной для тестирования должно быть естественное желание увеличить вычислительную мощность системы.

Анализ существующих систем

На данный момент имеется, как минимум, два больших пакета с различными бенчмарками различных функций MPI. Например, Intel® MPI Benchmarks [2] и MVAPICH: MPI over InfiniBand [3]. Оба пакета распространяются под свободной лицензией. Основным их недостатком является отсутствие визуализации результатов тестирования. Все результаты после выполнения тестов будут представлены на консоли.

Тестирование коммуникационных функций MPI

В процессе реализации данной работы для тестирования бенчмарков потребовалось реализовать несколько программ, распараллеливающих вычисления. Структура каждой из них практически идентична, за исключением типа используемых при вычислении данных и размерности задачи.

При реализации программ были выбраны контрольные значения для тестирования. Использованы основные типы передаваемых данных int и double. Четыре варианта размерности задачи: 200 вершин, 1000 вершин, 1500 вершин, и 2000 вершин. Данные значения объясняются следующим:

- Для построения графиков необходимо, как минимум, 3 значения. Было решено взять 8 для более точного построения графика.
- 200 вершин — это минимальное количество вершин, при котором время работы процесса, больше 0 миллисекунд.
- 2000 вершин — количество вершин при котором, время выполнения тестирования будет не очень большим.
- 600, 800, 1000, 1200, 1500 1800 вершин — промежуточные значения.

Вариантов использования процессов было решено взять 3:

- 3 процесса (из них 2 рабочих)
- 6 процессов (5 рабочих)
- 9 процессов (8 рабочих)

Таким образом, количество запускаемых параллельных приложений равно декартову произведению количества типов данных (T), количеству вариантов размерности задачи (R) и количеству вариантов используемых процессов (P). TxRxP = 2x8x3 = 48.

Для коммуникации между процессами, были выбраны основные коммуникационные функции: MPI_SEND и MPI_RECV.

За основу задачи для параллельного вычисления была принята задача нахождения кратчайших путей между всеми вершинами в графе алгоритмом Флойда-Уоршелла.

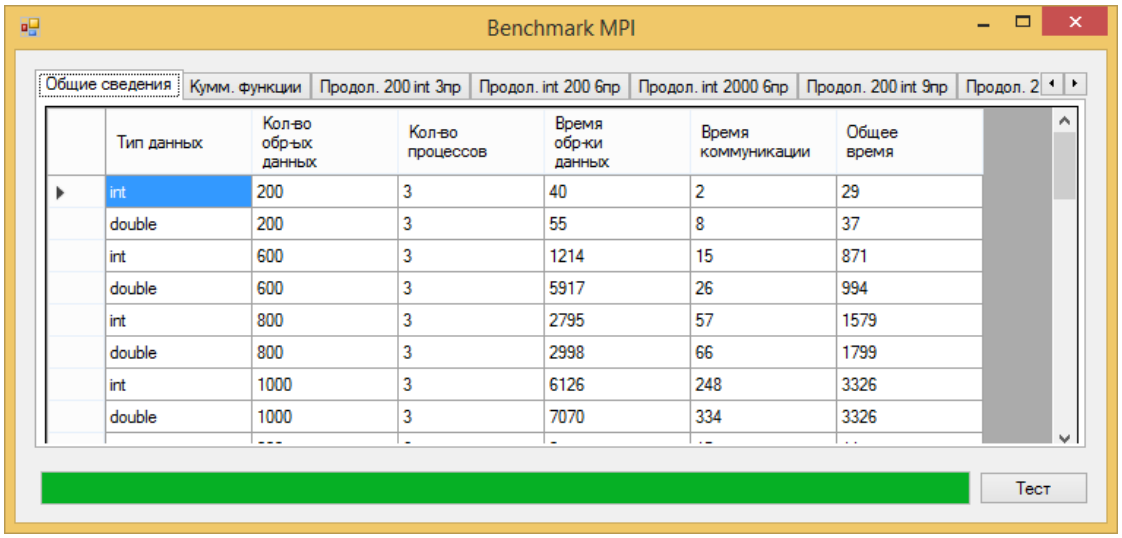
Выходные данные записываются в файлы в определенном формате с помощью функций MPI для ввода/вывода: MPI_FILE_OPEN и MPI_FILE_WRITE.

Используя выходные данные распараллеливающих приложений на основе MPI, были построены таблица и графики результатов в клиентском приложении.

Результаты тестирования

Программа отображает на главном окне набор табов, в каждом из которых имеются результаты тестирования MPI. Она позволяет посредством мыши запустить тест и просматривать результаты.

Результаты тестирования MPI представленные в виде таблицы общих сведений (Рисунок 1) тестирования и графиков: коммуникационных функций (Рисунок 2) и продолжительности выполнения расчетов и коммуникации процессов.



Тип данных	Кол-во обр-ых данных	Кол-во процессов	Время обр-ки данных	Время коммуникации	Общее время
int	200	3	40	2	29
double	200	3	55	8	37
int	600	3	1214	15	871
double	600	3	5917	26	994
int	800	3	2795	57	1579
double	800	3	2998	66	1799
int	1000	3	6126	248	3326
double	1000	3	7070	334	3326

Рисунок 1 - Таблица общих сведений тестирования

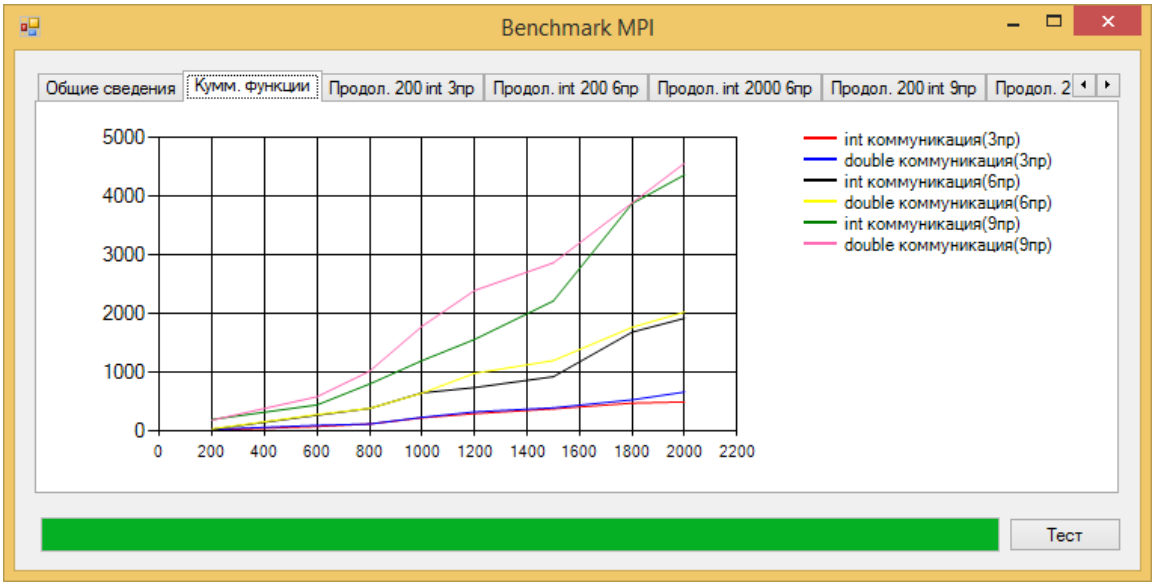


Рисунок 2 - График результатов тестирования коммуникационных функций

Выводы

Предложенный вариант решения задачи отличается от существующих тем, что в вычислениях участвует реальная задача вычисления кратчайших путей в графе, и визуализацией решения. В дальнейшем система может быть расширена исследованием других коммуникационных функций и другими бенчмарками.

Список литературы

1. MPI: The Message Passing Interface [Электронный ресурс]. – Режим доступа: http://parallel.ru/tech/tech_dev/mpi.html
2. Intel MPI Benchmarks [Электронный ресурс]. – Режим доступа: https://software.intel.com/en-us/articles/intel-mpi-benchmarks
3. MVAPICH: MPI over InfiniBand. Электронный ресурс:http://mvapich.cse.ohio-state.edu/benchmarks/
4. Rob van der Wijngaart. The NAS Parallel Benchmarks 2.4. Report NAS-02-007, October, 2002.
5. Jack J. Dongarra, Piotr Luszczek, and Antoine Petitetz. The LINPACK Benchmark: Past, Present, and Future. December, 2001.

ПРОЕКТ С РАЗВИВАЮЩИМИ ЗАНЯТИЯМИ ДЛЯ ДЕТЕЙ

Коновальчук Е.С., Козликина Е.Ю. – студенты
Алтайский государственный технический университет (г. Барнаул)

В работе рассмотрены вопросы создания проекта с развивающими занятиями для детей. Представленные материалы демонстрируют особенности проекта сайта и android-приложения, содержащих комплекс заданий и тестов для проверки и закрепления знаний у детей в рамках школьной программы.

Многие образовательные учреждения, занимающиеся предоставлением услуг в условиях современных информационных технологий, стремятся к созданию и обновлению собственных web-ресурсов и приложений. Это позволяет предоставлять актуальные услуги, эффективно проводить занятия с детьми, с возможностью контроля учебного процесса. Поэтому проект, содержащий развивающие занятия, является актуальным информационным ресурсом для многих учебных учреждений. После исследования аналогичных сайтов и приложений, были выявлены их достоинства и недостатки, что учитывалось в дальнейшем при разработке проекта.

Было принято решение разработать проект, содержащий различные задания на проверку знаний, полученных в учебное время. Приложение может использоваться как в работе на уроках, так и во внеурочное время. В качестве предмета была выбрана география (в дальнейшем планируется добавить другие дисциплины). Для лучшей проверки усвоенного материала были составлены различные формы заданий, например: тестовые задания, задания на соответствие, работа с картой и др. В приложении задания сгруппированы по темам. В случае не прохождения задания в указанный срок, тема становится недоступной. На прохождение каждого курса занятий предусмотрено определенное количество попыток, так же предусмотрена система оценивания по каждой пройденной теме, если задание

просрочено – оценка «0». По итогу прохождения всего курса выводится общая оценка, которая рассчитывается как среднее арифметическое. Для того, чтобы ученик мог планировать прохождение темы, в приложении будет присутствовать план занятий на каждую неделю.

На рис. 1 представлена структура разрабатываемого проекта. Главная страница предназначена для отображения основной информации о проекте. На странице «Предметы» отображается список предметов. На странице «Классы» отображается список классов, в которых изучается выбранный предмет. На странице «Темы» отображается список тем (тема может содержать подтему). На странице «План занятий» отображается план занятий по неделям.

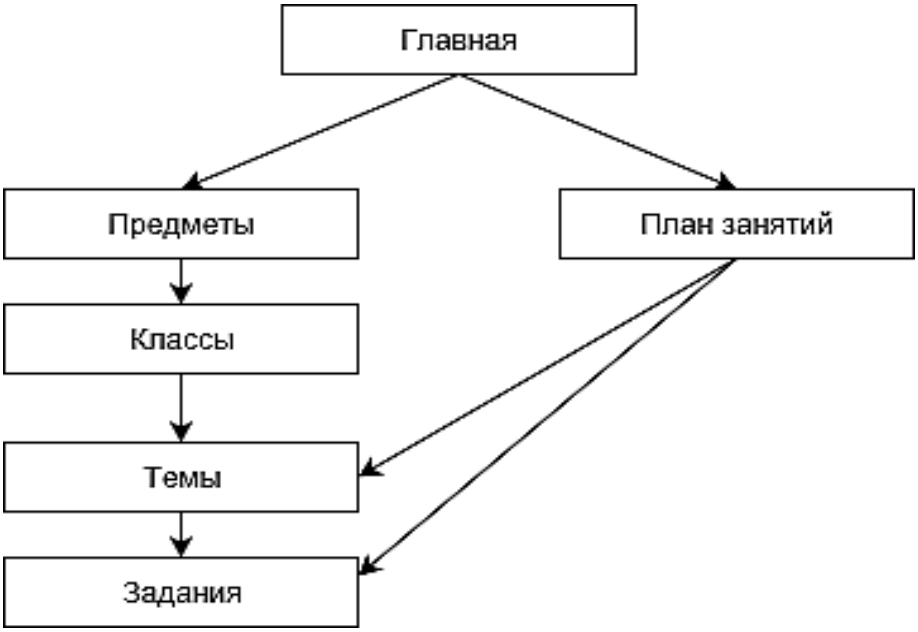


Рисунок 1 – Структура проекта

Web-приложение разрабатывается на языке программирования PHP, с использованием фреймворка Silex и сервера Apache [1]. С помощью Silex можно максимально отделить логические части кода друг от друга и сделать их независимыми. В качестве СУБД выбрана система MySQL [2]. Самым главным преимуществом этого СУБД является гибкость. Эта гибкость обеспечивается поддержкой огромного количества различных таблиц

Android-приложение разрабатывается на языке программирования Java в среде разработки Android SDK. Android SDK универсальное средство разработки мобильных приложений для операционной системы Android. Отличительной чертой от обычных редакторов для написания кодов является наличие широких функциональных возможностей

Все используемые программы – лицензионные. Развертывание ПО и опытная эксплуатация проекта планируется с начала 2018-2019 учебного года. Экспериментальная проверка проектных решений запланирована в конце текущего учебного года.

Список литературы

1. PHP Silex Docs [Электронный ресурс]. – Режим доступа: https://silex.symfony.com/doc/2.0/
2. Никсон Р. Создаем динамические веб-сайты. 4 изд. - СПб.: Питер, 2017. - 766 с.

БЕЗМАРКЕРНОЕ ОПРЕДЕЛЕНИЕ КУБА В ПРОСТРАНСТВЕ ПРИ РАЗРАБОТКЕ АЛГОРИТМОВ И ПРИЛОЖЕНИЙ ДОПОЛНЕННОЙ РЕАЛЬНОСТИ

Баев В.Е. - студент, Боровцов Е.Г. - к.т.н., профессор
Алтайский государственный технический университет (г. Барнаул)

Главная задача всех существующих IT-технологий состоит в том, чтобы так или иначе облегчить жизнь конечным пользователям и/или разработчикам ПО. Одной из таких технологий является дополненная реальность. Дополненная реальность позволяет представлять привычную информацию в форме интеграции ее с объектами реального мира. Главная идея такого подхода состоит в том, чтобы совместить некий виртуальный объект (текст, картинки, трехмерные модели и т.д.) и видеопоток, полученный с камеры устройства [1]. Примерная техническая схема представлена на рисунке 1.

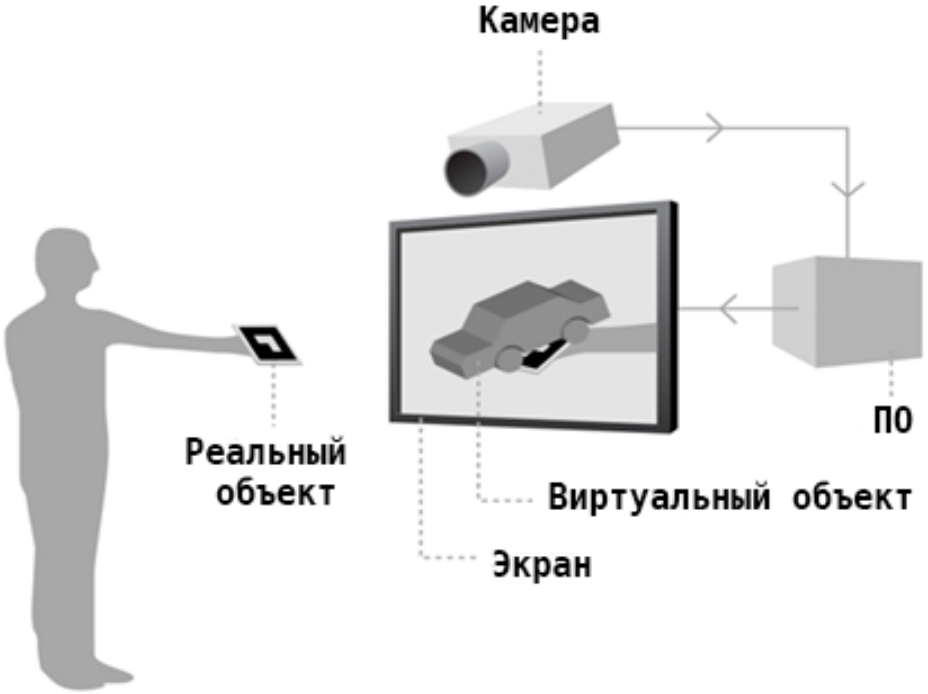


Рисунок 1 - Техническая схема дополненной реальности

Такое представление информации может найти применение почти во всех сферах жизни. От игр до образования (например, интерактивный учебник физики) и нужд бизнеса (например, визуализация инженерной системы).

Целью данной работы является проектирование и разработка программного продукта, способного определять простейшие трехмерные модели в пространстве. Приложением данного программного продукта будет являться программное обеспечение, реализующее функцию помощника в решении логических игр. В данном случае это будет мобильное приложение-помощник в решении задачи сборки кубика Рубика.

Для реализации данного ПО необходимо решить две задачи, из которых собственно и состоит технология дополненной реальности: определение поверхностей и наложение на видеопоток графической информации. В качестве инструментом, обеспечивающего решение первой задачи, выбрана библиотека компьютерного зрения OpenCV и ее имплементация для платформы Android, для второй – интерфейс для вывода 2D/3D графики OpenGL (OpenGL ES – OpenGL для встраиваемых и мобильных систем. Также за основу решения комбинаций кубика Рубика был выбран двухфазный алгоритм Коцембы [2].

Рассмотрим подробнее процесс определения физического кубика Рубика в пространстве. Данная задача решалась бы достаточно легко, если бы была возможность введением специфического маркера и дальнейшим наложением объектов на него, но в реалиях существующей задачи это нереализуемо, т.к. кубик Рубика имеет множество комбинаций и, соответственно, не имеет какого-то одного изображения. В подобных случаях единственным способом является безмаркерное определение объекта [3].

Методы безмаркерного определения ориентации объектов можно разделить на два типа: оптические и сенсорные. Основной недостаток сенсорных методов определения состоит в том, что для их работы необходимы специализированные датчики, что существенно ограничивает применение данного метода. Оптические безмаркерные методы хороши тем, что весь процесс определения объектов возлагается на программное обеспечение, однако это обстоятельство повышает их сложность. Одним из факторов сложности является неопределенность системы, т.е. необходимо иметь полную информацию о объекте в пространстве, чтобы можно было корректно его определить. Также немаловажную роль играют условия, в которых находится объект: перекрытие другим объектом, освещение, величина дисторсии камеры [4] и другие оптические помехи.

Безмаркерное определение объектов основывается на поиске ключевых точек на кадре видеопотока. Особая или ключевая точка изображения – это точка с характерной окрестностью, которая имеет особые признаки, существенно отличающие ее от основной массы точек. Это означает, что особая точка может быть угловой, изолированной точкой локального максимума (минимума) линии интенсивности, концом линии или точкой на кривой, где кривизна локально максимальна [5].

Для случая с мобильным приложением, когда ресурсы сильно ограничены по сравнению с персональными компьютерами, была подобрана оптимальная последовательность действий и соответствующие коэффициенты расчетов. Схема алгоритма представлена на рисунке 2 (в прямоугольнике указан использованный алгоритм на данной шаге).

Правильность определения объектов (при отлаженном алгоритме) во многом зависит от качества модуля камеры и освещения. С искусственным освещением ошибок было заметно больше, чем с естественным. Это было особенно заметно, когда источник света освещал основной объект неравномерно и отражался от поверхностей кубика Рубика.

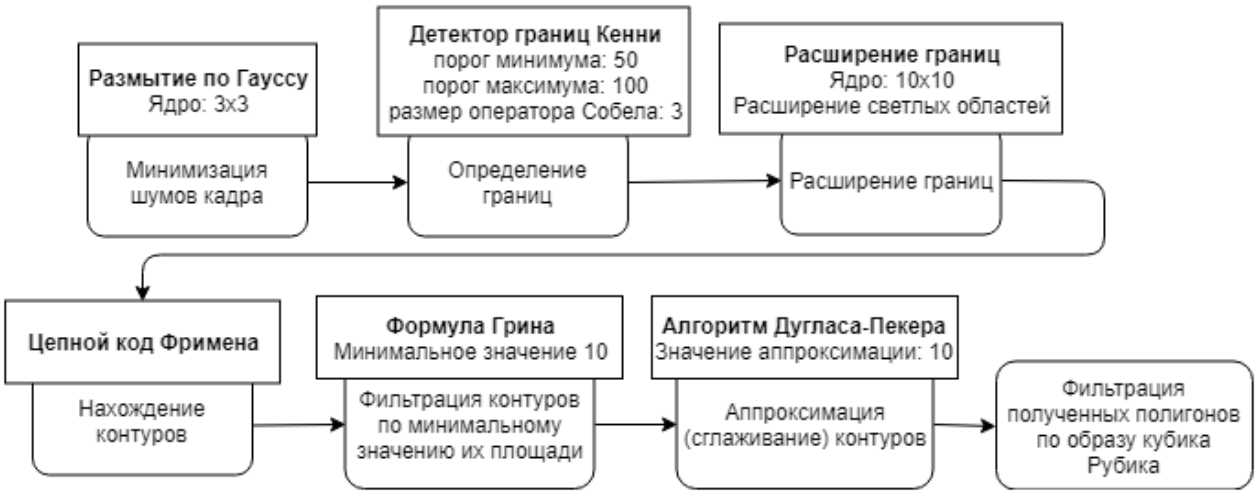


Рисунок 2 - Шаги алгоритма распознавания объекта в виде кубика Рубика

На вход подается кадр видеопотока, преобразованный в матричное представление [6]. В случае правильного определения, результатом является – массив полигонов кубика Рубика. Данный алгоритм дает в среднем 9-12 кадров с секунду при разрешении видеопотока в 1280x720 пикселов. Самыми нагруженными участками алгоритма являются: минимизация

шумов (показатель кадра в секунду падал с максимально возможных 30 до 18-21) и определение границ (от 18-21 до 11-13). На последнем шаге фильтрации осуществляется проверка углов каждой ячейки на гранях кубика (их сумма должна быть равна 360 с некоторой погрешностью) и значения площадей ячеек, которые должны быть равны с некоторым коэффициентом. Результат работы алгоритма по определению куба представлен на рисунке 3.

Рисунок 3 - Результат работы разработанного алгоритма

Безмаркерная технология дополненной реальности хоть и является довольно узконаправленной для конкретного объекта, но в случаях, когда нет возможности введения маркеров, при имеющейся схеме нахождения характерных точек, является хорошим инструментом распознавания.

В итоге для корректного безмаркерного определения трехмерной фигуры в пространстве (в данном случае куба) на мобильных платформах, необходимо находить компромисс между получаемым результатом и производительностью.

Список литературы

1. Дополненная реальность [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Дополненная_реальность
2. Kociemba's Algorithm [Электронный ресурс]. – Режим доступа: https://www.speedsolving.com/wiki/index.php/Kociemba's_Algorithm.
3. Markerless Augmented Reality with a Real-time Affine Region Tracker [Электронный ресурс]. – Режим доступа: http://calvin.inf.ed.ac.uk/wp-content/uploads/Publications/ferrari-isar01.pdf
4. Дисторсия [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Дисторсия
5. Создание панорамных изображений в системах технического зрения [Электронный ресурс]. – Режим доступа: http://www.rusnauka.com/5_NITS_2015/Informatica/3_186623.doc.htm
6. Mat - The Basic Image Container [Электронный ресурс]. – Режим доступа: https://docs.opencv.org/3.1.0/d6/d6d/tutorial_mat_the_basic_image_container.html

РАЗРАБОТКА ГРАФИЧЕСКИХ МОДУЛЕЙ ВИЗУАЛИЗАЦИИ ДЕЙСТВИЙ ИГРОКА ПРИ СИМУЛЯЦИИ СТРАТЕГИЧЕСКИХ ИГР С ИСПОЛЬЗОВАНИЕМ ДОПОЛНЕННОЙ РЕАЛЬНОСТИ

Колесников Е.А. – студент, Боровцов Е.Г. – к.т.н., профессор
Алтайский государственный технический университет (г. Барнаул)

В современном мире стратегические игры (головоломки, а точнее механические головоломки) являются важной частью нашей повседневной жизни. Они помогают людям развивать стратегическое мышление, мелкую моторику рук и т.д. Сейчас мы имеем огромное разнообразие различных головоломок, от самых простых до очень сложных, для решения которых, требуется сообразительность, а не специальные знания высокого уровня. Тем не менее, некоторые головоломки стимулируют теоретические и практические разработки учёных. Так, например, число возможных различных состояний кубика Рубика равно $(8! \times 38 - 1) \times (12! \times 212 - 1) / 2 = 43\ 252\ 003\ 274\ 489\ 856\ 000$, то есть более 43 квинтиллионов комбинаций. Несмотря на это, известно, что из любого состояния кубик можно собрать не более чем за 20 ходов. Иными словами, так называемый «алгоритм Бога» будет давать решения не длиннее 20 ходов.

На данный момент таких головоломок великое множество. Рассмотрим самые популярные из них. Например, решение головоломки «кубик Рубика» заключается в том, чтобы «собрать кубик Рубика»: поворачивая грани куба, вернуть его в первоначальное состояние, когда каждая из граней состоит из квадратов одного цвета. Также хорошо известна головоломка «Ханойская башня», задача в которой состоит в том, чтобы перенести пирамиду из восьми колец за наименьшее число ходов на другой стержень. За один раз разрешается переносить только одно кольцо, причём нельзя класть большее кольцо на меньшее.

Мы живем в эпоху компьютеризации и интернета. Во многих случаях достаточно запроса в интернет в поисковике, чтобы найти нужную информацию для решения той или иной задачи.

Возьмем тот же «кубик Рубика» - существует множество алгоритмов для решения этой головоломки, как для начинающих, так и для уже опытных людей. Как простому человеку, который ни разу не сталкивался с подобными задачами просто и легко собрать «кубик Рубика»? Ответу на этот вопрос и посвящена настоящая работа.

С развитием компьютерного зрения и технологии дополненной реальности решение этой задачи становиться реализуемым даже с помощью мобильных устрпойств. В рамках данной работы, разрабатывается мобильное приложение, способное считывать информацию о цветах и их положениях на гранях куба и давать подсказки в реальном масштабе времени для решение данной головоломки.

Задача состоит в том, чтобы визуализировать алгоритм решения с использованием дополненной реальности (в дальнейшем AR), посредством компьютерной графики (Примеры визуализации результатов работы алгоритма представлены на рисунках. 1 и 2.), основываясь на данных, полученных с камеры мобильного устройства под управлением ОС Android или очков дополненной реальности (в частности, Epson moverio BT-300).

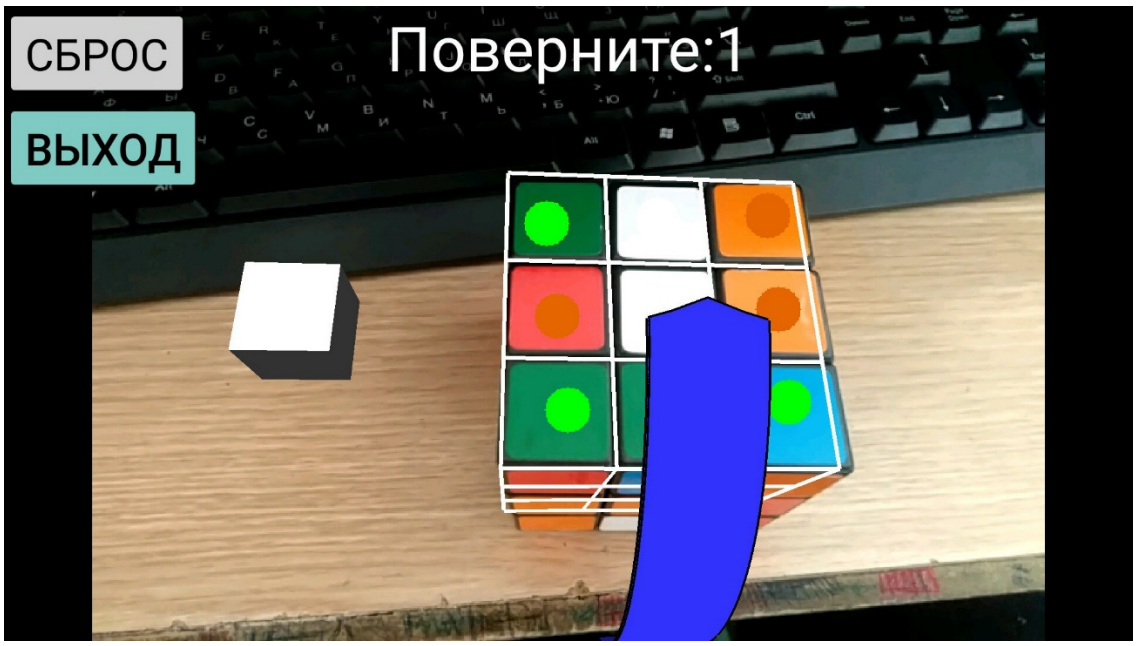


Рисунок 1 - Пример визуализации

Рисунок 2 - Пример визуализации

Разработанное приложение использует современный стек технологий, позволяющий сразу получить приложение для Android. При его реализации были использованы следующие технологии и инструменты разработки:

- OpenCV — библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом. Реализована на C/C++, также разрабатывается для Python, Java, Ruby, Matlab, Lua и других языков.
- Android SDK — универсальное средство разработки мобильных приложений для операционной системы Android. Отличительной чертой от обычных редакторов для написания кодов является наличие широких функциональных возможностей, позволяющих запускать тестирование и отладку исходных кодов, оценивать работу

приложения в режиме совместимости с различными версиями ОС Андроид и наблюдать результат в реальном времени (опционально). Поддерживает большое количество мобильных устройств, среди которых выделяют: мобильные телефоны, планшетные компьютеры, умные очки (в том числе Google Glass), современные автомобили с бортовыми компьютерами на ОС Андроид, телевизоры с расширенным функционалом, особые виды наручных часов и многие другие мобильные гаджеты, габаритные технические приспособления.

В дальнейшем предполагается постоянное пополнение списка головоломок, а также улучшение системы распознавания цветов и положения объектов в пространстве. С развитием мобильных процессоров и графических ускорителей предполагается дальнейшее увеличение скорости обработки данных и нагрузка на основную систему.

Список литературы

1. Open Source Computer Vision Library [Электронный ресурс]. – Режим доступа: https://docs.opencv.org/
2. Алгоритм Бога [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Алгоритм_Бога
3. Android sdk [Электронный ресурс]. – Режим доступа: https://developer.android.com/training/basics/firstapp/?hl=ru
4. Порев В.Н. Компьютерная графика : учебное пособие / В.Н. Порев. - М.: БХВ-Петербург, 2005 .- 432 с.
5. Введение в программирование шейдеров: часть 2 [Электронный ресурс]. – Режим доступа: https://habrahabr.ru/company/plarium/blog/281599/
6. Создание шейдеров [Электронный ресурс]. – Режим доступа: https://habrahabr.ru/post/333002/

РАЗРАБОТКА ПРОГРАММНОЙ СИСТЕМЫ КЛАСТЕРИЗАЦИИ ГРАФОВ ССЫЛОК

Понаморёв А.В. – магистрант
Алтайский государственный технический университет (г. Барнаул)

В настоящее время наблюдается значительный рост интереса к исследованиям в области интеллектуального анализа данных (англ. – data mining), одним из направлений которого является анализ различных сетевых структур с целью поиска ранее неизвестных, нетривиальных, практически полезных и доступных интерпретаций их свойств. В ряде случаев такие структуры естественным образом могут быть смоделированы с помощью графов ссылок – ориентированных невзвешенных графов. Примерами таких сетевых структур могут являться:

- социальные сети (Facebook, Twitter, Одноклассники);
- веб-страницы сети Интернет;
- массивы архивных документов и ссылок между ними;
- схемы белковых взаимодействий [1];
- схемы метаболических процессов [2].

Одним из способов их анализа является кластеризация соответствующих им графов ссылок. Под кластеризацией, как правило, понимается разбиение исходного графа на некоторое число подграфов (кластеров) таким образом, чтобы количество ссылок внутри

кластеров было бы больше, чем количество ссылок между кластерами. Прикладными задачами такого анализа могут быть:

- анализ целевой аудитории при маркетинговых исследованиях;
- быстрый поиск связанных документов;
- исследование функциональных ролей белков;
- снижение размерности входных данных для последующего детального анализа;
- визуализация структуры массивных графов.

Следует отметить, что при анализе некоторых графов (например, графов социальных структур) важным дополнением исходной задачи является то, что одна вершина может принадлежать одновременно нескольким кластерам (сообществам), а сам граф может иметь достаточно сложную структуру сообществ, включающую непересекающиеся, пересекающиеся и иерархически вложенные сообщества.

Актуальной задачей является разработка программной системы, позволяющей осуществить кластеризацию и поиск структуры сообществ в таких графах, при этом следует учесть:

- возможность работы с графами больших размеров;
- оптимальность используемых алгоритмов и структур данных;
- возможность работы как с ориентированными, так и с неориентированными графами;
- возможность автономной работы (без подключения к сети Интернет);
- возможность работы на различных платформах.

Задача кластеризации относится к классу *NP*-сложных, при этом, отличительной особенностью исследуемых графов является большое число вершин N (десятки миллионов) и сравнительно низкое число ребер $E << N^2$ (высокая разреженность). Учитывая это, поиск точного решения даже для небольших графов не представляется возможным, возможно лишь найти некоторое приближенное решение. Следует заметить, что не существует математически строгого определения качества кластеризации, существует только общепринятое понятие кластера (сообщества).

Одной из наиболее эффективных метрик, отражающих качество разбиения (кластеризации) графа является модульность (англ. - modularity), предложенная в [3]. Модульность в случае неориентированного графа может быть рассчитана как:

$$Q = \frac{1}{2m}\sum_{i,j}\left[A_{ij} - \frac{d_i d_j}{2m}\right]\delta(C_i, C_j)$$

где A_{ij} – матрица смежности исходного графа;

m – общее количество ребер в исходном графе;

d_i – степень i-ой вершины исходного графа;

C_i – номер кластера, к которому отнесена i-я вершина;

δ – дельта-функция, принимающая значение 1, если $C_i = C_j$, и 0 в противном случае.

Существуют модификации данной метрики для ориентированных графов, взвешенных графов, а также для учета поправки на ожидаемый размер кластеров.

Большинство наиболее эффективных методов кластеризации, применимых к большим графам так или иначе используют метрику модульности. К таким методам относятся:

- спектральная кластеризация (основана на поиске оптимального разбиения с помощью собственных чисел специальной матрицы модульности);
- методы на основе случайных блужданий (используют модульность в качестве критерия остановки разбиения исходного графа);
- методы на основе жадных оптимизаций модульности.

Одним из наиболее известных и эффективных алгоритмов является алгоритм, представленный в [4] и получивший название «Лувенский метод». С помощью данного алгоритма удалось обработать граф, представляющий сеть абонентов бельгийского оператора сотовой связи, состоящий из 2,2 миллионов вершин. Интересно, что полученная структура графа полностью соответствовала разделению абонентов на аудитории, общающиеся между собой на французском и немецком языках соответственно.

Алгоритм выполняется итеративно в 2 этапа – локальная оптимизация модульности путем объединения соседних вершин в сообщества и построение нового метаграфа, вершинами которого являются сообщества, полученные на предыдущем этапе (рисунок 1).

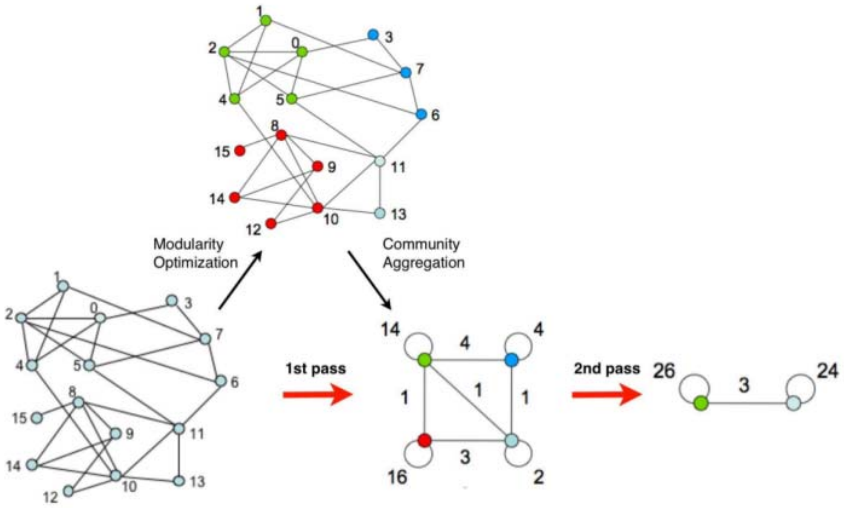


Рисунок 1 – Представление «Лувенского метода» кластеризации графов

Эффективными модификациями данного алгоритма являются методы multilevel [5] и SLM [6] (англ. – smart local moving), выполняющие дополнительные этапы локальной оптимизации модульности. Подробное сравнение некоторых из данных методов приведено в [7]. Наилучших результатов удалось достичь с помощью метода SLM, при этом, скорость его работы является достаточной для работы с графами, содержащими десятки миллионов вершин.

В том случае, если допускается принадлежность вершины одновременно к нескольким сообществам, задача кластеризации становится еще более вычислительно сложной. Наилучшими из известных на текущий момент являются методы, основанные на поиске параметров стохастических моделей, воспроизводящих матрицу смежности исходного графа. Наиболее показательной из них является модель AGM (англ. – affiliation graph model), предложенная группой исследователей Стэндфордского университета [8]. Данная модель представлена на рисунке 2.

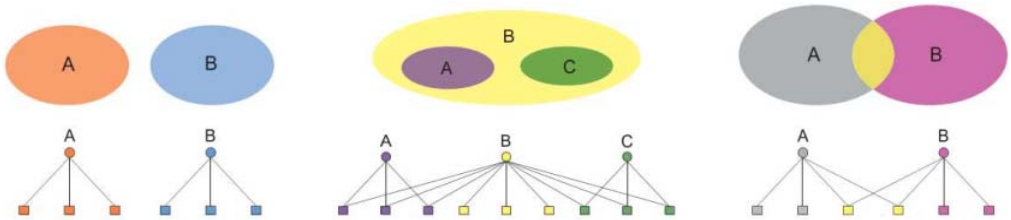


Рисунок 2 – Представление модели AGM

Модель AGM представляет собой двудольный граф B, в верхней доле которого отображены сообщества (кластеры), а в нижней доле – вершины исходного графа. С каждым сообществом сопоставлен единственный параметр p_c – вероятность наличия ребра между

двумя произвольно взятыми вершинами в сообществе *C*. Наличие ребра между долями определяет принадлежность вершины к конкретному сообществу. Общая вероятность *p* восстановления ребра между вершинами *u* и *v* в исходном графе *G(V,E)* определяется как:

$$p(u,v) = 1 - \prod_{k \in Cuv} (1 - p_k)$$

где C_{uv} – множество сообществ, к которым одновременно принадлежат вершины *u* и *v*.

С помощью данной модели можно представить практически любую структуру сообществ в исходном графе – непересекающихся, пересекающихся, иерархически вложенных (рисунок 2). Процесс кластеризации сводится к максимизации функции правдоподобия:

$$\arg\max_{B,\{p_c\}} L(B,\{p_c\}) = \prod_{(u,v)\in E} p(u,v) \prod_{(u,v)\notin E} (1 - p(u,v))$$

Однако, расчет оптимальных параметров данной модели – вычислительно сложная задача, что побудило авторов создать упрощенную модификацию модели AGM – модель BigCLAM (англ. – cluster affiliation model for big networks) [9]. Модель BigCLAM представлена на рисунке 3.

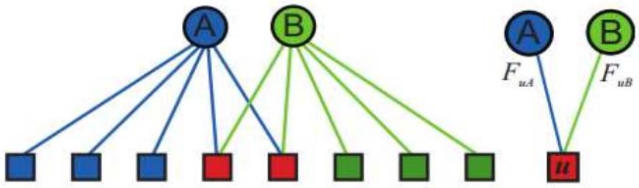


Рисунок 3 – Представление модели BigCLAM

Основное отличие от предыдущей модели – каждая вершина *u* может принадлежать сообществу *C* с некоторой вероятностью F_{uC}, в таком случае вероятность восстановления ребра между вершинами *u* и *v* в исходном графе определяется как:

$$p(u,v) = 1 - \exp(-\sum\nolimits_C F_{uC} \cdot F_{vC})$$

При записи вероятностей принадлежности в матрицу *F*, в которой F_u – вектор вероятностей принадлежности вершины *u* сообществам, логарифм функции правдоподобия может быть рассчитан как:

$$l(F) = \sum_{(u,v)\in E} \log(1 - \exp(-F_u \cdot F_v^T)) - \sum_{(u,v)\notin E} F_u \cdot F_v^T \qquad (1)$$

Задача кластеризации таким образом является задачей максимизации правдоподобия функции (1) и может быть сведена к неотрицательной факторизации матрицы смежности исходного графа:

$$\hat{F} = \arg\max_{F:F_{uC}\geq 0} l(F)$$

Существует модификация данной модели для работы с ориентированными графами – модель CoDA (англ. – communities through directed affiliations) [10]. По результатам исследования ряда графов социальных сетей было показано, что учет ориентации ребер исходного графа оказывает значительное влияние на результат кластеризации.

Ключевым этапом проектирования являлся окончательный выбор методов кластеризации и поиска сообществ с учетом их вычислительной сложности и требованиям к

памяти. При этом, необходимо было учесть ряд доработок и модификаций для работы с ориентированными графами и возможности параллельного выполнения. Для реализации были выбраны следующие методы:

- SLM для разбиения (кластеризации) ненаправленных графов;
- модификация SLM для работы с ориентированными графами;
- модификация BigCLAM с возможностью параллельного выполнения;
- модификация BigCLAM (CoDA) для работы с ориентированными графами.

Программная система реализуется на языке программирования C/C++ с использованием библиотеки для создания графических интерфейсов Qt, технологии OpenMP и на данный момент находится на этапе отладки и тестирования. Окончательный результат ожидается в конце мая 2018 г.

Список литературы

1. N. Krogan et al. Global landscape of protein complexes in the yeast saccharomyces cerevisiae // Nature. - 2006. - vol. 440. - no. 7084. - P. 637–643.
2. D.A. Fell and A.Wagner. The small world of metabolism. // Nature Biotechnology. – 2000. - no.18, p. 1121–1122.
3. Michelle Girvan, M.E. Newman. Community Structure in Social and Biological Networks. - Santa Fe Institute Working Paper: 2001-12-077.
4. Vincent D. Blondel, et al. Fast unfolding of communities in large networks, arXiv:0803.0476v2 [physics.soc-ph], 2008.
5. Rotta, R., & Noack, A. Multilevel local search algorithms for modularity clustering // Journal of Experimental Algorithmics. - 2011.
6. Ludo Waltman and Nees Jan van Eck A smart local moving algorithm for large-scale modularity-based community detection. arXiv:1308.6604 [physics.soc-ph], 2013.
7. Emmons S, Kobourov S, Gallant M, Börner K. Analysis of Network Clustering Algorithms and Cluster Quality Metrics at Scale. PLoS ONE 11 (7): e0159161. doi:10.1371/journal.pone.0159161. - 2016.
8. Jaewon Yang, Jure Leskovec. Community-Affiliation Graph Model for Overlapping Network Community Detection // IEEE International Conference On Data Mining (ICDM). - 2012.
9. Jaewon Yang, Jure Leskovec, Overlapping Community Detection at Scale: A Nonnegative Matrix Factorization Approach // IEEE International Conference On Data Mining (ICDM). - 2013.
10. Jaewon Yang, Julian McAuley, Jure Leskovec, Detecting Cohesive and 2-mode Communities in Directed and Undirected Networks, arXiv:1401.7375v1 [cs.SI], 2014.

РАЗРАБОТКА НАВИГАЦИОННОЙ СИСТЕМЫ ДЛЯ ПОМЕЩЕНИЙ НА ПРИМЕРЕ ГЛАВНОГО КОРПУСА АЛТГТУ

Панкратов С.А., Кобзев Д.И, Терехин А.М, Демченко А.Г. – студенты,
Крайванова В.А. – к.ф.-м.н., доцент
Алтайский государственный технический университет (г. Барнаул)

Когда человек попадает в незнакомое ранее место, ему трудно ориентироваться в пространстве. Если человеку нужно попасть из точки «А» в точку «Б», но он не знает маршрута, то он может прибегнуть к навигационной системе, которая не только проложит маршрут между двумя точками, но и выберет оптимальный из возможных маршрутов.

Трудно представить современный мир без систем такого рода. Навигационные системы применяются повсеместно: в картографии, в транспорте, в военной сфере. Внутри зданий навигационные системы часто реализуются в виде плана, такие планы обычно размещаются при входе в учреждения или торговые центры. Иногда, при входе в здание можно встретить интерактивную навигацию, например, сенсорный информационный терминал, где, кроме маршрута, пользователь может узнать краткую информацию по интересующему его объекту[8]. Однако не всегда такое средство навигации удобно, Так как потребность в навигации может возникнуть в любой точки здания, целесообразнее будет использовать какую-либо портативную систему навигации. На сегодняшний день, многие люди имеют мобильный телефон с возможностью выхода в сеть интернет. Сайт с навигационной системой обеспечит удобство при перемещении в помещении [7].

Существует огромное количество навигационных систем для наземного транспорта. Но для навигации внутри помещений систем в настоящее время очень мало. Из наиболее известных навигационных систем, такую возможность предоставляет приложение «2gis»[6].

Целью данной работы является разработка программного обеспечения, со следующими особенностями.

1. Приложение должно уметь находить кратчайшие маршруты в зданиях, между двумя точками.
2. Приложение должно предоставлять информацию об объектах внутри здания. Например, что находится в кабинете №XX (бухгалтерия, деканат и т.д.)
3. Кроссплатформенность приложения и его совместимость со всеми устройствами.
   a. Отсутствие привязки к какой-либо определенной операционной системе.
   b. Отсутствие необходимости устанавливать приложение.
4. Приложение должно предоставлять удобный графический интерфейс.

Для реализации кроссплатформенности было принято решение разрабатывать веб-приложение на основе HTML, JavaScript, CSS, PHP. Веб-приложение не требует установки и усилий по обновлению, и его можно использовать в браузере на любом устройстве (мобильный телефон, стационарный компьютер).

Для отображения карты используется формат svg - язык разметки масштабируемой векторной графики. Карту здания возможно перенести с архитектурного плана. Так как svg – это частный случай формата XML, все линии, точки и другие графические примитивы на карте представлены в виде кода [5]. При создании карты, на нее заносятся вершины графа, задающие точки, по которым может перемещаться посетитель. Каждой точке задается уникальный идентификатор. На основании этих идентификаторов формируется граф в виде списка смежности, по которому будет осуществлять свою работу алгоритм Дейкстры. Список смежности — один из способов представления графа в виде коллекции списков вершин. Каждой вершине графа соответствует список, состоящий из "соседей" этой вершины [2, с. 459].

Полученный после выполнения алгоритма Дейкстры путь необходимо отобразить на карте. Для этого необходимо изменить файл svg, добавив в него линии, соответствующие пути. Алгоритм Дейкстры, модификация файла svg, и вся внутренняя логика приложения была реализована с помощью языка PHP.

Пользовательский интерфейс, отображение и масштабирование карты реализовано с помощью технологий HTML, JavaScript, CSS.

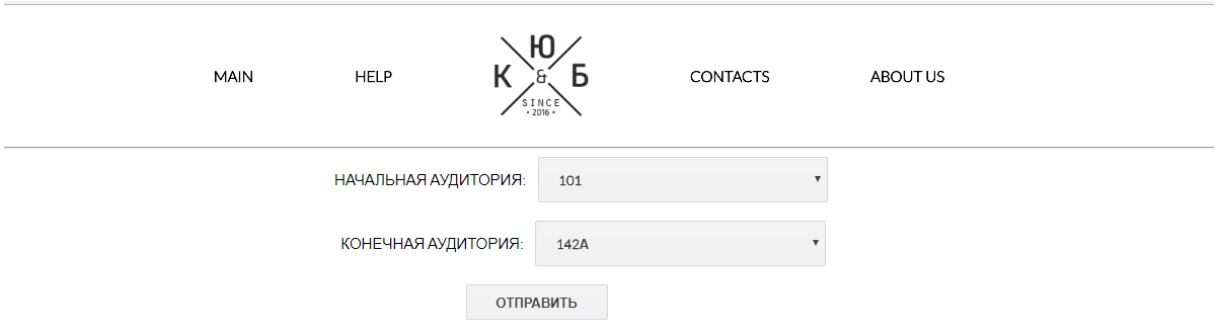


Рисунок 1 – интерфейс для выбора начальной и конечной точки маршрута

Рисунок 2 – результат работы приложения

На рисунках 1 и 2 представлен интерфейс разработанного прототипа приложения. В дальнейшем планируется заменить алгоритм Дейкстры на алгоритм A*. Алгоритм A* более продуктивен по времени благодаря эвристике [4].

Список литературы

1. MAXimal [Электронный ресурс]. - сайт, посвященный алгоритмам – Режим доступа http://emaxx.ru
2. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: построение и анализ. - М.: МЦМНО, 1999. – 960 с.
3. Кнут Д.Э. Искусство программирования: в 3-х томах. - 2-е издание. – М.: Мир, 1976 – 1978 , (3-е изд.: Вильямс, 2010).
4. Алгоритм A* [Электронный ресурс]. - Викиконспекты – Режим доступа https://neerc.ifmo.ru/wiki/index.php?title=%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_A*
5. SVG [Электронный ресурс]. - Википедия: свободная энциклопедия – Режим доступа https://ru.wikipedia.org/wiki/SVG
6. 2gis [Электронный ресурс]. – Карты: улицы, дома и организации города – Режим доступа https://2gis.ru/barnaul
7. Indoorsnavi [Электронный ресурс]. – Indoors Navigation: навигация и трекинг перемещений в зданиях – Режим доступа https://indoorsnavi.pro/
8. СмартКиоск [Электронный ресурс]. – СмартКиоск : Информационные киоски – Режим доступа http://smartkiosk.ru/interactive-kiosk-by-smartkiosk/

РАЗРАБОТКА КОМПЬЮТЕРНОЙ ИГРЫ ЖАНРА TIMEKILLER С СОВЕРШЕННО НОВОЙ КОНЦЕПЦИЕЙ

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

На данный момент в магазинах приложений GooglePlayStore и AppStore 90% игр относятся к жанру TimeKiller. Они служат для того чтобы скоротать время в ожидания чего-либо. И если игра имеет хороший функционал и интересную концепцию, то она обречена на популярность.

Из-за выше сказанного у меня появилась идея создать подобное приложение. Концепция имеет следующий характер: по экрану летает небольшой шар и задача для пользователя следующая: не дать этому шару вылететь за экран; игрок, может используя палец, стилус, мышь или другое управляющее устройство, чертить линии на экране, от которых тот самый шар будет отскакивать. При этом скорость шара будет увеличиваться в зависимости от количества набранных очков (очки набираются при ударе о линии на экране), а также на экране по мере набора очков будут появляться летающие линии, появляющиеся и исчезающие статические и движущиеся фигуры. Тем самым достигнута бесконечная вариация игровых состояний, а, следовательно, пользователь может лицезреть всё новые и новые ситуации, и выходить из них победителем используя свою ловкость и реакцию.

В созданной игре, основная проблема это эффективный контроль столкновений шара с линиями которые могут быть либо нарисованы пользователем, либо произвольно сгенерированы. На каждом шаге игры из всего множества линий на экране выделяется подмножество линий, с которыми шар потенциально может пересечься и анализ столкновений проводится только для них. Тем самым резко снижается объем необходимых вычислений. Если полученное подмножество не пусто, то проверяем будут ли пересекаться вектор движения шара (старые - новые координаты шара) и какая либо линия из нашего подмножества, если да, то вычисляем угол отражения от данной линии, новый вектор движения шара, удаляем линию и зачисляем игроку 1 заработанное очко. И так до тех пор, пока не наступит конец игры.

Созданная игра выполнена в минималистическом стиле, что радует глаз как заядлого игрока, так и человека, который решил зайти в неё на 5 минут. Изначально все цвета имеют серый оттенок, но впоследствии можно выбрать любой другой цвет. Backend выполнен по всем канонам объектно-ориентированного программирования и имеет хорошую производительность на любых устройствах под управлением системы Android. Концепция игры, предложенная мной, нова и достаточно оригинальна. Тем самым полученный продукт может однозначно привлечь определённую аудиторию и привнести в мир timekillergame нечто новое и свежее, что может послужить толчком к новому витку в развитии данного жанра.

В последствии игра получит многочисленные обновления которые преобразят её в лучшую сторону. Из спектра будущих обновлений наиболее актуальны следующие: изменение любых цветов игры, добавление новых фигур, которые будут иметь более сложную форму, а также дополнительные действия фигур, такие как пульсация, сужение или расширение; возможность активировать спасительные рамки вокруг экрана на несколько секунд или до столкновения, или и то, и другое, чтобы спастись от поражения; возможность менять звук соударения с препятствиями и фоновый звук, а также регулировка их громкости; общая таблица результатов всех пользователей, чтобы добавить соревновательную составляющую со всеми игроками со всего мира. Но, как и любая игра она должна сохранить свою изначальную стилистику и идею, чтобы остаться индивидуальной или первой на рынке.

РАЗРАБОТКА БЕСПЛАТНОГО И ВЫСОКОПРОИЗВОДИТЕЛЬНОГО ДВИЖКА ДЛЯ ИГРЫ ИЗ ЖАНРА «БРОДИЛКА»

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

В настоящее время разработка компьютерных игр весьма популярна. Имеется большое количество игровых движков и прочих средств, которые могут серьёзно помочь в этом. Но для мобильных устройств, аппаратные ресурсы которых весьма скромны, большинство универсальных игровых движков не позволяют добиться приемлемого быстродействия и комфортного времени отклика. Цель настоящей работы - создание максимально оптимизированного и простого в использовании движка игр-бродилок для платформы Android. Он позволяет создавать игры с 2D-графикой, где перед игроком ставится цель в виде поиска портала на всё новые и новые уровни. При этом игроку мешают противники, которые атакуют его при первой возможности и имеют разные характеристики, так же игрок может стрелять в противников из 3-х видов оружия: пистолет, автомат и пулемёт, характеристики которых соответственно варьируются (скорость стрельбы, наносимый урон, скорость перезарядки). Так же игрок может подбирать аптеки и коробки с боеприпасами на карте, которые он складывает в инвентарь и использует по мере необходимости. Предусмотрены различные уровни сложности игры и настраиваемый режим, от сложности зависят следующие параметры: урон, наносимый игроку от противников, количество ресурсов, которые появляется на игровом поле, а также скорость и дальность виденья противников. Имеется возможность разместить на карте здания, через стены которых противники не видят игрока. Враги видят только перед собой в области окружности соответствующей 60 – 80 градусам, что позволяет обходить последних за спиной.

Главной и основополагающей задачей является реализация искусственного интеллекта противника, его взаимодействие с окружающими объектами, с игроком и тому подобным. Перемещение противника характеризуется направлением движения и скоростью. Изначально все противники имеют случайное направление и скорость движения. По истечению определённого времени движения, противнику генерируется угол поворота от исходного направления и скорость этого поворота. По завершении этого действия генерируется новое время движения, тем самым противники не поворачиваются резко, так что это бы вызвало у игрока дискомфорт и невозможность играть. Если противник врезается в здание или край карты, то он генерирует те же величины (угол и скорость поворота) и выполняет те же действия, как и в случае смены направления движения без остановки. Особым является случай, когда противник видит игрока, в такой момент он начинает следовать за ним, тем самым направление движение изменяется, время движения по определённому направлению замораживается, а скорость соответственно не меняется. Враг двигается за игроком до тех пор, пока либо игрок не скроется из поля действия противника и в таком случае он движется по направлению движения к последней точке где был игрок используя до конца сгенерированное время движения по направлению, либо если противник настиг игрока, то в таком случае последнему наносится определённый урон, а сам противник замораживается на заданное время, примерно равное времени преодоления игроком от 1 до 3-х радиусов просматриваемой противником области, что даст достаточно времени чтобы убежать от угрозы и не получить нового преследования, ибо в противном случае играть станет очень проблематично.

Так же присутствует грамотно реализованная стрельба с возможностью прострела стен домов, что позволяет игроку выжить в оцеплении врагов. А грамотно реализованный инвентарь позволяет собирать на карте аптечки, боеприпасы и оружие, а затем использовать их по необходимости, а поскольку инвентарь ограничен по размеру, то получим что

пользователю будет необходимо определяться с наиболее приоритетными целями на каждом моменте игры.

В движке уже реализованы джойстики, что делает возможным не использовать сторонние библиотеки.

На подобную концепцию можно накрутить множество различных вариаций противников, различные анимации атаки и тем самым получить продукт, который будет индивидуален и обособлен от подобных творений.

Для демонстрации возможностей игрового движка была создана игра демонстратор. На устройстве со средними и близкими к низким характеристикам она выдавала в 93% случаев стабильное число кадров в минуту, а также одинаковую скорость работы вне зависимости от быстродействия процессора устройства и разрешения экрана.

В итоге данный движок имеет стабильную скорость работы, хорошее быстродействие и не имеет ничего лишнего, что попросту нагружало бы систему в «пустую». Так же значительным плюсом является то, что он бесплатный.

В будущем планируется добавить следующие не маловажные вещи: увеличить интеллект противников (при преследовании игрока они оповещают своих собратьев, которые находятся в определённой области центром которой является сам преследователь), реакцию на выстрелы (выстрел издаёт звук, на который сбегаются враги, находящиеся в определённой окружности с некоторым радиусом и центром в точке произведения выстрела), стреляющие враги и враги которые будут пытаться обходить игрока с разных сторон; новые виды оружия, зданий, добавление некоторого количества брони для игрока.

К ВОПРОСУ О РАЗРАБОТКЕ ТЕСТОВЫХ ЗАДАНИЙ (НА ПРИМЕРЕ ДИСЦИПЛИНЫ «ИССЛЕДОВАНИЕ ОПЕРАЦИЙ»)

Жуйков А.А. – студент, Астахова А.В. – к.э.н., доцент
Алтайский государственный технический университет (г. Барнаул)

При освоении новой учебной дисциплины преподавателю необходимо контролировать уровень освоения студентами учебного материала: знаний, умений и навыков. Обучающиеся при подготовке к занятиям также должны убедиться в усвоения ими приобретенных знаний. Данная задача особенно актуальна в современных условиях работы вузов, при постоянно увеличивающемся времени на самостоятельную работу студентов при соответствующем снижении контактного времени работы с преподавателем.

Одним из инструментов (оценочных средств) контроля знаний, умений и навыков являются тесты. С точки зрения автора тезисов данного доклада, этот инструмент достаточно эффективен, если его использовать в процессе изучения и по результатам изучения каждой темы осваиваемой дисциплины. Наличие объемной базы тестовых заданий в некоторой тестирующей компьютерной оболочке позволяет преподавателю за короткий промежуток времени оценить достаточно большое количество обучающихся.

Очевидно, что составление тестовых заданий требует глубокого знания соответствующей учебной дисциплины. От уровня этих знаний зависит качество тестовых вопросов и соответственно – эффективность тестов как оценочных средств. Здесь следует отметить, что прежде, чем приступить к разработке тестовых заданий по одному из разделов дисциплины «Исследование операций», автор данных тезисов не только изучил теорию по теме «Модели задач линейного программирования», но и решил достаточное количество задач различной сложности, освоив соответствующие алгоритмы. По-видимому, разработка тестовых заданий студентом – это один из методов контроля усвоения им материала.

При составлении тестовых вопросов требуется учитывать, что они должны быть

различной сложности, то есть, пройдя тест, студенты могут набрать различное количество баллов, соответствующее принятой четырёх бальной шкале (от 2 до 5). Самыми простыми, с точки зрения автора тезисов, являются вопросы на определения и вопросы, контролирующие знание алгоритмов (из серии «Как делать?»); самыми сложными – вопросы из серии «Почему?». Правильные ответы на вопросы последнего типа свидетельствуют о том, что студент овладел соответствующем материалом.

Анализ литературных источников позволил выявить некоторые особенности составления тестовых вопросов. Если придерживаться этих суждений, можно создать такой спектр вопросов, который поможет оценить знания обучающихся в достаточно полной мере.

При составлении тестов, прежде всего, необходимо обеспечивать однозначность самого вопроса. Варианты ответа не должны иметь много смыслов, это позволит отвечающему однозначно интерпретировать как ответы, так и сами вопросы. С учетом вышесказанного автором данных тезисов разработаны тесты по разделу «Модели задач линейного программирования» дисциплины «Исследование операций».

Ниже приведены примеры из этих тестов, иллюстрирующие особенности выполнения проделанной работы.

Любой тест должен быть валидным. «Валидность означает пригодность тестовых результатов для той цели, ради чего проводилось тестирование» [1]. Тестирование в дисциплине исследование операций ставит цель: контроль, освоения обучающимися знаний теоретического материала; умений находить решения задач моделей исследования операций; навыков построения моделей. Приведём пример валидного и не валидного теста.

Пример 1 – валидный тест с выбором единственного варианта ответа:

«Модель оптимизационной задачи распределения ресурсов является линейной, когда»:

- в функцию цели управляемые параметры входят в первой степени, в ограничения – в любой степени;
- во все ограничения модели управляемые параметры входят в первой степени, в функцию цели – в любой степени;
- в некоторые ограничения модели управляемые параметры входят в первой степени, в функцию цели – в любой степени;
- в некоторые ограничения модели управляемые параметры входят в первой степени, в функцию цели – в первой степени;
- во все ограничения модели управляемые параметры входят в первой степени и в функцию цели – в первой степени.

Здесь верным является последний вариант ответа: потому что функция цели состоит из комбинации параметров модели, которые находится в первой степени и в ограничениях первые степени, при этом допускаются любые операции между различными параметрами, не приводящие к повышению степени математической модели. Поэтому обучающийся, зная названные аргументы, сможет однозначно интерпретировать как вопрос, так и ответ.

Пример 2 – не валидный тест с выбором нескольких ответов:

«Как найти решение задачи линейного программирования (ЗЛП) геометрическим методом?»:

1) решение может находиться в вершине многоугольника решений;
2) решение может находиться на стороне многоугольника решений;
3) решение может находиться внутри многоугольника решений;
4) решение может находиться вне многоугольника решений.

Тест является не валидным, так как в вопросе не сказано, что речь идёт о двумерной модели ЗЛП (в общем случае имеется в виду не многоугольник решений, а многогранник решений). Кроме того, в вопросе не сказано, каким является решение: оптимальным или просто допустимым.

Другой особенностью, которую необходимо учитывать при составлении тестовых заданий, является следующая. В тестовом вопросе не должно быть подсказок и намёков на ответ; при ответе сам студент должен либо вспомнить ответ из текста – теории по данной теме, либо ответ может быть перефразирован, не в ущерб однозначности. Такой тестовый вопрос позволяет выяснить, насколько глубоко обучающийся овладел данной темой. Иллюстрацией сказанного является пример 3, тест которого перефразирован на основе примера 1.

Пример 3 – тест с выбором нескольких вариантов ответов:

«Модель оптимизационной задачи распределения ресурсов является линейной, когда»:

- во все ограничения модели управляемые параметры входят в первой степени, в функцию цели – в любой степени;
- управляемые параметры модели входят в первой степени в функцию цели;
- в некоторые ограничения модели управляемые параметры входят в первой степени, в функцию цели – в первой степени;
- в ограничения модели управляемые параметры входят в первой степени;
- в некоторые ограничения модели управляемые параметры входят в первой степени, в функцию цели – в первой степени;
- управляемые параметры модели как во всех её ограничения, так и в функцию цели входят в первой степени.

Правильный ответ содержится в высказываниях 2, 3, 6. Очевидно, что студент, недостаточно твёрдо понимающий, что такое линейная модель, встретит затруднения при ответе на поставленный вопрос.

Примером теста (см. пример 4), позволяющим оценить степень овладения студентом изученного материала является также тест, в котором обучающийся должен не только знать список этапов построения модели ЗЛП, но и переосмыслить порядок выполнения этих этапов. Такие тесты, очевидно, не являются тривиальными.

Пример 4 – тест с выбором вариантов правильных последовательностей:

«Установить логически допустимые последовательности вариантов при выборе этапов построения модели ЗЛП:»

а – Выявление управляемых параметров модели;

б – Выявление неуправляемых параметров модели;

с – Выявление показателей эффективности;

д – Построение функции цели и выявление критерия оптимальности;

е – Построение ограничений, накладываемых на управляемые параметры модели;

Варианты ответов:

1) а б с д е
2) а б е с д
3) с б а д е
4) а б д с е

Логика построения оптимизационных моделей ЗЛП позволяет сделать выбор верных ответов: 1), 2), 3).

При составлении вариантов ответов на тестовые вопросы необходимо также учитывать основные особенности формулировок ответов, такие как: повторяемость слов и одинаковая смысловая направленность ответов. Учет этого позволит свести к минимуму вероятность угадывания ответа, исходя исключительно «из логики». Если какие-то слова являются ключевыми в ответе на тестовый вопрос, то они должны встречаться во всех ответах. Сказанное иллюстрируется примером 5, в котором в качестве ключевых слов используются такие понятия как «управляемые и неуправляемые параметры модели».

Пример 5 – тест, допускающий возможность ответить на вопрос с использованием не только логики вывода, но и опыта построения моделей:

«При построении модели ЗЛП следует учитывать, что»:
- управляемые параметры модели обязательно входят в функцию цели;
- неуправляемые параметры обязательно входят в функцию цели;
- во все ограничения модели входят управляемые параметры;
- во все ограничения модели входят неуправляемые параметры.

Знания соответствующего теоретического материала и использование опыта построения моделей ЗЛП позволяют выбрать следующее правильные ответы: 1, 3.

Довольно интересными и достаточно сложными для обучающихся тестами являются вопросы, ответы на которые содержат пропуск слова или фразы в тексте. В таких тестах требуется вставить пропущенный текст: цифровой ответ решённой задачи; слово или словосочетание из определения; оператор или операцию алгоритма и проч. (см. пример 6).

Пример 6 – тест с записью пропущенного слова в нужном падеже:

«Оптимальное решение ЗЛП, если оно существует и если оно единственное, находится в _____ многогранника решений»

Ответ: вспомнив соответствующую теорему, студент определяет, что пропущенным словом является «вершине».

В заключение следует отметить, что автором данных тезисов было разработано и передано на кафедру прикладной математики для использования в системе Ilias порядка 50 тестовых вопросов по теме «Модели задач линейного программирования» дисциплины «Исследование операций».

Список литературы

1. Аванесов В.С. Знания как предмет педагогического измерения // Педагогические измерения. – 2005. - №3.
2. Методика составления и проведения тестов [Электронный ресурс]. – Режим доступа: http://xni1abbnckbmcl9fb.xnp1ai
3. Правила составления тестовых заданий [Электронный ресурс]. – Режим доступа: http://www.gumer.info/bibliotek_Buks/Pedagog/testing/4.4.php

КЛАССИФИКАЦИЯ БАНКОВСКИХ ПЛАТЕЖЕЙ НА ОСНОВЕ АНАЛИЗА ТЕКСТА НА ЕСТЕСТВЕННОМ ЯЗЫКЕ

Сироткин К.О. – студент
Алтайский государственный технический университет (г. Барнаул)

В настоящее время, при низких стоимостях хранения и скоростях обмена данными, человечество ежедневно накапливает огромное количество необработанных данных, значительную часть которых составляет текст: от переписок пользователей социальных сетей до записей о банковских транзакциях и логах интернет провайдеров. Текстовая информация может иметь различные виды: она может иметь стандартизированную или не стандартизированную форму, быть представлена числами, знаками или являться словами и предложениями, написанными на естественном языке.

Одной из актуальных проблем в области анализа текстовой информации является задача классификации и анализа банковских транзакций. Сегодня, крупные банки применяют технологии машинного обучения для анализа транзакции клиентов с целью выявления потенциальных случаев мошенничества и краж, а также для оптимизации денежного обращения, моделирования вероятности дефолта малого бизнеса и даже анализа и генерации

исковых заявлений. Большая часть информация о транзакциях хранится в строго определенном формате, что делает этап подготовки данных для обработки довольно простым. Несмотря на это, даже в таких данных могут присутствовать фрагменты текста, написанного на естественном языке, что значительно подготовку данных и их оценку.

Именно такая проблема решалась в ходе разработки системы классификации и контроля корректности платежей для ООО «Биллинговый центр», г. Барнаул.

Задача заключалась в классификации записей из банковской выписки со счёта организации по типам платежей, таким как поступление денежных средств от физического лица, оплата судебных издержек, возврат денежных средств, не владея информацией о том, кому принадлежат счета плательщиков. Таким образом, задача состояла в отнесении платежа к одному из классов исключительно на основе анализа поля «Назначение платежа», содержащего предложения на естественном языке.

Для решения задачи было использовано несколько подходов, первым из которых являлось распространенное в подобных ситуациях тематическое моделирование – способ построения модели коллекции текстовых документов, которая определяет, к каким темам относится каждый из документов. В роли документов выступали значения поля «Назначение платежа», а в роли тем – категории, по которым выполнялась классификация. В качестве алгоритмов тематического моделирования были взяты LDA (Latent Dirichlet Allocation – Латентное размещение Дирихле) и LSA (Latent Semantic Analysis – Латентно-Семантический Анализ). Результатами работы данных алгоритмов являлись списки ключевых слов и их весов, которые сигнализировали о принадлежности платежа к одной из категорий. Оценка данных списков показала, что алгоритмы придавали слишком большое значение терминам, употребимым в большинстве категорий, таким как названия населенных пунктов и части адресов, упуская более значимые для отдельных категорий, но редко употребимые термины.

Поэтому в следующем подходе было необходимо придать больший вес терминам, специфичным для различных категорий платежей. Для этого входные данные были предварительно обработаны регулярными выражениями, удаляющими знаки препинания и даты. Затем данные, для которых были заранее известны классы, к которым они относятся, были разбиты на триады (последовательные тройки символов) и для каждой из категорий были выбраны триады с наибольшей относительной частотой. Это позволило учесть специфику каждой из категорий.

Таким образом, была проведена подготовка данных, которая позволила представить текст на естественном языке в виде, подходящем для обработки алгоритмами машинного обучения. На следующем этапе планируется применить к подготовленным данным такие алгоритмы классификации, как Байесовский классификатор, метод k-ближайших соседей и метод опорных векторов. В зависимости результатов точности классификации тестовой выборке будет выбран один из данных алгоритмов.

Список литературы

1. 8 кейсов применения Machine Learning от Сбербанка [Электронный ресурс]. – М., 2016. – Режим доступа: http://futurebanking.ru/post/3213
2. Rami Ayadi, Mohsen Maraoui, Mounir Zrigui, LDA and LSI as a Dimensionality Reduction Method in Arabic Document Classification// ICIST. – 2015.
3. Открытый курс машинного обучения. Тема 7. Обучение без учителя: PCA и кластеризация [Электронный ресурс]. - М., 2017. – Режим доступа: https://habr.com/company/ods/blog/325654/

АВТОМАТИЗАЦИЯ КЛАССИФИКАЦИИ ВИДЕОЗАПИСЕЙ

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

Проблема классификации видео является актуальной проблемой в области компьютерного зрения. На данный момент не существует надежных алгоритмов, способных решать данную задачу на уровне, хоть отдаленно сопоставимым с человеческим. Кроме того, даже существующие алгоритмы в основном способны классифицировать видео лишь по ограниченному числу категорий. Для решения задач такого класса применяются три основных подхода или их комбинации. Такими подходами являются анализ субтитров, встроенных в видео или сгенерированных при помощи распознавателей речи, анализ звука, присутствующего на видео, и анализ графической информации – последовательности кадров видеопотока.

Одно из преимуществ текстовых подходов заключается в том, что они задействуют огромный опыт и знания, накопленные в результате исследований по анализу и классификации текстовых документов. Другой положительный момент состоит в том, что взаимосвязь между словами и определенным жанром видео достаточно очевидна. Недостатком метода является то, что диалоги в большинстве своем не описывают обстановку и происходящее в кадре, а следовательно, текста диалогов зачастую недостаточно для классификации видео. Второй недостаток — не у каждого видео есть встроенные субтитры, и не для каждого видео их можно сгенерировать – при отсутствии диалогов.

Подходы, основанные только на аудио, встречаются чуть чаще, чем текстовые. Один из их плюсов состоит в том, что аудио подходы требуют меньше компьютерных ресурсов, чем видео. Также если признаки (черты) необходимо сохранить, аудио требует меньше места, чем видео.

Так или иначе, самым широко распространяемым является метод анализа графической информации для классификации видео. На сегодняшний день самым успешным методом классификации графической информации (изображений) является применение сверточных нейронных сетей (CNN – англ. Convolutional Neural Networks). Они приобрели популярность после успеха в соревновании по классификации изображений ImageNet в 2012 году, когда сверточная нейронная сеть, ныне известная как AlexNet, уменьшила процент ошибок при классификации с 25% до 15%, в сравнении с предыдущим годом. Все последующие года были посвящены совершенствованию архитектуры нейронных сетей и увеличению их сложности, так в 2015 процент ошибок удалось свести к 3%.

Именно поэтому в данной работе был реализован подход, задействующий сверточную нейронную сеть, обученную на наборе данных CIFAR-10, что удалось избежать недостатков аудио и текстового подходов. Реализованная система с хорошей степенью точности классифицирует видео по одной из десяти категорий. Так большинство видео, на которых главным действующим объектом является один из объектов CIFAR-10, нейронная сеть выполняет корректную классификацию с вероятностью 50-70%. Естественно, при более качественном наборе данных для обучения с большим числом классов, можно переобучить нейронную сеть для выполнения классификаций по большему числу классов.

Список литературы

1. CoastlineAutomation [Электронный ресурс]. - Режим доступа: https://blog.coast.ai/five-video-classification-methods-implemented-in-keras-and-tensorflow-99cad29cc0b5 / Five video classification methods implemented in Keras and TensorFlow
2. Darin Brezeale, Diane J. Cook Automatic Video Classification: A Survey on the Literature // IEEE Transactions on Systems Man and Cybernetics Part C (Applications and Reviews). – 2008.

3. Mittal C. Darji, Dipti Mathpal A Review of Video Classification Techniques // International Research Journal of Engineering and Technology (IRJET). – 2017. - Vol. 4(6).
4. Prashant Shambharkar, Nirmit Srivastava, Alok Yadav, Aseem Sharma, Ankit Katiyar A Survey on Classification of Videos using Data Mining Techniques // International Journal of Computer Applications. - 2012.

ПРОЕКТИРОВАНИЕ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ ДЛЯ ДИРЕКТОРАТА СОРЕВНОВАНИЙ ПО ПРОГРАММИРОВАНИЮ

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

Каждый год на протяжении вот уже сорока двух лет (начиная с 1977 года) проводится международная студенческая олимпиада по программированию [1]. Победители и призеры данной олимпиады по праву считаются лучшими программистами, а крупнейшие IT-компании регулярно приглашают участников финала олимпиады в свои ряды.

Олимпиада — командное соревнование. Каждая команда состоит из трёх студентов. К участию допускаются студенты высших учебных заведений, а также аспиранты первого года обучения. Студенты, дважды участвовавшие в финальной стадии олимпиады, или пятикратно принимавшие участие в региональном отборе, не допускаются к участию. Есть ограничение по возрасту: участники старше 24 лет к участию не допускаются [2].

Неоспоримым фактом является наличие у участников данной олимпиады первоклассных навыков программирования, высокой скорости написания кода, умения быстро анализировать задачу, находить ее проблему и решение. Все перечисленное является большим преимуществом для выпускников при устройстве на работу.

Ежегодно на базе Алтайского Государственного Технического Университета им. И.И. Ползунова проводятся различные олимпиады и соревнования по программированию. В каждом из которых принимают участие множество школьников и студентов из разных городов Сибири, России и Ближнего Зарубежья.

С каждым годом проведения подобного рода соревнований их престиж растет и количество участников постоянно растет. Таким образом, целесообразно упростить и автоматизировать некоторые процессы в рамках проведения соревнований. Например, одним из таких процессов является процесс регистрации команд для участия в соревнованиях.

Учитывая то, что информация о соревнованиях и возможность регистрации на них участников должны быть доступны для всех желающих, целесообразно реализовать взаимодействие системы и пользователя посредством web-интерфейса (далее по тексту – сайта).

Рассмотрим основную функциональность проектируемой системы.

Со стороны пользователя сайта

Пользователи сайта осуществляют регистрацию команд на соревнования. Для того, чтобы иметь такую возможность необходимо иметь аккаунт в системе, регистрация которого осуществляется самостоятельно с указанием электронной почты. В дальнейшем на почту будут присылаться информационные сообщения, например, об успешной проверке регистрационных данных участников. Пользователю доступны для просмотра все команды, зарегистрированные на актуальные и прошедшие соревнования, а также тренера команд, руководители, сопровождающие и т.д. Если соревнование проводится в очной форме, то на

этапе регистрации обязательно указывается, будет ли человек лично присутствовать на соревнованиях.

Со стороны администратора сайта

Администратор сайта пользуясь собственным аккаунтом имеет возможность создавать соревнования, публиковать по нему информацию и материалы, а также решения, тесты, разборы и результаты после проведения соревнования. После завершения этапа регистрации участников организаторы формируют различные списки участников, команд, их сопровождающих и т.д.

В связи с тем, что с момента окончания регистрации и до начала проведения соревнований состав команды может измениться, необходимо предусмотреть возможность изменения регистрационных данных. Предоставленные данные проверяются на корректность организаторами соревнований. Статус проверки данных отображается на странице соревнования, а также в профиле пользователя, регистрировавшего участников. В случае внесения изменений в регистрационные данные они подлежат повторной проверке организаторами.

Кроме того, некоторая информация должна быть доступна для всех посетителей сайта. Например, перечень правил соревнований, инструкции по регистрации команд, полезные ссылки, страницы с информацией о соревнованиях, фотогалерея и т.д. Также в системе должен иметься функционал, позволяющий формировать различного вида списки участников, тренеров, руководителей и т.п., экспортировать регистрационные данные в формат XML для печати сопутствующих материалов.

Учитывая, что регистрационные данные являются ценной информацией, необходимо предусмотреть возможность бекапа и хранения архивов данных прошлых лет.

Исходя из вышесказанного, для реализации описанной системы выбран PHP фреймворк Symfony [3] версии 3.4, шаблонизатор Twig [4], база данных MySQL [5]. В качестве интегрированной среды разработки выбрана IDE PHP Storm [6] от компании JetBrains.

Одним из основных процессов в разрабатываемой системе является регистрация участников на соревнования. На рисунке 1 представлена соответствующая UML диаграмма последовательностей.

Рисунок 1 - Диаграмма последовательности регистрации участников соревнований

Таким образом, в рамках статьи были проанализированы основные требования к разрабатываемой системе, определен ее основной функционал, а также описаны выбранные

инструменты для реализации. На момент написания статьи имеется рабочий прототип системы, функциональность которого в дальнейшем будет постепенно расширяться, пока не будет реализован оптимальный функционал системы.

Список литературы

1. ACM ICPC [Электронный ресурс]. – Режим доступа: https://icpc.baylor.edu/
2. Международная студенческая олимпиада по программированию [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Международная_студенческая_олимпиада_по_программированию
3. What is Symfony [Электронный ресурс]. – Режим доступа: https://symfony.com/what-is-symfony
4. The flexible, fast, and secure template engine for PHP [Электронный ресурс]. – Режим доступа: https://twig.symfony.com/
5. The world's most popular open source database MySQL [Электронный ресурс]. – Режим доступа: https://www.mysql.com/
6. PhpStorm Lightning-smart PHP IDE [Электронный ресурс]. – Режим доступа: https://www.jetbrains.com/phpstorm/

ПРОГРАММНО-ТЕХНИЧЕСКИЙ КОМПЛЕКС КОНТРОЛЯ РЕЗУЛЬТАТИВНОСТИ ВЕЛОТРЕНИРОВОК

Попов А.А., Рогалева А.А. – студенты, Троицкий В.С. – к.ф.-м.н., доцент
Алтайский государственный технический университет (г. Барнаул)

Со времён изобретения велосипеда, индустрия не стоит на месте. С каждым годом компании выпускают на рынок множество гаджетов, позволяющих спортсменам более точно отслеживать свои физические данные и на их основании оптимально выстраивать тренировочный процесс.

Однако эти новомодные технические решения по карману далеко не всем. Цель данной работы – разработка бюджетного гаджета для занятий велоспортом, стоимость которого каждый сможет регулировать сам добавляя или убирая те или иные датчики.

Предполагается, что электронная часть гаджета собирается пользователем самостоятельно из модулей Arduino[1]. Необходимые крепежные элементы и сам корпус устройства распечатываются на 3D принтере по нашим моделям. Это позволит каждому подобрать функционал под собственные потребности. Программная же часть разрабатывается нами, включает в себя поддержку множества подключаемых модулей и способна работать с любой их комбинацией.

Велокомпьютер способен определять, накапливать и отображать на своём экране следующую информацию:

- текущая скорость;
- пройденное расстояние за тренировку;
- пройденное расстояние за все тренировки;
- суммарный подъем в гору за тренировку;
- средний угол уклона за тренировку;
- средняя скорость за тренировку;
- продолжительность тренировки;
- текущая дата/время;

- каденс (скорость вращения педалей в минуту);
- температура воздуха.

Предусмотрено взаимодействие велокомпьютера с внешними устройствами через Bluetooth[2] модуль. Поэтому дисплей также является опциональным модулем, который можно не подключать в целях экономии.

Стандартная комплектация велокомпьютера включает в себя:

- электронную плату Arduinonano или ее аналог на микроконтроллере ATmega 328;
- источник питания от 6 до 20 вольт(предпочтительно литиевый аккумулятор);
- плату зарядки TP4056;
- датчик атмосферного давления BMP180 или BMP280;
- датчик Холла 3144 или готовый модуль на его основе;
- дисплей;
- bluetooth модуль;
- провода, резисторы, тактовые кнопки;
- небольшой магнит;
- модуль для подключения micro SD карты памяти;
- модуль часов DS1302;
- корпус, напечатанный на 3D принтере.

Нами разрабатывается специальное приложение для смартфонов на операционной системе Android [3]. Оно позволяет использовать смартфон в качестве экрана(клавиатуры) велокомпьютера, позволяет подключать дополнительные датчики (например фитнес браслет), позволяет проводить расширенный мониторинг тренировки, строит графики результативности тренировок и основываясь на них выдает советы.

Одна из уникальных функций нашего приложения - построении тренировки с учётом ИМТ [4] пользователя на основе одного из выбранных направлений:

- снижение массы тела;
- поддержание массы тела.

При использовании данной функции приложение посылает сигнал на телефон или фитнесс браслет (если он в есть), с помощью которых пользователь может поддерживать необходимый темп тренировки, оставаясь в кардиозоне, соответствующей выбранному направлению.

Для реализации Android-приложения используется интегрированная среда разработки AndroidStudio и язык программирования Java. Прошивка велокомпьютера разрабатывается в ArduinoIDE на языке программирования C++.

На сегодня нами завершена разработана общей архитектуры программно-технического комплекса. Сформулированы требования к функциональности велокомпьютера, составлен список необходимых для этого аппаратных и программных компонентов. Готов проект размещения электронных плат и других элементов внутри корпуса устройства, разрабатывается 3D модель корпуса и крепежных элементов для печати на 3D принтере. Ведется разработка программного обеспечения.

Список литературы

1. Arduino [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Arduino
2. Bluetooth [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Bluetooth
3. Android [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Android
4. ИМТ [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Индекс_массы_тела

ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОГО И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ СИСТЕМЫ ПРИНЯТИЯ РЕШЕНИЙ ТЕРАПЕВТОМ

Шляхова С.К. – студент, Астахова А.В. – к.э.н., доцент
Алтайский государственный технический университет (г. Барнаул)

Данный IT-проект выполняется по заявке КГБУЗ «Городская поликлиника №7, г. Барнаул», чем и обусловлена его актуальность.

В рассматриваемом медицинском учреждении внедряется программный продукт «АРМ поликлиника», который, с точки зрения автора данных тезисов, а также с точки зрения специалистов поликлиники, требует значительных доработок. Одним из недостатков названного программного продукта является отсутствие информационного и программного обеспечения для врачей-терапевтов, позволяющего оперативно принимать решения по постановке и уточнению диагноза пациента, т.е. отсутствие экспертной системы (ЭС) принятия решений.

Экспертная система– это программный продукт, использующий экспертные знания специалиста для обеспечения высокоэффективного решения неформализованных задач в узкой предметной области (в нашем случае – медицине). Основу экспертной системы составляет база знаний предметной области, которая накапливается в процессе построения и эксплуатации ЭС.

Наглядным примером ЭС является «Домашний доктор» [1]. Это медицинская экспертная система, которая определяет характер заболевания, основываясь, в том числе, на ответах пользователя, полученных в результате диалога (см. рис.1).

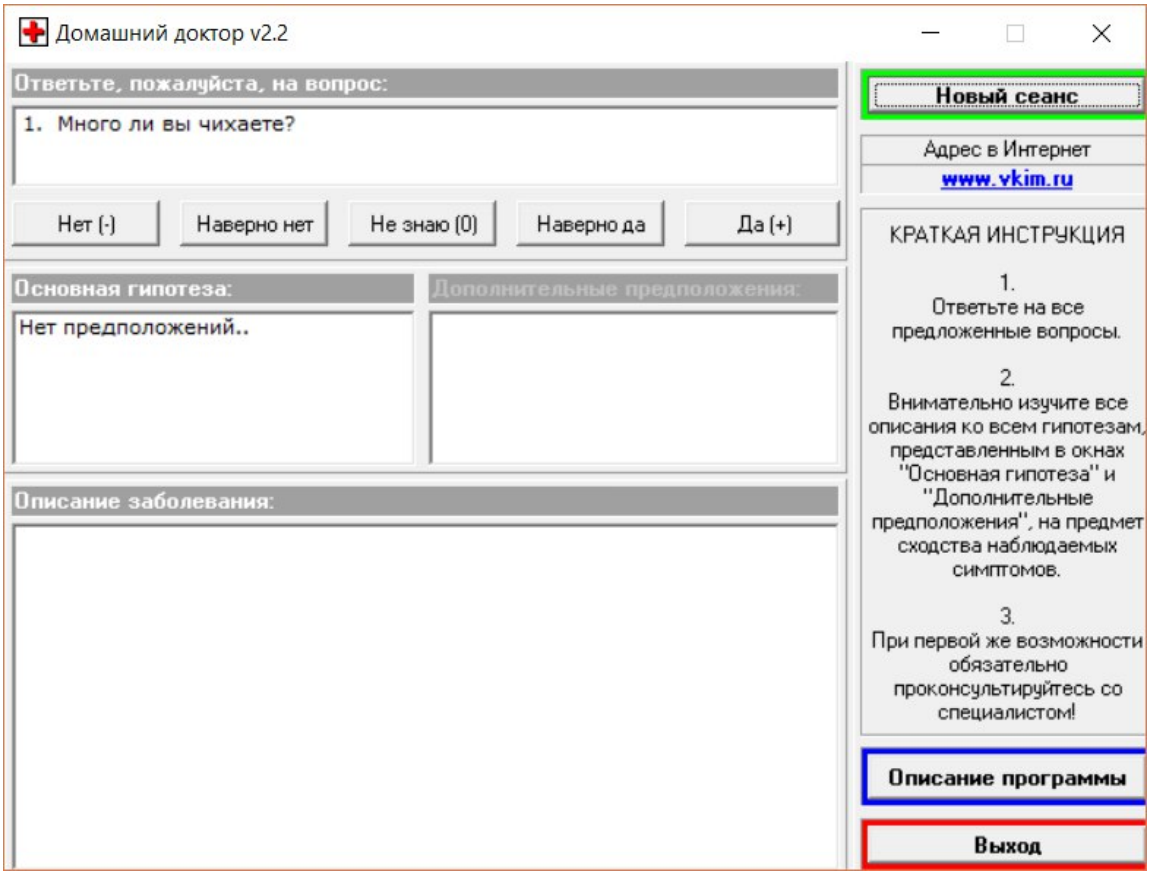


Рисунок 1 – Интерфейс экспертной системы «Домашний доктор 2.2»

Особенностью выбора ответов пользователя на вопросы системы является использование шкалы, содержащей значение параметра на интервале [-1;1]. Так, значение -1 означает «нет», значение +1 означает «да», значение 0 – «затрудняюсь ответить», на [-1;0] ответ соответствует фразе «скорее нет, чем да», на [0;1] «скорее да, чем нет». Очевидно, что шаг на шкале ответов можно уменьшить, тем самым уточнив варианты ответов. Данная особенность экспертных систем базируется на логике Заде, построенной на анализе нечетких множеств Заде Л. [2].

Ответы пациента на вопросы системы заносятся, как факты, в базу знаний (БЗ), которая помимо фактов предметной области, связанных с состоянием больного в данный момент времени, содержит правила вывода. Правила вывода строятся по принципу: «если…,то…» или «если…, то …, иначе…». База знаний используется в ЭС подсистемой, называемой «механизмом логического вывода», или системой принятия решений. Любое решение, формируемое ЭС, на запрос пользователя выдается с учетом двух требований:

- на доске объявлений пользователь должен видеть обоснование вывода (все посылки);
- рассчитывается и выдается на экран вероятность достоверности принятого системой решения.

Формирование базы знаний – один из самых трудоемких и сложных этапов построения реальной экспертной системы. Однажды сформированная экспертом БЗ в дальнейшем может пополняться в режиме реальной эксплуатации ЭС, в том числе, – путем использования в системе алгоритмов самообучения. Очевидно, что, чем полнее база знаний, тем выше вероятность заключения, сделанного экспертной системой.

Оценка возможностей ЭС «Домашний доктор» специалистом- терапевтом позволяет заключить, что данная ЭС имеет область использования не столько в лечебной практике терапевта, сколько в домашних условиях, с целью получения пациентом грамотных формулировок симптомов имеющегося недомогания организма, и/или выдачи предварительных рекомендаций, в том числе, перед посещением врача.

Для более полной картины рассмотрим еще одну экспертную систему под названием «MEDAI» [3]. Как следует из описания данного продукта, это интернет-приложение, которое позволяет с высокой достоверностью судить о том, что происходит с больным, как его дальше диагностировать и лечить, на основании жалоб, анамнеза и клинических анализов.

Оценка экспертом –терапевтом данного программного продукта, с точки зрения возможности использования его в практической работе, позволяет заключить, что данная ЭС не удобна для оперативной работы терапевта, т.к.:

- требует значительных затрат времени из-за неоправданно длинных диалогов;
- не выдает значение вероятности, с которой делает вывод;
- неэффективно организует формирование экранных форм по содержанию (наличие в формах неоправданно избыточной информации).

Следует заметить, что к различным ЭС пользователи предъявляют различные требования. Так, ЭС в геологии, заменяющие или облегчающие предварительные этапы геологоразведки могут потребовать у пользователя достаточно большое количество его времени, из-за большого объема БЗ и достаточно сложной системы вывода. ЭС врача специалиста должна работать в оперативном режиме и выдавать на доску объявлений монитора лишь объективную, с точки зрения специалиста, информацию. Другими словами, избыточность информации на экране монитора врача во время приема пациента недопустима.

Структура проекта типовой ЭС представлена на рисунке 2. Система включает в себя следующие шесть составных частей.

- **Пользовательский интерфейс.** Механизм, с помощью которого происходит общение пользователя с ЭС.

- **Рабочая память.** База данных о фактическом состоянии исследуемых пациентов, содержащая факты предметной области, используемые в правилах вывода.
- **Машина логического вывода.** Программный компонент, который обеспечивает формирование логического вывода (принимая решение на основе фактов и правил вывода и, используя рабочий список правил), располагая логически возможные решения по приоритетам и выдавая пользователю решение с наивысшим приоритетом.
- **Рабочий список правил.** Созданный машиной логического вывода и расположенный по приоритетам список правил, шаблоны которых удовлетворяют фактам, находящимся в рабочей памяти.

В рабочем списке правил возникают конфликты, если различные активизированные правила имеют одинаковый приоритет и машина логического вывода должна принять решение о том, какие из этих правил необходимо запустить.

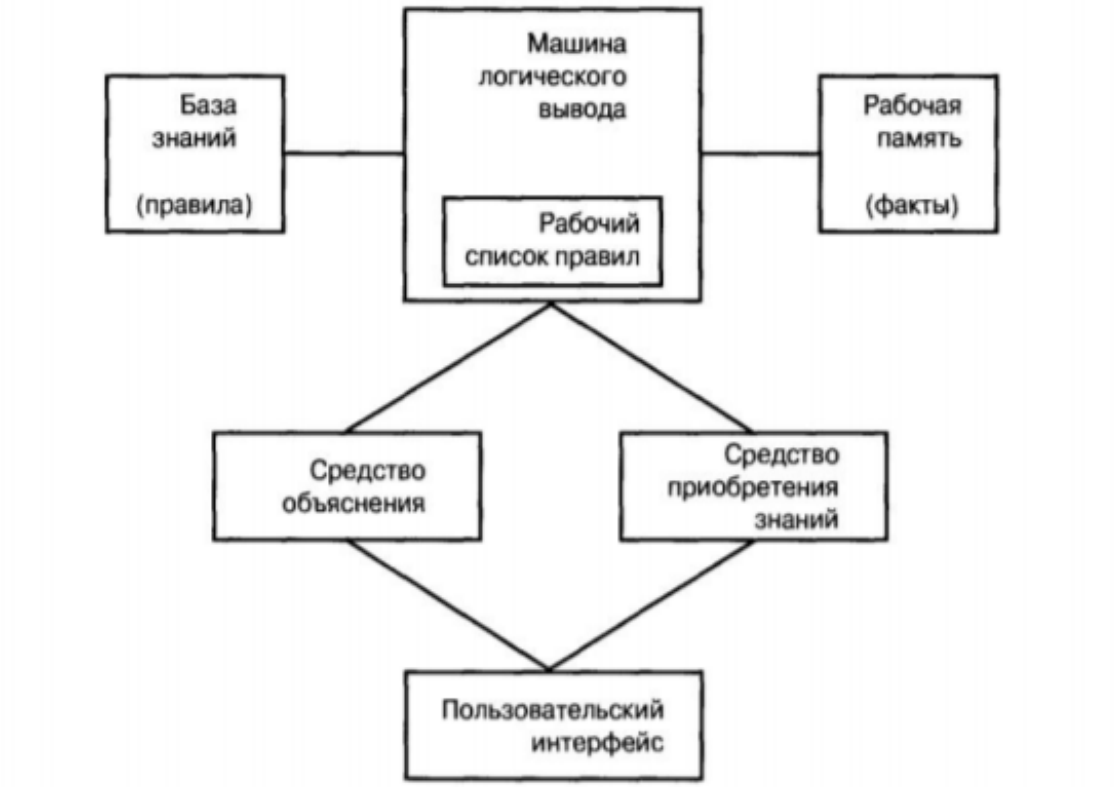


Рисунок 2 – Структура экспертной системы

- **Средство приобретения знаний**. Автоматизированный способ, позволяющий пользователю вводить знания в систему, а не привлекать к решению задачи явного кодирования знаний инженера по знаниям.

Данное средство не является обязательным элементом ЭС и может отсутствовать. Это инструментальное средство в некоторых экспертных системах способно обучаться, осуществляя вывод правил по методу индукции на основании примеров, и автоматически вырабатывать правила. Для выработки правил в машинном обучении применялись также другие методы и алгоритмы, такие как искусственные нейронные сети и генетические алгоритмы.

- **Средство объяснения.** Компонент, позволяющий объяснить пользователю ход рассуждений системы.

Главной особенностью ЭС является предусмотренное в ней средство объяснения, которое дает представление пользователю о том, как система пришла к определенному заключению. Система, основанная на правилах, способна легко ответить на вопросы о том, как было получено определенное заключение, поскольку хронология активизации правил и содержимое рабочей памяти можно сохранять в стеке.

Основное назначение любой ЭС – оперативное решение экспертных задач в узконаправленной предметной области. Например, ЭС для офтальмолога по диагностике глаукомы (а не по всем заболеваниям органа зрения). Напомним, что при этом ЭС выдает диагноз пациента для конкретного случая с определенной вероятностью.

Разработка АРМ терапевта со встроенным блоком принятия решений осуществлялась с учетом основных вышеописанных особенностей ЭС, но с одним принципиальным отличием. Это отличие состоит в том, что терапевт – это врач, имеющий широкую специализацию. Если разрабатывать ЭС терапевта, то получим базу знаний, которую на практике в оперативном режиме вряд будет целесообразно использовать. Как показывает практика работы, терапевту во время приема пациента требуется информационная поддержка в виде развернутого описания выявленного диагноза (диагнозов) и соответствующие варианты лечения. Решение принимает не информационная система, а сам врач, система лишь предоставляет информацию для принятия решений.

В этой связи предлагается в рамках АРМ терапевта в базе данных хранить для каждого пациента всю информацию о симптомах заболевания, с привязкой ее к фактическим датам и в базе знаний – хранить правила вывода для основных диагнозов.

В настоящее время автором данных тезисов разработана структура базы данных (см. рис. 3) и базы знаний (БЗ) системы принятия решений (СПР), разрабатывается ПО.

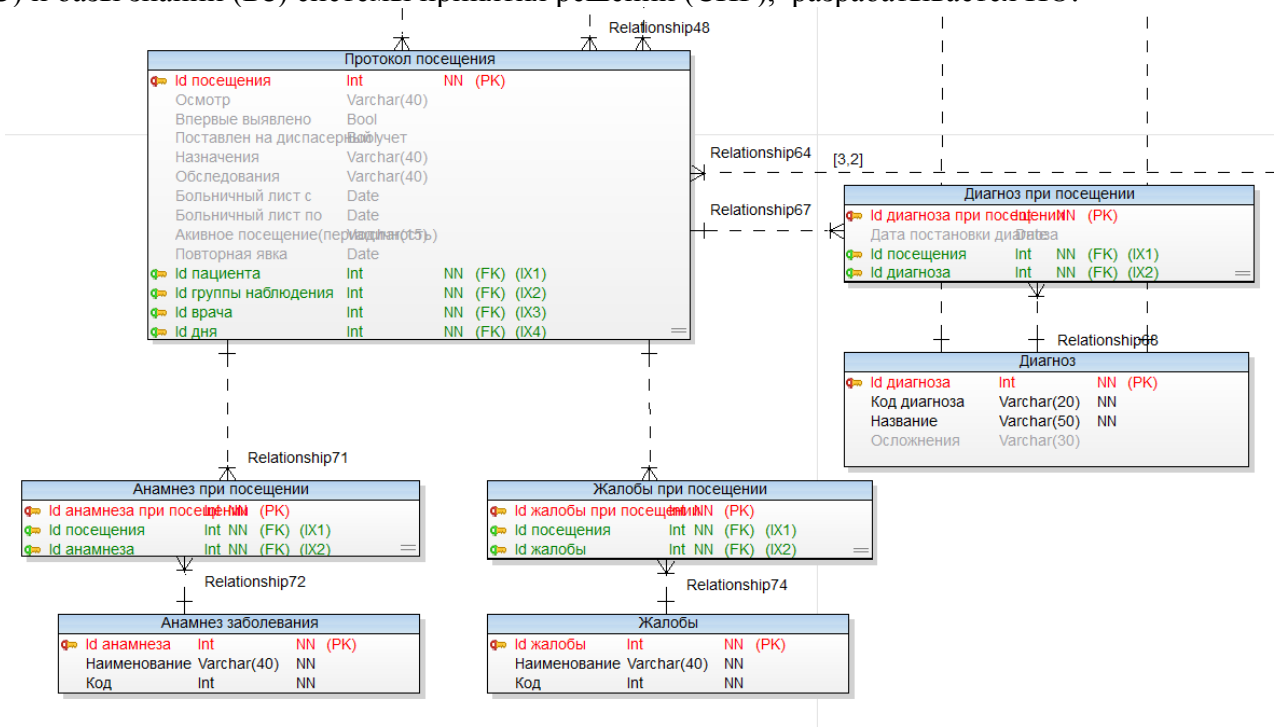


Рисунок 3 – Фрагмент базы данных СПР

Данная система не заменяет врача, а оказывает ему информационную помощь в постановке диагноза при условии, если врач обратился к этой системе. При ведении протокола приема, врач опрашивает пациента и заполняет поля, такие как: жалобы и анамнез заболевания, выбирая соответствующее из предоставленных справочников. Далее, нажимая кнопку «Поставить диагноз», врач при желании может обратиться к СПР.

Приведем пример входных данных, отображающих факты предметной области, используемые для постановки диагноза.

Жалобы пациента:

- сухость во рту, жажда;
- повышенная утомляемость;
- кожный зуд.

Анамнез:

- количество мочи заметно больше обычного (при сохранении или уменьшении питья);
- снижение тактильной чувствительности;
- снижение массы тела.

Приведенные здесь посылки используются в виде достоверных событий предметной области (фактов) и позволяют СПР из системы правил вывода БЗ выделить правила:

«сухость во рту, жажда» => Инсулинзависимый сахарный диабет +10%;
Гиперпаратиреоз +10%;
«повышенная утомляемость» => Инсулинзависимый сахарный диабет +10%;
Ишемическая кардиомиопатия +10%;
Туберкулез легких очаговый +10%;
«кожный зуд» => Токсокароз +10%;
Педикулез +10%;
Атопический дерматит +10%;
«количество мочи заметно больше обычного (при сохранении или уменьшении питья)» => Инсулинзависимый сахарный диабет +10%;
Гипокалиемия +10%;
«снижение тактильной чувствительности» => Недостаточность хрома + 10%;
«снижение массы тела» = > Инсулинзависимый сахарный диабет +10%;
Болезнь Аддисона +10%;
Эмфизема легких +10%;

В результате на выходе СПР врач получит следующую информацию6

Жалобы:

- сухость во рту, жажда;
- повышенная утомляемость;
- кожный зуд.

Анамнез:

- количество мочи заметно больше обычного (при сохранении или уменьшении питья);
- снижение тактильной чувствительности;
- снижение массы тела.

Диагноз:

Инсулинзависимый сахарный диабет с вероятностью 67% (4 из 6 пунктов принадлежат данному заболеманию):

+ сухость во рту, жажда;
+ повышенная утомляемость;
+ количество мочи заметно больше обычного (при сохранении или уменьшении питья);
+ снижение массы тела.

Таким образом, получив на вход описание задачи в виде списка жалоб и анамнезов, СПР интерпретирует данные, применяя правила вывода для каждого элемента из списка. В результате работы врач получает список подходящих диагнозов с определенной вероятностью, а также «ход рассуждений СПР». Аналогично описанной процедуре формируется список рекомендаций по лечению.

Следует заметить, что вероятность 67 % (и иная вероятность) не устроит на практике ни врача, ни больного. Поэтому логика принятия решений врачом при окончательной

постановке диагноза в СПР может «остаться за кадром». Если же БЗ достаточно полна, то вероятность верного решения, принятого системой, увеличивается.

Список литературы

1. Домашний доктор [Электронный ресурс]. – Режим доступа: http://www.aiportal.ru/downloads/expert-systems/home_doctor_2_2.html
2. Понятие лингвистической переменной и его применение к принятию приближённых рассуждений. - М., 1976.
3. MEDAI [Электронный ресурс]. – Режим доступа: http://medai.ru
4. Джозеф Джарратано, Гари Райли. Экспертные системы: принципы разработки и программирование. Пер. с англ. - М.: Издательский дом «Вильямс», 2006. - 1152 с.
5. Продеус А.Н., Захрабова Е.Н. Экспертные системы в медицине. - М.: Век +, 1998. - 361 с.

ПОСТРОЕНИЕ ПРОЦЕССА РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ НА ОСНОВЕ МЕТОЛОГИЙ MDD И SCRUM

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

В каждой компании, занимающейся разработкой программного обеспечения, настроен свой процесс разработки, который складывался годами. Но, к сожалению, даже спустя долгое время в процессах разработки есть существенные недочеты, причем каждый участник процесса выделяет свои. Рассмотрим проблемы, с которыми сталкиваются члены команды разработки.

Недостатки процесса разработки со стороны разработчика ПО:

- Нет четкой архитектуры продукта.

  Зачастую продукты начинают разрабатывать, руководствуясь девизом "как-нибудь сделаем", вследствие чего продукт начинает развиваться без четкой архитектуры и предметной области. Необходим человек, который будет заниматься проектированием архитектуры, которую в дальнейшем можно будет задокументировать и структурировать.
- Не всегда удается соблюсти четкое отделение бизнес логики от UI.
- Плохая масштабируемость.

Со стороны инженера по качеству ПО:

- В процесс разработки не включены инженеры по качеству.

  В коде присутствуют ошибки, которые могли быть закрыты при участии в разработке инженера по качеству.
- Нет карты тестирования.

  Зачастую тяжело уследить за всеми взаимосвязями и учесть их при тестировании функционала.

Со стороны менеджера проекта:

- Не всегда есть возможность получить четкую формулировку технического задания от клиента.

- Большая часть обсуждения задач происходит на словах из-за чего часто происходит недопонимание и задача реализуется не так как задумано.

Со стороны клиента:

- Документация по проекту не всегда актуальна.

Данные проблемы можно решить, если перестроить процесс разработки, но для этого нужно выбрать за основу подходящую методологию разработки ПО или их совокупность.

Рассмотрим следующие методологии разработки программного обеспечения:

- Scrum
- Kanban
- XP
- TDD
- MDD

Scrum

Это набор принципов, на которых строится процесс разработки, позволяющий в жёстко фиксированные и небольшие по времени итерации, называемые спринтами (sprints), предоставлять конечному пользователю работающее ПО с новыми возможностями, для которых определён наибольший приоритет. Возможности ПО к реализации в очередном спринте определяются в начале спринта на этапе планирования и не могут изменяться на всём его протяжении. При этом строго фиксированная небольшая длительность спринта придаёт процессу разработки предсказуемость и гибкость.

Scrum разбивает проект на части, которые сразу могут быть использованы Заказчиком для получения ценности, называемые заделами продуктов (product backlog). И несмотря на то, что «задел продукта» — достаточно верный перевод и используется в профессиональной литературе, в российской практике чаще всего используется просто «беклог». Затем эти части приоретизируются владельцем продукта – представителем Заказчика в команде. Самые важные «кусочки» первыми отбираются для выполнения в Спринте – так называются итерации в Scrum, длящемся от двух до четырех недель. В конце Спринта Заказчику представляется рабочий инкремент продукта – те самые важные «кусочки», которые уже можно использовать.

Основная структура процессов Scrum вращается вокруг пяти основных встреч: упорядочивания беклога, планирования спринта, ежедневных летучек, подведения итогов спринта и ретроспективы спринта.

Kanban

Метод управления разработкой, реализующий принцип «точно в срок» и способствующий равномерному распределению нагрузки между работниками. При данном подходе весь процесс разработки прозрачен для всех членов команды. Задачи по мере поступления заносятся в отдельный список, откуда каждый разработчик может извлечь требуемую задачу.

Канбан основан на четырёх основных принципах:

1. Опора на существующие методы разработки.

   Канбан начинается с существующих методов разработки и стимулирует в них дополнительные изменения.

2. Предварительная договорённость о проведении важных изменений.

   Команда разработчиков должна учитывать, что постоянные изменения — это способ улучшить существующий процесс разработки, однако проведение глобальных

перемен имеет большой риск. Канбан поощряет небольшие и эволюционные изменения.

3. Уважение к существующему порядку, ролям и обязанностям.
4. Поощрение инициативы: приветствуются проявления инициативы каждого разработчика.

Kanban намного менее строгий, нежели Scrum – он не ограничивает время спринтов, нет ролей, за исключением владельца продукта. Kanban даже позволяет члену команды вести несколько задач одновременно, чего не позволяет Scrum.

XP

Основными целями XP являются повышение доверия заказчика к программному продукту путем предоставления реальных доказательств успешности развития процесса разработки и резкое сокращение сроков разработки продукта. При этом XP сосредоточено на минимизации ошибок на ранних стадиях разработки. Это позволяет добиться максимальной скорости выпуска готового продукта и даёт возможность говорить о прогнозируемости работы. Практически все приемы XP направлены на повышение качества программного продукта.

TDD

TDD – это методика разработки через тестирование. Она заключается в организации автоматического тестирования разрабатываемых приложений путем написания модульных, интеграционных и функциональных тестов, определяющих требования к коду непосредственно перед написанием этого самого кода. Сначала пишется тест, который проверяет корректность работы еще ненаписанного программного кода. Этот тест, разумеется, не проходит. После этого разработчик пишет код, который выполняет действия, требуемые для прохождения теста. После того, как тест успешно пройден, по необходимости осуществляется рефакторинг (доработка и переработка) написанного кода, причём рефакторинг осуществляется под контролем прохождения тестов.

Эта методология позволяет добиться создания пригодного для автоматического тестирования приложения и очень хорошего покрытия кода тестами, так как ТЗ переводится на язык автоматических тестов, то есть всё, что программа должна делать, проверяется. Также TDD часто упрощает программную реализацию: исключается избыточность реализации — если компонент проходит тест, то он считается готовым.

MDD

Разработка, управляемая моделями – это такой стиль разработки программ, когда главными артефактами являются модели, а по ним генерируется код и другие прикладные артефакты. Модель – это описание системы с конкретной точки зрения, которое допускает пропуск несущественных деталей, чтобы интересующие характеристики были видны наиболее отчетливо.

В MDD модели используются не только как наброски или чертежи, но и как основные артефакты, из которых с помощью трансформаций эффективно создаются реализации. В MDD модели, ориентированные на предметную область приложения, являются основным ориентиром при разработке программных компонентов. Код и другие целевые решения генерируются при помощи трансформаций, разработанных с участием как экспертов по моделированию, так и экспертов в предметной области.

Генерация кода и других артефактов, связанных с платформой реализации, - это важная часть MDD, но автоматизация в стиле MDD делает гораздо больше. В проекте по разработке программного обеспечения должно производиться множество не относящихся к коду артефактов и многие из них могут полностью или частично создаваться на основе моделей.

Постановка задачи

На основе оценки и сравнения данных методологий было решено построить новый процесс разработки программного обеспечения, основанный на совокупности методологий разработки MDD и Scrum (рисунок 1).

За основу будет взята методология MDD. Первым этапом данного процесса будет обсуждение предметной области продукта, что требует бизнес и на какие части проект будет разбит, учтутся взаимосвязи между выделенными частями также, помимо этого, будут согласованны приоритеты для каждой из частей. На данном этапе и начинается проектирование архитектуры проекта.

Затем, следуя приоритету, обсуждается конкретная часть проекта, в зависимости от требуемого функционала она может быть разбита на более мелкие части, которые в свою очередь попадут в пул задач, которые будут реализовываться по методологии Scrum.

Часть задач попадет в спринт, оставшиеся будут лежать в бэклоге. Задачи из спринта будут обговорены более детально на планировании, будут обсуждены все нюансы и определено время необходимое для реализации данной задачи. Реализованные задачи будут протестированы вручную, а благодаря подходу MDD можно будет произвести генерацию интеграционных тестов, покрывающих все взаимосвязи реализуемых частей.

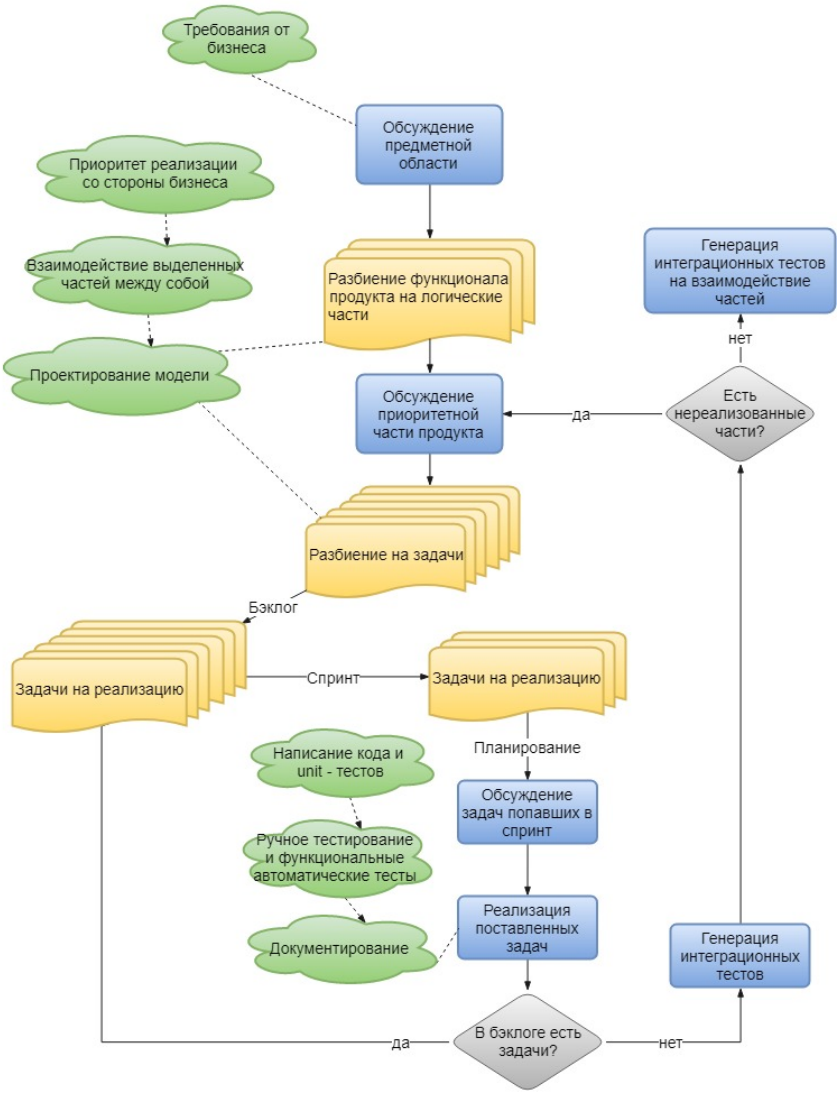


Рисунок 1 – Процесс разработки на основе MDD и Scrum

Тестирование будет производиться в три этапа:

1. Написание unit-тестов

   Почти что каждый метод, реализованный в коде, будет покрыт unit-тестами, которые будут отвечать за то, что в зависимости от входных данных будет возвращен верный результат. Лучше всего писать по одному unit-тесту на каждую ветвь метода.

2. Генерация интеграционных тестов

   В модели, созданной в начале разработки, будут видны взаимосвязи всех компонентов, которые мы будем воспринимать как граф со связями. Тесты будут генерироваться таким образом, чтобы был проход из каждой вершины графа – нашего модуля по всем достижимым ребрам – связи между модулями по одному разу.

3. Написание функциональных тестов

   Написание тестов инженером по качеству для проверки того, что пользователь получит ожидаемый результат после выполнения определенных действий.

При использовании данного процесса разработки будет разработан продукт с хорошей архитектурой, покрытый тестами и полностью задокументированный.

Список литературы

1. Шевелева А.Г., Старолетов С.М. Целесообразность применения методологии MDD в компании по производству программного обеспечения [Электронный ресурс] // XIV Всероссийская научно-техническая конференция студентов, аспиран- тов и молодых ученых "Наука и молодежь – 2017". Секция «Информационные технологии». Подсекция «Программная инженерия». – Барнаул: АлтГТУ, 2017. - С. 76-81. - Режим доступа: http://edu.secna.ru/media/f/pi2017v3.pdf
2. Шевелёва А.Г., Старолетов С.М. Построение процесса разработки и тестирования интеллектуальных систем обработки информации на основе методологии MDD [Электронный ресурс] // Ползуновский альманах. – 2017. - №4. –с. 99-101. - Режим доступа: http://new.elib.altstu.ru/journals/Files/pa2017_04_3/pdf/099sheveliov.pdf
3. M. Völter, T. Stahl, J. Bettin, A. Haase, S. Helsen. Model-Driven Software Development: Technology, Engineering, Management [Электронный ресурс]. - Wiley, 2006. - 446 p. - Режим доступа: http://www.voelter.de/data/books/mdsd-en.pdf
4. Старолетов С.М. Моделирование распределенных многокомпонентных программных систем и их тестирование на основе автоматных вероятностных моделей / С.М. Старолетов, Е.Н. Крючкова – Барнаул: Изд-во АлтГТУ, 2011. – 107 с. ISBN 978-5-7568-0859-9
5. Бек К. Экстремальное программирование: разработка через тестирование. - Питер, 2003. – 224 с.
6. IBM Rational Software Architect [Электронный ресурс]. - Режим доступа - https://www.ibm.com/developerworks/ru/library/mdd/ch1/ch1-pdf.pdf

ПРОЕКТИРОВАНИЕ НАДЕЖНОЙ АРХИТЕКТУРЫ МНОГОПОТОЧНОГО ТОРГОВОГО БОТА ДЛЯ КРИПТО-БИРЖ В УСЛОВИЯХ УЗКОГО КАНАЛА СОЕДИНЕНИЯ С БИРЖАМИ

Козлов Н.С. – магистрант
Алтайский государственный технический университет (г. Барнаул)

Введение

В связи с активным развитием финансового крипто валютного рынка, и его молодостью, инструментов автоматического или полуавтоматического анализа представлено мало. Большая часть требуемого функционала по взаимодействию с рынком либо не реализовано, либо является частной разработкой для какой-либо закрытой организации. Автоматическая торговля на молодом финансовом рынке, как например, рынок крипто валют находится на пике своей актуальности, в связи с простотой предсказуемости такого рынка.

Но сложность такого рода торговли затрудняется особенностями работы с Биржами, на которых проходит торговля. Многие биржи предоставляют API для работы со своими крипто активами, однако количество обращений к такой бирже и даже через API ограничено определенными партерами, такими как частота запросов или объём передаваемых данных. Кроме того, биржи обладают достаточно разными форматами работы с API (зачастую эти API плохо документированы) и при работе с каждой новой биржей приходится тратить большое количество времени на разбор и подготовку к использованию. Также, торговля на бирже часто предполагает некоторую совокупность операций, в результате которой можно получить прибыль, и чтобы уменьшить риски, такие операции должны проводиться единовременно, либо максимально близко по времени, что влечет за собой активное общение с API биржи. При излишне активном общении с биржей, либо по случайному стечению обстоятельств (видимо, сервера отвечающие за API бирж написаны несовершенно) некоторые запросы получают неожиданные, неправильные, некорректные ответы, а иногда вообще не получают ответа. Что вызывает большие затруднения при работе с Биржами.

Задача

Создать бота, позволяющего проводить торговые операции и операции перевода активов на биржах, а именно:

1. Выделение самых выгодных пар валют между биржами. Если конкретнее, то определение пары бирж, разница в курсах в паре на которых максимальна.
2. Выполнение операций торговли и перевода крипто активов на биржах и между биржами соответственно, некоторые из операций должны быть проведены одновременно.

В связи с вышеописанными затруднениями с работой с API бирж, реализация и первой и второй частей задачи становится достаточно сложной. Требуется аккуратная и тонкая работа с многопоточным программированием, кешированием информации, написание оптимального кода, для повышения производительности и уменьшения временных разрывов между операциями и модульная архитектура, для быстрого изменения частей системы (в связи с тем, что крипто рынок изменчив, многие средства анализа его поведения быстро устаревают). Эти требования влекут за собой достаточно сложный, но увлекательный процесс построения архитектуры соответствующего приложения.

Предлагаемая архитектура

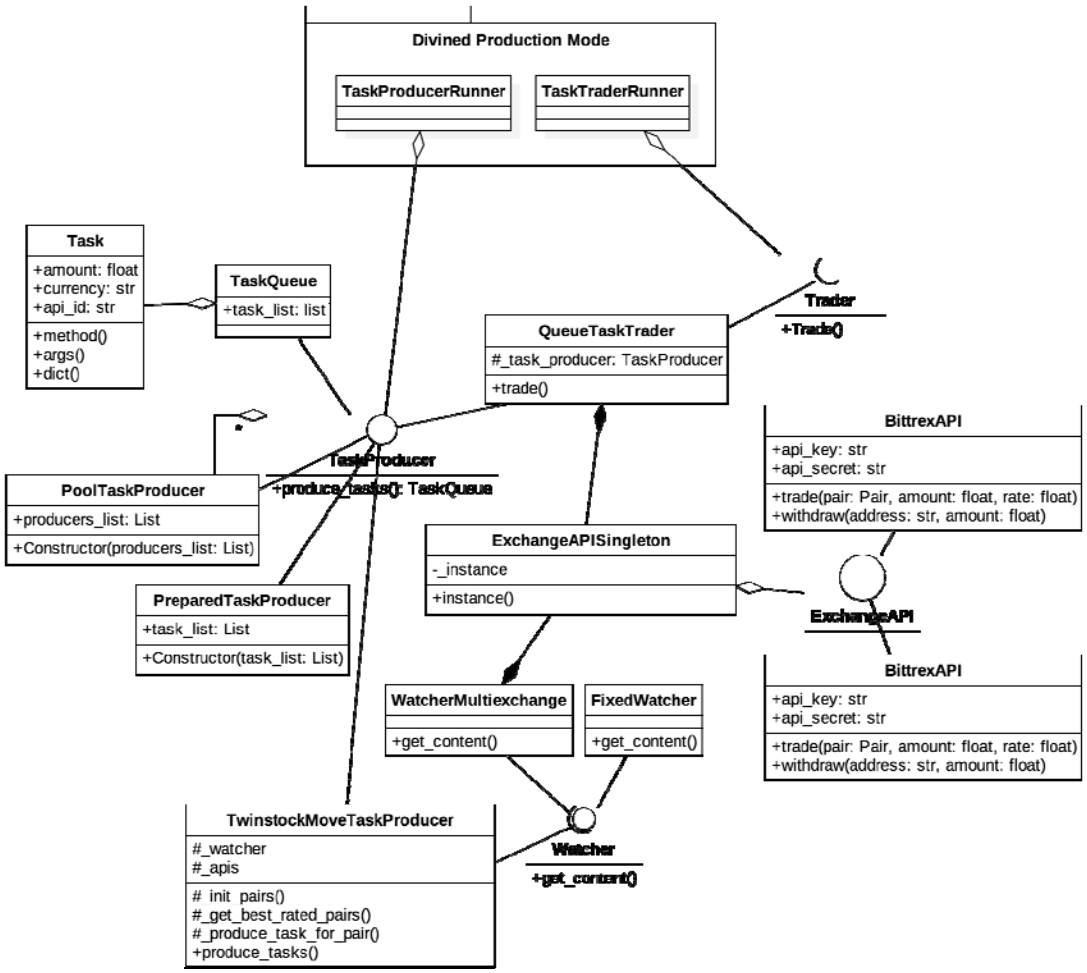


Рисунок 1 - Предлагаемая архитектура приложения

Основная идея этой архитектуры – параллельный доступ к API бирж и последовательный доступ к методам в рамках одного API. Приложение, построенное на такой архитектуре, подразумевает в себе наличие минимум трех крупных модулей, разграниченными интерфейсами, что позволяет легко изменять один модуль не меняя другие. Сами модули можно описать так:

1. Модуль доступа к API - модуль, отвечающий за API бирж, обязательно связывается с остальными через Singleton, поскольку все биржи предоставляют WebSocket API и создание большого числа таких объектов сказывается на быстродействии всей системы.
2. Модуль «Аналитики состояния бирж» - модуль, отвечающий за обработку данных на биржах и создание «задач» для торгового модуля бота. Очень важно отделить аналитику и принятие решений от выполнения операций на бирже.
3. Модуль «Торговли» на бирже либо биржах. Модуль, отвечающий за выполнение всех операций, созданных на этапе аналитики. Самый сложный модуль.

Рассмотрим приложение, целью которого является проведение межбиржевого арбитража, в котором операции по покупке или продаже валюты на одной паре на разных биржах будут проводиться одновременно.

Главная сложность такого бота в том, чтобы минимизировать количество обращений к API бирж и обрабатывать неожиданное поведение последних. Для работы с каждым API биржи выделяется отдельный поток, в котором проходят независимо все операции как при построении приложения, так и при выполнении итерации.

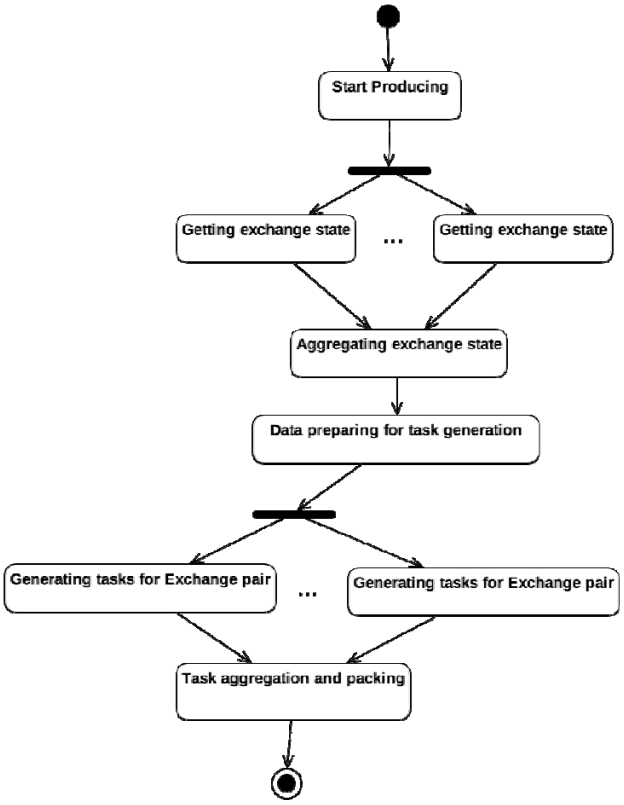


Рисунок 2 - Алгоритм выполнения итерации торговли

В первую очередь при выполнении итерации торговли приложение должно одновременно опросить все биржи запросом на получение Ticker'а – сводки состояния валют на бирже и их текущих курсов. После этого полученные данные обрабатываются уже однопоточно и в них ищутся пары бирж, разница в курсах, между которыми больше заданного порога. Все такие пары собираются в набор данных и проходят на следующую стадию обработки, где по выделенным парам запрашиваются «биржевые стаканы». Поскольку это дорогая операция, то параллельно у одной биржи стаканы запрашивать нельзя, иначе есть риск быть забаненным и не иметь возможность работать с этой биржей некоторое время, от нескольких минут до нескольких дней. На основе аналитики стаканов формируются задачи для торговли, например: купить на первой бирже определенное число крипто валюты по определенной цене и соответственно второе задание, которое зависит от первого – продать определенное число крипто валюты (которая покупается на другой бирже) по определенной цене, которая выше, чем при покупке.

Задачи, сформированные для нескольких бирж, распределяются и пакуются так, чтобы совершить сначала все независимые операции, затем все, операции, которые зависели только от уже выполненных, затем снова от выполненных и так, пока все задачи не будут распределены, таким образом достигается потокобезопасность выполнения задач. Выполнение задач возлагается на пул потоков.

В процессе выполнения задач, запросы к бирже проходят через один интсанс API для каждой биржи, что позволяет гарантировать выполнение и очередность выполнения запросов к API, что позволяет минимизировать некорректные реакции биржи на запросы и своевременно на них реагировать.

Для проведения экспериментов была реализована тестовая API биржи, оборачивающая реально существующие биржи, но позволявшая торговать на них без наличия крипто валют на счетах.

Выводы

В результате проведения экспериментов, торговый бот находил и выполнял сделки на «тестовой» паре бирж с разницей в курсе до 6% суммарным объёмом прибыли до 0.5 BTC за сутки с минимальным количеством некорректных реакций биржи на запросы. В качестве

стабильной и безопасной архитектуры для торговых ботов такой подход является очень хорошим.

Список литературы

1. Документация Threading API для языка Python 3.6 [Электронный ресурс]. - Режим доступа: https://docs.python.org/3/library/threading.html
2. Учимся писать многопроцессные и многопточные приложения на Python [Электронный ресурс]. - Режим доступа: https://habrahabr.ru/post/149420/
3. Global cryptocurrency benchmarking study [Электронный ресурс]. - Режим доступа: https://cointelegraph.com/storage/uploads/view/2017-global-cryptocurrency-benchmarking-study.pdf

РАЗРАБОТКА РАСШИРЯЕМОГО ИНТЕГРИРОВАННОГО МЕССЕНДЖЕРА

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

В современном мире мессенджеры являются важной частью нашей повседневной жизни. Это очень удобное средство общения, позволяющее обмениваться любым видом информации, будь то голосовое общение, текстовая переписка, изображения и видео, а также пересылка любых файлов. Благодаря таким преимуществам, постепенно пропадает необходимость пользоваться обычными голосовыми вызовами и SMS/MMS сообщениями. Достаточно иметь подключение к сети Интернет и мессенджеры могут не только заменить устаревшие способы общения, но и намного расширить возможности мобильного, и не только, общения.

С каждым годом создаётся всё больше и больше таких мессенджеров. Каждый пытается предложить что-то новое. Новый уровень безопасности, новые виды сообщений (например, стикеры, геолокация и т.д.), исчезающие статусы и прочее. Безусловно, мы, как потребители, от этого только выигрываем. Нам на руку растущая конкуренция среди мессенджеров, как раз благодаря такой конкуренции происходит интенсивное развитие сферы и внедрение новых технологий и функций в мессенджеры.

Однако есть и весьма существенные минусы у такого многообразия сервисов общения. Интернет, как средство коммуникации между людьми, изначально существует на открытых стандартах. Это касается всех его уровней, от физического и до прикладного. Например, протокол HTTP или группа протоколов, связанных с e-mail открыты и доступны каждому. Любой может создать свой веб-сайт или e-mail клиент. Вся информация для этого находится в свободном доступе и её количество просто огромно. В случае с мессенджерами ситуация иная. В конкурентной борьбе за аудиторию пользователей каждый сервис был создан изолированным от остальных. Каждая компания-разработчик сделала свою уникальную структуру API обмена сообщениями [6,7]. Мы не имеем возможности, скажем, из WhatsApp отправить сообщение собеседнику в Telegram. Проводя аналогию, можно представить, насколько было бы неудобно обмениваться электронной почтой, если бы, например, из почтового клиента Mail.ru нельзя было бы отправить письмо пользователю Gmail и наоборот. Или, например, отправлять sms-сообщения можно было бы только на телефон того же производителя, что и телефон отправителя. Именно такая ситуация произошла с мессенджерами. Каждый из нас вынужден иметь установленными на мобильном устройстве по 5 – 10 (а иногда и больше) различных приложений для обмена сообщениями, чтобы иметь возможность полноценно общаться со всеми.

По статистике [8] на начало 2018 года наибольшей популярностью в России обладает мессенджер WhatsApp – 59% пользователей смартфонов в России. Кроме него, в ТОП-5 популярнейших мессенджеров входят Viber и ВКонтакте – 36% и 32% соответственно, а замыкают пятёрку лидеров Telegram (19%) и Facebook (14%). Из этого можно сделать несколько выводов:

- Почти все пользуются несколькими мессенджерами.
- Несмотря на то, что есть явный лидер – WhatsApp, которым пользуется подавляющее большинство, доля каждого из остальных мессенджеров значительна, поэтому не представляется возможным отказаться от всех мессенджеров, кроме лидирующего, и при этом остаться «на связи» со всеми своими контактами.

Это действительно большая проблема, которая доставляет значительные неудобства пользователям.

Решением данной проблемы является приложение-агрегатор, объединяющее в себе мессенджеры. Программные решения, так или иначе реализующие это, уже есть. Рассмотрим и сравним некоторые из них.

Таблица 1

	OL Portal	Franz	All-in-One Messenger	Opera “Messenger in the sidebar”
Поддерживаемые платформы	Android, iOS	Windows, Mac, Linux	Любой браузер на ПК, поддерживающий расширения	Браузер Opera
Мобильное приложение?	Да	Нет	Нет	Нет
Количество поддерживаемых сервисов	11	65	29	3
Способ объединения	Вкладочный интерфейс			Выделенная область в основном конце браузера
Главное достоинство	Мобильное приложение	Количество поддерживаемых сервисов	Кроссплатформенность среди ПК	Удобство использования при малых размерах окна
Главный недостаток	Вкладочный интерфейс	Не является мобильным приложением	Устанавливается как браузерное расширение, но работает как отдельная программа	Количество поддерживаемых сервисов
Является ли полным решением проблемы	Нет	Нет	Нет	Нет

Из этого можно сделать вывод, что разработка мобильного приложения с расширяемой архитектурой для агрегации мессенджеров всё еще остается актуальной. Разработанное приложение выгодно отличается от рассмотренных аналогов по следующим позициям:

- Это мобильное приложение. А наиболее остро обозначенная проблема существует именно на мобильных устройствах.
- Все мессенджеры объединены на одном экране, диалоги из всех располагаются в одном списке, упорядоченные в хронологическом порядке.
- Возможность объединить два личных диалога из разных мессенджеров в один

(например, диалоги с Александром Александровым в WhatsApp и Telegram можно объединить в один диалог). В таком случае, сообщения из обоих диалогов будут отображаться вместе, на одном экране диалога и будут упорядочены в хронологическом порядке.

Разработанное приложение использует современный стек технологий, позволяющий сразу получить приложение для Android и iOS, а с незначительными изменениями исходного кода получить и веб-версию приложения. Были использованы следующие технологии и инструменты разработки:

- React – JavaScript библиотека для построения интерактивных пользовательских интерфейсов [2].
- React Native – JavaScript библиотека, позволяющая получить нативное мобильное приложение, используя JavaScript и React [3].
- Redux – библиотека, реализующая одноимённую идеологию разработки приложений; отвечает за управление состоянием приложения [4,5].

Реализация идеи лёгкой расширяемости приложения – добавления поддержки новых мессенджеров – основана на том, что каждый модуль, отвечающий за работу с API одного мессенджера, имплементирует один и тот же общий интерфейс. Этот же интерфейс имплементирует и класс, содержащий в себе экземпляры всех классов API. Таким образом, этот класс является удобной точкой доступа ко всем API.

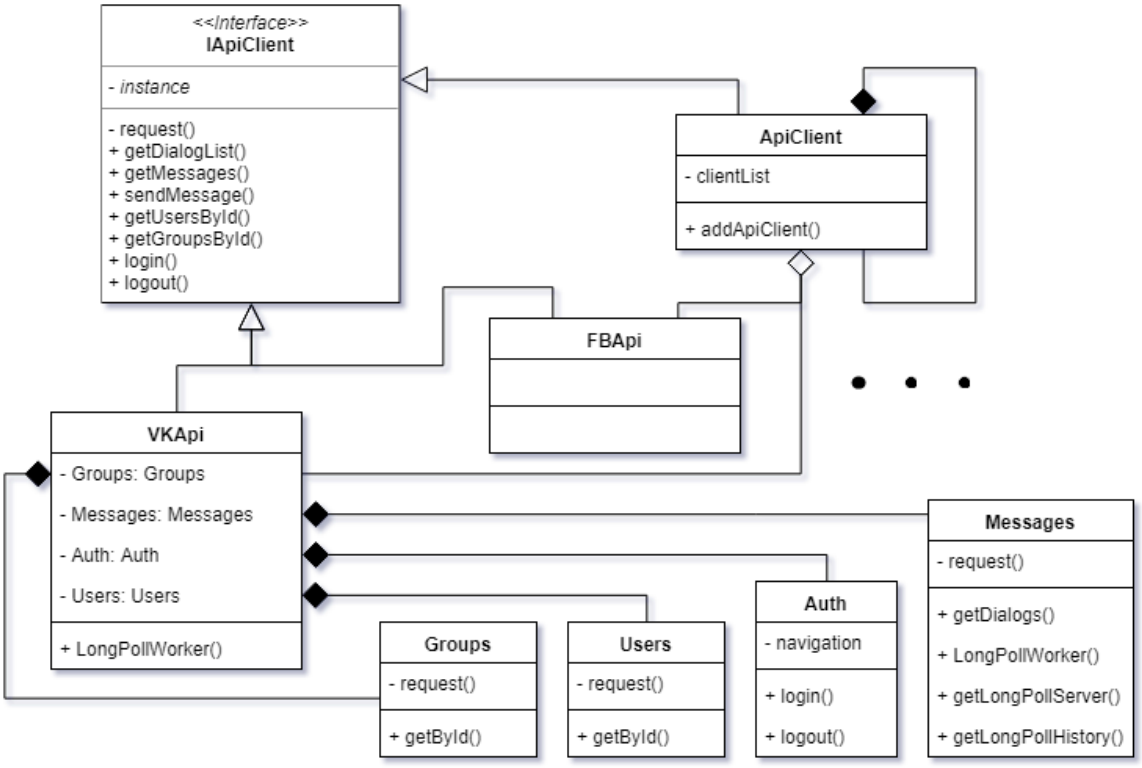


Рисунок 1 – Диаграмма классов API-модуля программы

В дальнейшем будет постоянно пополняться список поддерживаемых мессенджеров, а также будет реализована веб-версия приложения. Веб-версия имеет преимущество перед приведёнными выше решениями в том, что не требует установки дополнительных программ или расширений для браузера; не принуждает пользователя к использованию конкретного браузера и не отнимает пространство просмотра веб-сайтов. Для работы веб-версии приложения будет нужна лишь одна вкладка в браузере.

Список литературы

1. Почему ваш любимый мессенджер должен умереть [Электронный ресурс] // Хабрахабр. – Режим доступа: https://habrahabr.ru/post/272937
2. React - A JavaScript library for building user interfaces [Электронный ресурс]. – Режим доступа: https://reactjs.org
3. React Native ·A framework for building native apps using React [Электронный ресурс]. – Режим доступа: https://facebook.github.io/react-native
4. Redux [Электронный ресурс]. – Режим доступа: https://github.com/reactjs/redux
5. Архитектура модульных React + Redux приложений [Электронный ресурс]. // Хабрахабр. – Режим доступа: https://habrahabr.ru/post/326484/
6. Документация | Разработчикам – ВКонтакте [Электронный ресурс]. – Режим доступа: https://vk.com/dev/manuals
7. Платформа Messenger [Электронный ресурс]. – Режим доступа: https://developers.facebook.com/docs/messenger-platform
8. Статистика по использованию мессенджеров в России [Электронный ресурс] // Rusability. – Режим доступа: https://rusability.ru/news/opublikovana-svezhaya-statistika-po-ispolzovaniyu-messendzherov-v-rossii/

КОМПЬЮТЕРНАЯ РЕАЛИЗАЦИЯ КЛАССИЧЕСКОЙ НАСТОЛЬНОЙ ИГРЫ

Ушакова Е.О. – студент, Троицкий В.С. – к.ф.-м.н., доцент
Алтайский государственный технический университет (г. Барнаул)

Настольные игры созданы для того, чтобы объединить, сблизить людей, которые в нее играют. Что нужно для настольной игры? В основном это какое-то поле, на котором происходят все главные действия, обязательно правила, различные фишки кубики и прочие необходимые атрибуты. Также нужна компания – так как такие игры рассчитаны хотя бы на двоих игроков. Пусть все это есть. Началась игра и все подсчеты: выигрыши и проигрыши, различные бонусы и очки придется делать самим. Это утомительно. Кроме того, в настольную игру нельзя играть на расстоянии. А вот в компьютерная реализация настольной игры лишена этих недостатков. Игроки могут находиться в разных уголках планеты, все подсчеты ведутся компьютером, кроме того красочная анимация и музыкальное сопровождение сделает игровой процесс очень захватывающим.

Цель данной работы - это компьютерная реализация настольной игры. Концепция этой игры очень проста: есть замок, к которому нужно добраться, есть путь к нему. Путь состоит из клеток, одна клетка – одно передвижение фишки. Игрок, добравшийся первым до замка выигрывает. Рассмотрим подробности. В начале игры игрокам раздают по три карты. К моменту раздачи колода уже сформирована. Порядок карт в колоде на каждую игру формируется случайным образом. На руках не может быть больше восьми карт. Выбрать порядок хода игроков можно раскрутив рулетку. Рулетка раскручивается нажатием на нее мышью и останавливается в произвольный момент времени. На ней цифры от 1 до 5. У кого выпала наибольшая цифра – ходит первым. Чтобы сделать ход, игрок вращает рулетку и передвигает фишку на выпавшее количество очков. Для передвижения фишки можно использовать клавиатуру или кликнуть на нужную клетку мышью. Изначально все фишки стоят на первой клетке. Клетки могут быть разных цветов. Если фишка попадает на клетку синего, фиолетового, оранжевого или красного цвета игрок должен вызвать противника на поединок. Каждый игрок выбирает из тех карт, что есть у него на данный момент. Логично выбрать карту с большим необходимым параметром: Магия, Сила, Здоровье, Мудрость. В

поединке участвует параметр того цвета, на котором стоит фишка. Если игрок выигрывает в поединке (то есть его карта больше, чем карта соперника), игрок перемещает свою фишку вперед на количество клеток, указанное на его карте, если проигрывает - назад на количество клеток карты соперника. Ничья - игрок стоит на месте. После поединка игроки берут по одной карте из колоды (клик мышью на колоду). Далее делает ход следующий игрок. Если фишка попала на фиолетовую клетку, происходит кража карт у соперника (только если у соперника больше двух карт), то есть игрок забирает любую карту у соперника (карты лежат вверх рубашкой). Если фишка попадает на белую клетку, то нужно продолжить движение до следующей белой клетки. Если эта клетка ближайшая к замку, то фишка остается на месте. И черная клетка – возврат на ближайшую черную клетку. Если на клетке изображено сердце – нужно взять карту из колоды, если изображен цветок – положить карту в колоду. При попадании фишки на последнюю красную клетку перед замком, происходит поединок. Фишка должна стать на эту клетку, даже если на рулетке выпало число больше, чем нужно, чтобы дойти до замка (то есть игрок не сможет поставить фишку в замок без поединка). Фишка попадет в замок, если будет выигран поединок, иначе – возврат назад на указанное на карте соперника количество клеток.

На текущий момент придумана концепция игры, готов проект пользовательского и игрового интерфейса для 2D-реализации. Созданы все игровые объекты (карты, поле, фишки и т.д.). Ведется разработка программного обеспечения. В качестве среды разработки выбрана Intellij IDEA, в качестве языка программирования JavaFX. По нашим оценкам, срок завершения проекта весна 2019 года.

РАЗРАБОТКА И ВНЕДРЕНИЕ WEB-СЕРВИСА ДЛЯ ОРГАНИЗАЦИИ ВЗАИМОДЕЙСТВИЯ С СИСТЕМОЙ ПЛАТЕЖНОГО ПРОЦЕССИНГА С ПОМОЩЬЮ СМС-СООБЩЕНИЙ

Карсский Е.М. - магистрант
Алтайский государственный технический университет(г. Барнаул)

Актуальность

В настоящее время невозможно представить жизнь без телекоммуникаций в виде интернета или сотовой связи. Сотовую связь и SMS можно использовать для коммуникации между людьми, но это не единственная возможность применения данных технологии. Большое количество услуг можно получить имея в руках самый простой телефон с возможностью отправки SMS сообщений и звонков[1].

Используя SMS сообщения можно взаимодействовать с банковскими системами, производить переводы между счетами[4]. Запускать рекламные кампании среди постоянных клиентов магазина, банка или другого учреждения[5]. Часто встречаются клиентские сервисы с возможностью управления своими финансами через SMS сообщения.

Из всего сказанного выше можно сделать вывод то, что на данный момент SMS в большей степени широко используется в бизнесе. SMS остается недорогим и актуальным способом коммуникации по сей день.

Данный способ коммуникации можно применять и сфере принятия платежей. Для этого платёжному агенту необходим телефон с возможностью отправки SMS и специализированный сервис, который будет обрабатывать эти сообщения. Очевидным плюсом SMS является то, что не требуется покупки специализированного терминала, согласования места его установки, оплаты аренды за его установку и покупки специализированного программного обеспечения. Достаточно лишь создать сервис для обработки SMS и определить структуру сообщений.

Постановка задачи

Все SMS сообщения имеют определенную структуру. Далее текст SMS назовём командой. Каждая команда представляет собой набор параметров, которые разделены символами “#”,”*” или “.” Каждая команда начинается с ключевого символа, после которого идёт символ разделитель, остальные параметры.

В соответствии с тем, что сказано выше определён список команд, который представлен в таблице 1.

Таблица 1. Список команд мобильного терминала

Команда	Структура	Параметры
Оплата мобильной связи	[\d]+#[\d]+	номер телефона и сумма
Платеж по услуге	8#[\d\w\s]+#\d+#\d+#\d+	Идентификатор абонента, сумма, идентификатор услуги, номер абонента для отправки результата проведения платежа.
Проверка номера абонента	8#[\d\w\s]+#\d+#\d+	Идентификатор абонента, сумма, идентификатор услуги.
Проверка статуса платежа	2#\d+#\d+	Идентификатор плательщика и сумма платежа.
Отмена платежа	0#\d+#\d+	Идентификатор плательщика и сумма платежа.
Отчёт по платежам	1#XXXXXX#XXXXXX	Дата начала и конца отчетного периода. Если отсутствуют, то отчет формируется за текущий день.
Проверка баланса агента	3	Доп параметров нет.
Регистрация агента	9#\d+	Идентификатор родительского агента(может отсутствовать).

В качестве результата необходима отправка сообщения с текстовым описанием результата, которое будет понятно человеку.

Важным требованием является отказоустойчивость и возможность обработки большого числа сообщений.

Предлагаемое решение поставленной задачи

Для решения поставленной задачи предлагается разработка web-сервиса обработки SMS сообщений на основе протокола SMPP v3.4 [2,4].

На рисунке 1, представлен сценарии использования программного обеспечения конечным пользователем.

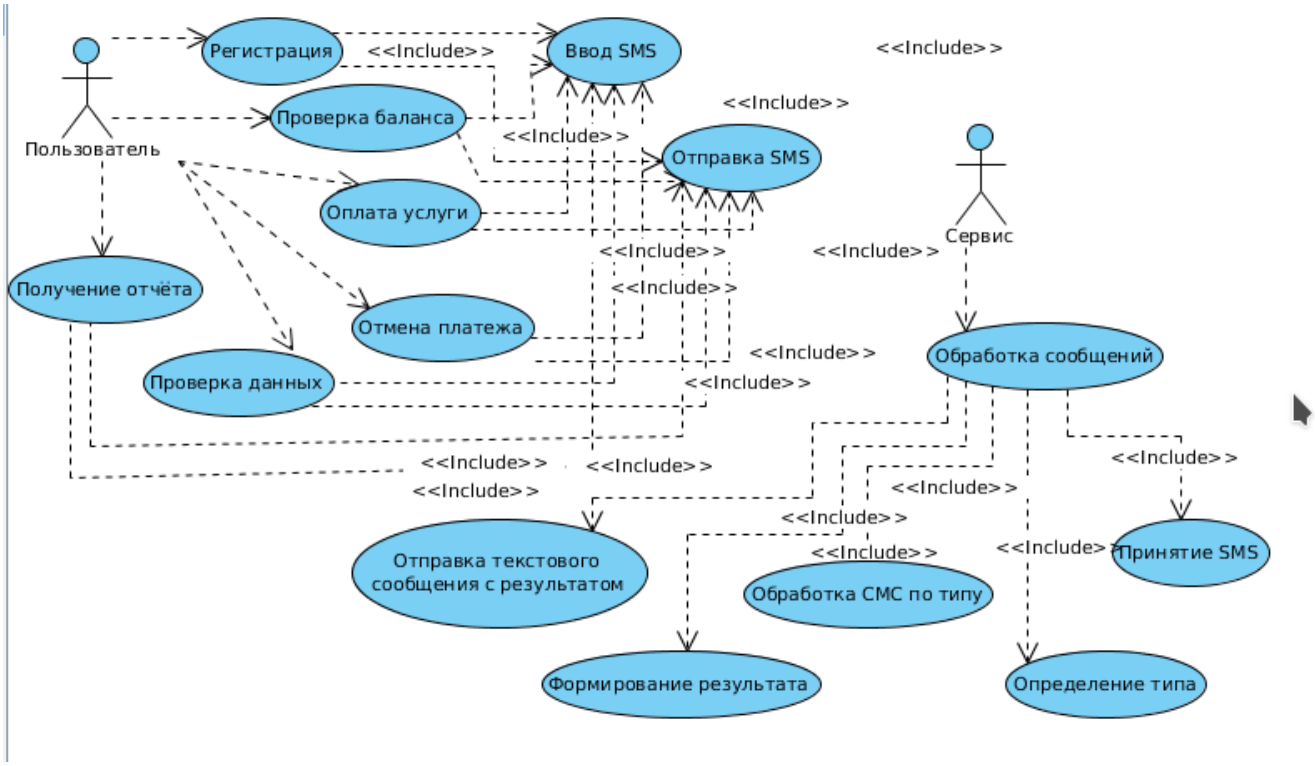


Рисунок 1 - Диаграмма использования сервиса

На рисунке 2 изображена динамическая схема работы сервиса.

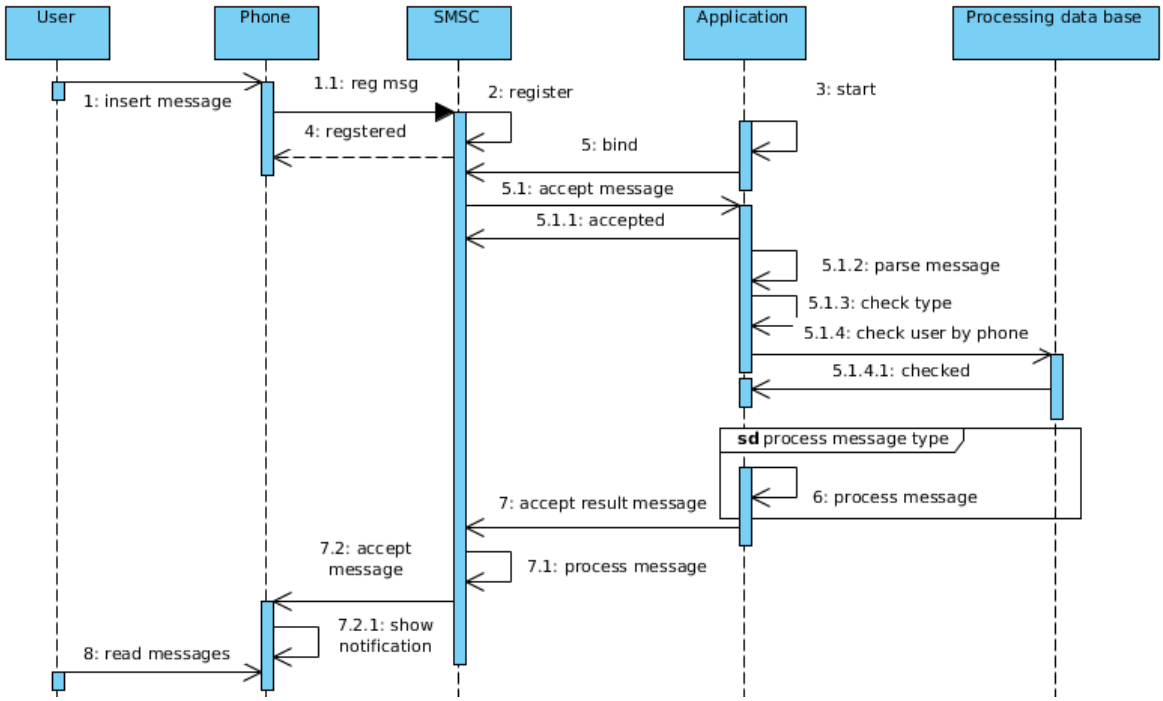


Рисунок 2 - Диаграмма последовательности действий сервиса

Заключение

Данный web-сервис позволяет производить практически все те же действия, что и платежный терминал со специальным программным обеспечением. Так как мобильный телефон имеется практически у каждого человека и каждый владелец сотового телефона может стать платежным агентом, то появляется огромная возможность расширения сети агентов практически до всего населения страны у которых есть телефоны. Как следствие рост оборота денежных средств и бизнеса.

В разработанном приложении учтены потребности в отказоустойчивости, локализации текстов SMS, проведено масштабное тестирование.

Отметим, что при проектировании приложения использованы паттерны “Дилегирование”, “Адаптер”, “Простая фабрика”, “Цепочка обязанностей”. Всё это позволит быстро реагировать на изменения рынка и новые требования, легко вносить новую функциональность.

Список литературы

1. Beeline USSD Инструкция [Электронный ресурс]. - Режим доступа: https://beeline.ru/customers/pomosh/question/voprosy-i-dokumenty/korotkie-komandi/
2. SMPP Protocol Specification v3.4. [Электронный ресурс] - Режим доступа: http://docs.nimta.com/SMPP_v3_4_Issue1_2.pdf
3. JSMPP: Mobile messaging for JAVA [Электронный ресурс]. - Режим доступа: https://github.com/opentelecoms-org/jsmpp
4. “900” USSD Руководство пользователя [Электронный ресурс]. - Режим доступа: https://www.sberbank.ru/common/img/uploaded/files/pdf/mob_ruk2.pdf
5. SigmaSmS API Documentation [Электронный ресурс]. - Режим доступа: https://sigmasms.ru/api/smpp/

ОТКАЗОУСТОЙЧИВАЯ СИСТЕМА РАСПРЕДЕЛЁННЫХ ВЫЧИСЛЕНИЙ НА ОСНОВЕ ТЕХНОЛОГИИ .NET

Проскурин Д.Ю. – магистрант
Алтайский государственный технический университет (г. Барнаул)

Введение

В настоящее время, сильно возросло количество информации, которую необходимо обрабатывать с минимальными задержками, практически в режиме реального времени. Также, появились вычислительные задачи, которые требуют большого количества вычислительных ресурсов. Зачастую, аппаратных мощностей одного компьютера может просто не хватать, чтобы обеспечить достаточную скорость вычислений. Для того, чтобы решить эту проблему, было решено разбить задачу таким образом, чтобы несколько компьютеров, решая каждый свою часть задачи и взаимодействуя друг с другом, могли в конце концов получить решение.

При решении подобного рода задач, часто возникает потребность в написании кода, который отвечает за взаимодействие вычислительных узлов между собой. Постоянная реализация одних и тех же типовых алгоритмов отнимает время разработчика, кроме того – это может приводить к ошибкам, что замедляет разработку приложения. Также, возникала потребность запускать приложение на компьютерах, работающих по управлением различных

операционных систем. Зачастую, чтобы этого добиться, приходилось реализовывать приложение с учётом специфики каждой операционной системы, в которой должно было работать приложение.

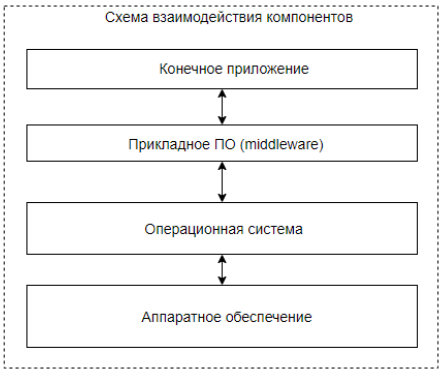


Рисунок 1 - Общая схема взаимодействия приложения и промежуточного слоя

Таким образом, возникла идея в создании промежуточного слоя абстракции, который бы брал на себя ответственность за работу с операционной системой, а также реализовывал некоторые типовые алгоритмы для удобства построения распределённых приложений.

Распределённые вычисления и системы

Обычно, для выполнения распределённых вычислений используют распределённые системы. Существует множество различных вариантов определения распределённых систем:

- система, в которой взаимодействие и синхронизация программных компонентов, выполняемых на независимых сетевых компьютерах, осуществляется посредством передачи сообщений;
- набор независимых компьютеров, не имеющих общей совместно используемой памяти и общего единого времени (таймера) и взаимодействующих через коммуникационную сеть посредством передачи сообщений, где каждый компьютер использует свою собственную оперативную память и на котором выполняется отдельный экземпляр своей операционной системы. Однако эти операционные системы функционируют совместно, предоставляя свои службы друг другу для решения общей задачи;
- спектр систем, от слабо связанных многомашинных комплексов, представляемых набором персональных компьютеров, объединённых в единую сеть, до сильно связанных многопроцессорных систем [1].

С точки зрения программирования – совокупность процессов, которые могут взаимодействовать между собой с помощью передачи сообщений. Каждый процесс имеет собственное состояние, представляемое набором данных и команд. Процесс не может вмешиваться в работу других процессов.

Чаще всего, вычисления в распределённых системах выполняются с помощью независимых процессов, которые взаимодействуют между собой посредством передачи сообщений. Из-за различия аппаратных характеристик компьютеров на которых выполняются процессы, а также географической удалённости их друг от друга, сообщения могут передаваться с непредсказуемыми временными задержками. Чтобы бороться с такими неопределённостями, существует две модели вычислительных систем: *синхронные* и *асинхронные*.

В *синхронных* вычислительных системах каждое действие ограничено минимальным и максимальным временем выполнения, т.е. доставка сообщений, выполнение какого-либо действия и отклонение локальных часов от точного времени имеет предсказуемую

погрешность. На практике, если алгоритм требует соблюдения этих временных характеристик, то при задержке сверх предполагаемой, могут произойти ошибки при вычислении результата.

В *асинхронных* распределённых системах никаких ограничений на передачу сообщений, отклонение системных часов и выполнение операций не накладывается. Таким образом:

- сообщение может быть доставлено через неизвестное, но конечное время после его отправки;
- операция может быть выполнена за конечное, но изначально неизвестное время;
- системные часы могут иметь произвольное отклонение, т.е. система на них не полагается.

Сеть Интернет построена на основе асинхронной модели вычислений.

Основные примитивы взаимодействий в распределённых системах

Базовыми примитивами для взаимодействия в распределённых системах является примитивы send и receive, которые позволяют отправлять и принимать сообщения. Примитив send должен указывать на процесс-получатель и данные которые должны быть отправлены, а receive – на процесс, от которого можно получать сообщения и буфер, куда будут помещены данные.

При реализации данных примитивов может возникнуть несколько проблем, например ситуация, кода данные переданы в команду send, а в следующей строке кода они изменяются.

```
var p=100;
send(receiver, p);
p=1;
```

Если рассмотреть псевдокод, приведённый выше, то возникает неопределённость – какое значение получит адресат: 100 или 1. Если команда send инициирует передачу данных с помощью подсистемы и вернёт управление в вызывающий поток, то значение в переменной p будет перезаписано, и получатель получит неверную информацию. Чтобы этого избежать, существует несколько вариантов.

Блокирующие операции send и receive без буферизации

В этом случае, возврат из команды send не осуществляется, пока не будет вызвана команда receive и данные не будут получены. Это подразумевает дополнительный обмен сообщениями о том, что связь установлена и данные можно передавать (handshake).

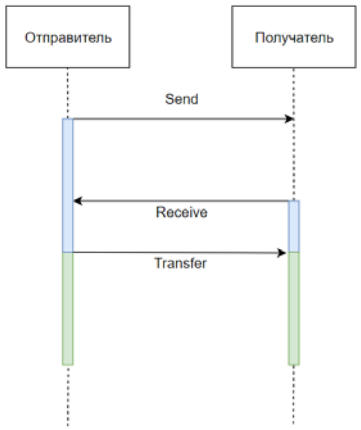


Рисунок 2 - Диаграмма блокировок при передаче сообщений без буферизации

На рисунке 2 представлена диаграмма взаимодействия процессов при обмене сообщениями. Синим цветом, обозначены ожидания установления соединения и вызова команды для получения данных, а зелёным – непосредственно передача данных.

Данный подход обладает несколькими недостатками. Если данный подход применять в асинхронной системе, то блокировка между вызовом send у отправителя и receive у получателя может быть существенной по времени. Также возможна полная взаимоблокировка процессов, например, когда получатель вызывает команду send, и отправитель вызывает команду send.

Блокирующие операции send и receive с буферизацией

Самым простым способом решения проблем, возникающих при обмене данными без буферизации, является использование дополнительных буферов на сторонах получателя и отправителя. Данные перед отправкой копируются в буфер отправителя, а затем управление возвращается вызывающему потоку. После этого, данные из буфера передаются по сети получателю и сохраняются в специальный буфер, независимо от того, вызвана ли команда receive.

Когда получатель вызывает команду receive, то вызывающий поток блокируется, проверяется буфер сообщений и сообщение копируется в адресное пространство вызывающего процесса.

Также, возможно применение только одного буфера на стороне получателя. В таком случае, после вызова команды send, вызывающий поток блокируется, данные передаются по сети и помещаются в буфер получателя (независимо от вызова команды receive), и управление возвращается в поток вызвавший команду send.

Введение дополнительных буферов, приводит к необходимости обработки ситуаций их переполнения. Возможны 2 варианта обработки такой ситуации: команда send возвращает ошибку, либо блокирует отправителя до тех пор, пока буфер не будет освобождён. Также, ограниченный размер буфера может привести к взаимоблокировке процессов, например, когда они оба вызывают команды send и размеры их буферов недостаточны для помещения данных.

Вывод

Таким образом, предпочтительнее реализовать примитивы взаимодействия send и receive с использованием дополнительных буферов у получателя и отправителя. Данное решение позволит системе не тратить время на ожидание готовности получателя к приёму данных. Программисту, для корректного функционирования системы, нужно будет реализовывать обработку ситуаций, когда данные по каким-либо причинам не могут быть помещены в буфер, чтобы избежать взаимной блокировки процессов.

Список литературы

1. Косяков М.С. Введение в распределённые вычисления. – СПб: НИУ ИТМО, 2014. – 155 с.
2. Таненбаум Э., Ван Стеен М. Распределенные системы. Принципы и парадигмы. – СПб: Питер, 2003. – 877 с.

СИМУЛЯТОР ФУТБОЛЬНОГО ТРЕНЕРА И СИСТЕМА ПРОВЕДЕНИЯ ВИРТУАЛЬНЫХ ФУТБОЛЬНЫХ СОРЕВНОВАНИЙ НА ОСНОВЕ ТЕХНОЛОГИИ БЛОКЧЕЙН

Басараб С.А., Лященко В.И. – студенты, Старолетов С.М. – к.ф.-м.н., доцент
Алтайский государственный технический университет (г. Барнаул)

Технология блокчейн и футбол

В наше время большую популярность набирает технология блокчейн, при этом акцент постепенно смещается с использования спекулятивных инструментов вроде электронных денег к разработке доверенных децентрализованных приложений. Блокчейн представляет собой четкую структурированную базу данных с фиксированными правилами построения цепочек блоков. Каждый такой блок хранит ссылку на предыдущий блок, транзакции и подписывается хеш-функцией, которая предотвращает подмену данных, мошенничество и т.д. Используется децентрализованное хранение копий блоков у узлов сети и алгоритмы достижения конценцуса для определения валидных блоков, что обеспечивает безопасность сделок внутри сети. На данный момент данная технология блокчейн находят все новые применения, начиная от использования в информационных системах крупных банков и заканчивая, например, Государственным земельным кадастром.

В то же время, технология блокчейн позволяет проводить транзакции и обменивать виртуальные деньги на товары, при этом можно расширить данные возможности и производить обмен на рабочие ресурсы, а точнее виртуальные рабочие ресурсы.

Известно, что что приносит прибыль, то продается и перепродается с высокой скоростью. Сегодня занятия спортом - большая часть досуга, не менее большая и наблюдение за выступлением любимых команд и спортсменов. Спортом номер 1 считается футбол и, по сообщениям ФИФА, объем международного рынка футбольных трансферов в 2017 году составил $6,37 млрд [1] (около 20 годовых бюджетов РФ). Следовательно, нужно искать возможности проведения таких трансферов (оплата за переход игроков между клубами) с использованием технологии блокчейн, ведь это позволит сделать такие трансферы прозрачными, безопасными, а разработчики могут получить неплохие комиссии за разработку и обслуживание такой системы. Особенно актуальна разработка такой системы в 2018 год - год Чемпионата мира по футболу в России, ведь внимание прессы приковано к этому событию.

Постановка задачи

В рамках данного исследования предполагается моделирование вышеуказанных задач путем создания виртуальных футболистов (“криптофутболистов”), которые имеют некоторые параметры, влияющие на исход матчей; создание виртуальных команд, применение тактических расстановок, симуляция матчей (режим футбольного тренера). Также виртуальных футболистов можно будет выращивать (оттачивать навыки на игре и тренировках), осуществлять трансферы (продавать и покупать) за криптовалюту. С использованием хорошего состава футболистов можно будет побеждать турниры, состоящие как из компьютерных команд, так и других виртуальных команд, созданных другими людьми.

Рассмотрим подробнее функционал и логику симулятора, а также вопросы с созданием и трансфером криптофутболистов.

Симулятор футбольного тренера

Симулятор – алгоритм или приложение, задача которого состоит в имитации управления каким-либо процессом или аппаратом. В нашем случае это имитация работы футбольного тренера команды, в котором пользователю предлагают попробовать себя в его роли и познакомиться с футболом поближе. Идея создания такого симулятора однозначно связана с

грядущим Чемпионатом Мира по футболу в России 2018 года. В первую очередь целью создания симулятора является привлечение большего внимания к данному событию людей всех возрастов и народностей. Главной целью этой игры является забить мяч в ворота соперника большее число раз, чем это сделает команда соперника в установленное время [2]. Мяч в ворота можно забивать ногами или любыми другими частями тела (кроме рук). Команда на поле выставляет состав из 11 игроков, один из которых вратарь. Еще есть 6 игроков на скамейке запасных для замены игроков на поле. Замен можно делать столько, сколько это позволено регламентом турнира или матча. Стандартное количество замен позволяет совершить данное действие только три раза.

Футбольный тренер – это лицо, отвечающее за функционирование футбольного клуба или национальной сборной. Тренер клуба несет ответственность непосредственно перед президентом клуба.

Наш симулятор будет иметь возможность покупать и продавать игроков, то есть будет трансферный рынок. Где за криптовалюту можно совершать финансовые действия. Так же будет предматчевое состояние, где можно применять игроков в состав, которые имеются у нас в команде. Будет возможность смотреть характеристику игрока (игровые параметры). Для команды можно выбирать расстановку на матч, назначать какого-то игрока капитаном. Ну и самое главное находить соперника по сети и играть матч своими командами. За победу в матче можно будет получить какое-то количество криптовалюты. А симуляция самого матча проходит с помощью алгоритмов и формул.

Логика симуляции матча

Каждый игрок имеет свои параметры:

- Скорость
- Дриблинг
- Удар
- Защита
- Передачи
- Физическая сила

Так же есть запас сил. По этим параметрам высчитывается рейтинги атаки игрока, защиты и владения мячом (в середине поля). Формулы (1), (2), (3) имеют эмпирически подобранные взвешенные коэффициенты.

Расчет рейтинга для атаки:

$$\text{Рейтинг атаки игрока} = \text{Скорость}*0.2 + \text{Дриблинг}*0.2 + \text{Удар}*0.3 + \text{Защита}*0.05 + \text{Передачи}*0.15 + \text{Физическая сила}*0.1 \quad (1)$$

Расчет рейтинга для защиты:

$$\text{Рейтинг защиты игрока} = \text{Скорость}*0.05 + \text{Дриблинг}*0.05 + \text{Удар}*0.05 + \text{Защита}*0.4 + \text{Передачи}*0.1 + \text{Физическая сила}*0.35 \quad (2)$$

Расчет рейтинга для владения мячом:

$$\text{Рейтинг владения мячом игрока} = \text{Скорость}*0.15 + \text{Дриблинг}*0.1 + \text{Удар}*0.05 + \text{Защита}*0.1 + \text{Передачи}*0.4 + \text{Физическая сила}*0.2 \quad (3)$$

Все эти рейтинги рассчитываются для полного запаса сил, и во время матча этот запас уменьшается, значит со временем будут меняться (линейно уменьшаться) и рейтинги игрока.

Рейтинги игроков, в свою очередь, в сумме составляют рейтинги команд. Рейтинг команды позволят узнать вероятность успеха при совершении какого-то из действий. Чем

одна команда сильнее другой, тем больше успех на совершения события в матче. Но при этом нужно учитывать еще и вариант случайности (ведь все совершают ошибки, команды могут играть на кураже, идти вперед за болельщиками либо за лидерами). Полная вероятностная модель игры очень сложна, поэтому для простоты можно считать большинство событий в игре исходя только из рейтинга команды с внесением случайных поправок.

Если одна команда атакует, то значит другая обороняется. И если успеха в атаке не было, то команды меняются ролями. Теперь первая обороняется, а вторая атакует. А если конкретнее, то команде сначала нужно защититься, пройти середину поля (линия владения мячом), а потом только атаковать. Если всё успешно, то команда забивает гол.

Расчёт рейтинга команды зависит от расстановки, так как игроки на своих позициях по-разному вовлечены в атаку, во владение мячом и в защиту. Например, центральный защитник на 99% отрабатывает в обороне команды, на 70% участвует во владении мячом, а в атаке задействован всего лишь на 20%. Рассмотрим это на примере расстановки 4-4-2 (рисунок 1).

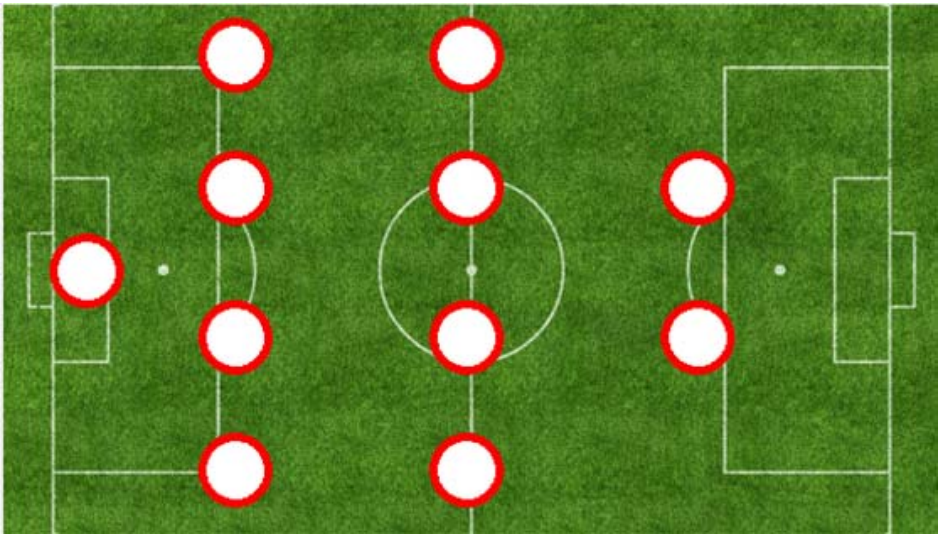


Рисунок 1 – Вариант расстановки 4-4-2

На рисунке 2 в процентах указана вовлеченность игроков в **защиту команды.**

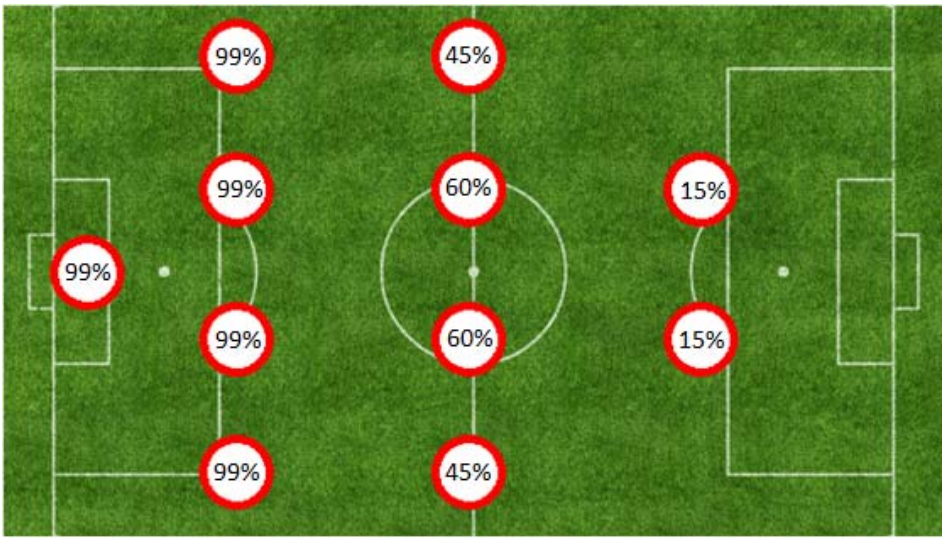


Рисунок 2 – Вовлеченность игроков в защите

То есть, для подсчета рейтинга команды, рейтинги каждого игрока делятся на 100% и умножаются на процент вовлеченности. А потом получившиеся рейтинги складываются. Это и получится рейтинг команды в защите. И если рейтинг защиты выше, чем рейтинг атаки у соперника, то больше вероятность что наша команда перехватит мяч. Точно так же с владением мячом и атакой, только процент вовлеченности в атаку команды и игроков на своих позициях будет другой.

На рисунке 3 в процентах указана вовлеченность игроков в **атаку команды**.

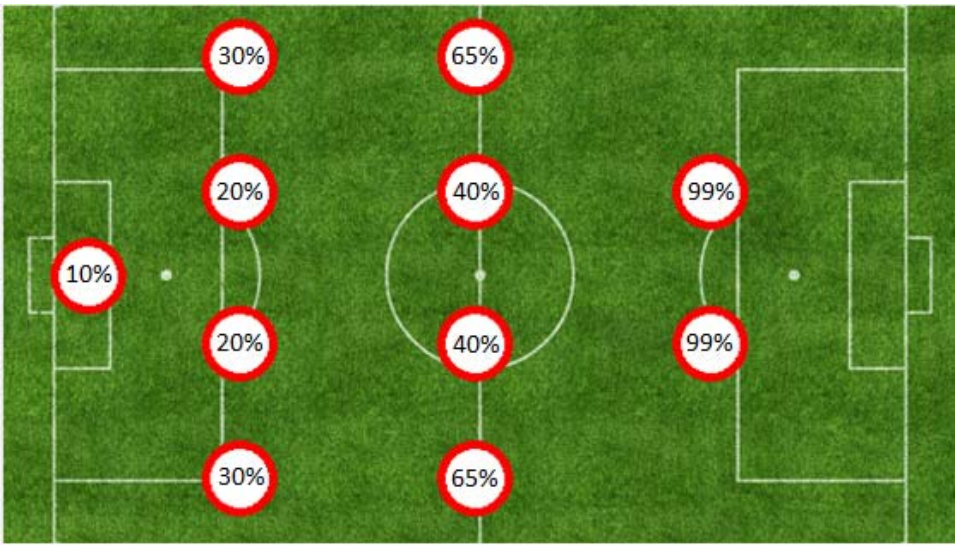


Рисунок 3 – Вовлеченность игроков в атаке.

При успешном завершении атаки команда забивает гол. На рисунке 4 в процентах указана вовлеченность игроков во **владение мячом командой.**

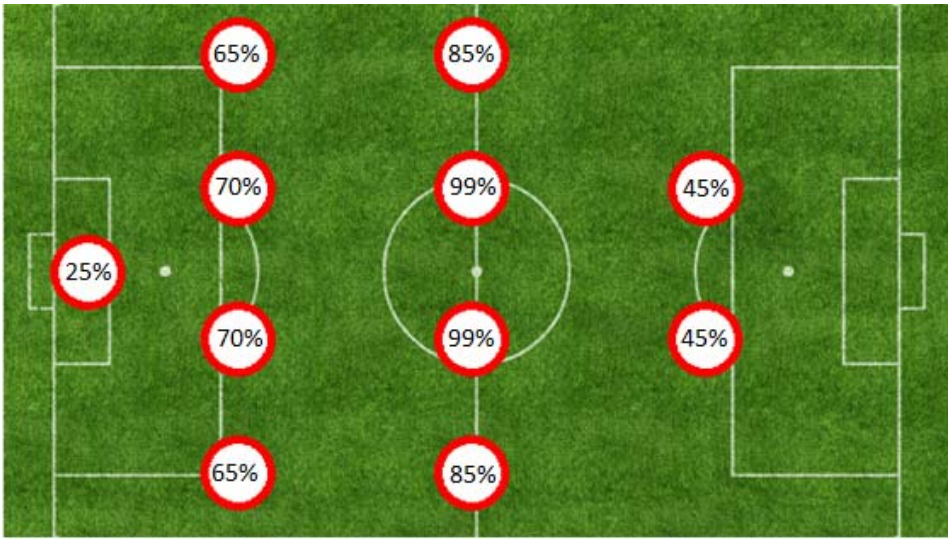


Рисунок 4 – Вовлеченность игроков во владение мячом.

Под владением мячом подразумевается переход от защиты в атаку, при этом команда соперника может перехватить мяч.

С помощью рейтингов команды можно построить баланс матча. Который на примере будет выглядеть, как показано на рисунке 5.

Рисунок 5 – Баланс матча в некоторый отрезок игры

Это пример атаки первой команды, у которой рейтинг больше, чем защита у второй команды. Таким образом вероятность забить гол больше. И при выборе случайного числа от 0 до 100 будет решаться исход данного события. При переходе от защиты в атаку сравниваются рейтинги владения мячом команд.

Состояние разработки

На данный момент реализована часть приложения - возможность играть виртуальный матч между двумя командами.

Для этого пользователю(тренеру) нужно выбрать расстановку команды. И если он готов, то начать симуляцию матча. В ходе матча идёт время и могут происходить какие-либо события. Игроки могут забивать голы или автоголы, получить желтые или красные карточки. Все эти события учитываются и отображаются в приложении. Так же можно производить замены игроков на поле, узнавать эффективность команды и вести текстовую трансляцию матча. Так же можно выйти из игры, но при этом мы получим техническое поражение. Все события пока реализованы по нажатию кнопок, которые находятся в окне приложения симулятора футбольного тренера.

Предполагаемая работа с криптофутболистами

Сами криптофутболисты будут описаны в виде Smart-контрактов платформы Etherium по аналогии с сервисом CryptoKitties [3].

Каждая команда будет владеть футболистами и в результате игр они будут менять свои параметры. Футболист описывается как контракт в виде набора свойств и алгоритмами их изменения. Он может быть выставлен на трансфер и перейти в другую команду за криптовалюту. Виртуальный тренер тратит деньги за покупки игроков, выручает их за продажи и выигранные матчи. Таким образом, наше исследование на выходе получит децентрализованное приложение, скрывающее за собой распределенную базу данных футболистов в виде набора их навыков. Планируется развернуть экземпляр приложения в тестовой сети и наладить обмен с реальной криптовалютой, имеющей небольшую цену, например, DogeCoin.

Список литературы

1. RNS информационное агенство [Электронный ресурс]. - Режим доступа: https://rns.online/sports-economy/Mirovoi-rinok-futbolnih-transferov-za-god-viros-na-327--do-637-mlrd-2018-01-30/
2. IFAB. Laws of the Game 2017/18 [Электронный ресурс]. - Режим доступа: http://www.fifa.com/mm/Document/FootballDevelopment/Refereeing/02/90/11/67/Lawsofthegame2017-2018-EN_Neutral.pdf
3. Proglib. Криптовалюта и CryptoKitties [Электронный ресурс]. - Режим доступа: https://proglib.io/p/blockchain-cryptokitties/

ИСПОЛЬЗОВАНИЕ БИБЛИОТЕКИ UNIVERSE ДЛЯ ПОСТРОЕНИЯ ДЕРЕВЬЕВ ПОВЕДЕНИЯ ПЕРСОНАЖЕЙ В ИГРОВЫХ МИРАХ

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

Введение

В статье предлагается использовать библиотеку Universe для создания модели поведения игрового персонажа в виде нейронной сети, строящей дерево поведение. Universe – это платформа для создания и обучения игровых нейронных сетей. С помощью библиотеки OpenAI Gym существует возможность сравнения и разработки различных алгоритмов обучения нейронных сетей для игровых персонажей.

Актуальность

На сегодняшний день в игровой индустрии все чаще возникает необходимость создания разумного игрового персонажа, способного к нестандартным решениям во время игры. Пользователям надоедают компьютеры-противники действующие по одной схему снова и снова. Для решения проблемы стали создаваться модели поведения в виде нейронной сети. Нейронной сетью – называют математическую модель, построение которой основано по принципу организации и функционирования биологической нейронной сети. Основная сложность, при разработке нейронной сети, это её обучение. Для обучения вам понадобятся реальные данные и тесты. После определенного количества итераций, обучения нужно будет закончить, но если мы захотим внести изменения в нейронную сеть, то нужно будет заново переучивать сеть и так каждый раз при внесении правок. В этом и заключается основной недостаток данной архитектуры. Второй немаловажный недостаток, это невозможность предугадать поведение или логику объекта в конкретный момент времени, хотя он является и в тоже время плюсом.

Рисунок 1 – схема архитектуры

Решение проблемы

Для создания и обучения нейронной сети можно воспользоваться открытой универсальной платформой Universe. Universe – это программная платформа для создания и обучения искусственного интеллекта на основе игр и других приложений. Данная библиотека распространяется бесплатно и имеет открытый исходный код на Python. С помощью платформы Universe возможно любому человеку обучать и оценивать алгоритмы нейронной сети в неимоверно широком диапазоне критериев и ситуаций. Платформа поддерживает большое количество сред разработки и тестирования благодаря библиотеке OpenAi Gym. Universe существенно ускорит и улучшит скорость обучения нейронной сети тем самым увеличив её качество. Для решения проблемы предугадывания поведения игрового объекта, будем создавать с помощью нейронной сети дерево поведения. Дерево поведения – это ориентированный ациклический граф, узлами которого являются возможные варианты поведения робота или игрового персонажа. Ширина дерева указывает на возможные действия, который можешь совершить объект. Высота его поддеревьев указывает на сложность алгоритма. Архитектура слияния дерева поведения и нейронной сети в базовых ситуациях, будет выполнять команды сгенерированного дерева поведения, а при появлении нестандартных ситуаций, генерировать решения с помощью нейронной сети.

Заключение

Таким образом, рассмотрев подход разработки архитектуры искусственного интеллекта в игровых персонажах с помощью нейронной сети, генерирующей дерево поведения можно сделать вывод:

- Нейронная сеть позволяет создавать логику более непредсказуемой, но сложной в тестирование.
- С помощью деревьев поведения создается гибкая и легко тестируемая логика.
- При совмещении двух технологий одновременно получаемая архитектура дает возможность для принятия неординарных решений в необычных ситуациях и одновременно логика поддающаяся тестированию в стандартных ситуациях.

Одно из главных преимуществ данного подхода в том, что дерево строиться с помощью нейронной сети, значит, для пользователя стоит лишь найти нужную игру или среду, имитирующую конкретную ситуации для обучения нейронной сети и получить желаемое дерево поведения. После построения дерева существует возможность редактировать это дерево, и добавлять стандартные ситуации в логику.

Список литературы

1. Пасюта А.А, Старолетов С.М. Целесообразность использования модели деревьев поведения по сравнению с автоматной моделью при программировании действий игровых персонажей [Электронный ресурс] / XIV Всероссийская научно-техническая конференция студентов, аспирантов и молодых ученых. – 2017. - Режим доступа: http://book.lib-i.ru/25tehnicheskie/368225-1-ministerstvo-obrazovaniya-nauki-rossiyskoy-federacii-federalnoe-gosudarstvennoe-byudzhetnoe-obrazovateln.php
2. Universe Documentation [Электронный ресурс]. - Режим доступа: https://github.com/openai/universe
3. Роботы, лабиринты и архитектура поглощения [Электронный ресурс]. - Режим доступа: https://www.ibm.com/developerworks/ru/library/j-robots
4. Utility AI: Поведенческие деревья уходят в прошлое [Электронный ресурс]. - Режим доступа: https://www.progamer.ru/dev/utility-ai.htm
5. Delmer S. Behavior Trees for Hierarchical RTS AI [Электронный ресурс] / S. Delmer . - Plano: The Guildhall at SMU, 2013. - 10 с. - Режим доступа:

http://www.smu.edu/~/media/Site/guildhall/Documents/Theses/Delmer_Stephan_Thesis_Final.ashx?la=en
6. Кулинарный путеводитель по архитектурам AI [Электронный ресурс] - Режим доступа -https://habrahabr.ru/post/173187/
7. Unity 5.x. Программирование искусственного интеллекта в играх: пер. с англ. Р.Н. Рагимова. - М.: ДМК Пресс, 2017. - 272 с.

РАЗРАБОТКА СИСТЕМЫ АНАЛИЗА АНОМАЛИЙ В БОЛЬШИХ ДАННЫХ (BIGDATA) НА ОСНОВЕ КОРТИКАЛЬНЫХ АЛГОРИТМОВ

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

Введение

В связи с развитием техники количество информационных потоков вокруг нас стремительно увеличивается. Данные разного характера можно извлечь практически из всего, даже из того, что по умолчанию является неживым. Примером может служить любое электронно-вычислительное устройство или даже объекты, некоторое время назад не имевшие к ЭВМ прямого отношения (например, дома, автомобили – объекты человеческой деятельности, появившиеся на свет лишь на основе знаний механики – в настоящее время содержат в себе весомое количество различных электронных устройств и даже микрокомпьютеров).

С ростом числа потоков данных и способов их получения появляется проблема анализа этих данных. Проблема анализа данных охватывает как минимум две задачи. С одной стороны, на основе имеющихся данных можно попытаться спрогнозировать и предсказать будущее поведение системы (экстраполяция) или же определить поведение системы на имеющемся отрезке времени в те моменты времени, когда данные отсутствовали (интерполяция). С другой стороны, информация имеет обыкновение искажаться, а вычислительные устройства имеют обыкновение допускать ошибки в вычислениях. Поэтому существует вероятность возникновения неожиданных данных в определённые моменты времени, связанная с искажением информации или же неправильной работой системы.

Исходя из этого можно сделать вывод, что анализ данных является довольно важной задачей. И решать её можно разными средствами: с помощью компьютеров, с использованием искусственных нейронных сетей. Может её решать и человек. Ведь анализ данных разного характера часто оказывается нетривиальной задачей, а человеческий мозг под стать данной задаче способен проявлять нетривиальную изобретательность. Но, к сожалению, человек работает зачастую медленнее компьютера и также допускает немало количество ошибок. При такой постановке вопроса хорошим решением была бы искусственная модель человеческого мозга (разума), реализованная машинно.

Такую модель как Иерархическая временная память (*Hierarchical Temporal Memory – **HTM***) в 2004 году предложил Джеф Хокинс в своей книге «Об интеллекте» [1].

В данной работе для анализа данных и поиска в них аномалий мы будем использовать именно модель HTM и платформу для интеллектуальных вычислений от компании Numenta - NuPIC.

Общее описание HTM

Иерархическая временная память является технологией машинного самообучения, которая нацелена на повторение структурных и алгоритмических свойств коры головного мозга (в частности, неокортекса (лат. *neocortex*) – новых областей коры головного мозга, которые у низших млекопитающих только намечены, а у человека составляют основную часть коры).

Как и следует из ее названия, модель *HTM* в основе своей является некоторой системой памяти. Сети *HTM* обучаются много раз на изменяющихся входных данных и механизм их работы основан на запоминании большого множества паттернов и их последовательностей. Модель *HTM* неотъемлемо связана с параметром времени. В ней информация всегда хранится в распределенном виде.

Однако, возможности *HTM* не привязаны к каким-то конкретным видам данных. Её можно использовать в работе не только с человеческими сенсорными потоками, но и, например, с данными от радаров, инфракрасного зрения, данными от показаний различных датчиков в технических устройствах или даже потока данных с финансовых рынков, метеорологических данных или данных о веб трафике [3].

Принципы HTM

Иерархия

Сеть *HTM* состоит из регионов, организованных в иерархию. Регион *HTM* является основным строительным блоком, функциональной единицей её памяти и способности к предсказанию. Типичным случаем является, когда один регион представляет из себя один уровень в иерархии *HTM*. И по мере восхождения вверх по этой иерархии всегда присутствует конвергенция данных (соединение элементов дочернего (нижнего) региона на одном элементе родительского (верхнего) региона). Благодаря наличию обратных связей, при движении вниз по уровням иерархии происходит также и дивергенция информации (то есть, разделение).

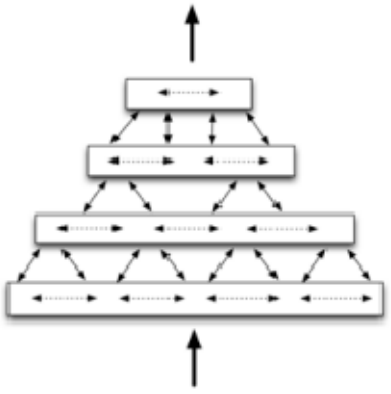


Рисунок 1 – Упрощённая диаграмма четырёх регионов HTM, организованных в четырёхуровневую иерархию

Преимуществом иерархической организации является её высокая эффективность. За счёт многократного использования в комбинациях на более высоких уровнях паттернов, выученных на предыдущих уровнях иерархии, иерархическая организация существенно сокращает время обучения и необходимые объёмы памяти [3].

Регионы

Понятие регионов, из которых складывается иерархия НТМ, происходит из биологии. Неокортекс представляет собой большой слой нервной ткани толщиной около 2 мм. Биологи делят неокортекс на разные области или «регионы», в первую очередь на основе того, как регионы соединяются друг с другом. Некоторые регионы получают входные данные непосредственно от органов чувств, а другие регионы - только после того, как данные будут обработаны несколькими другими регионами. Это связь между регионами определяет иерархию.

Все неокортикальные области схожи в строении. Они различаются по размеру местоположению в иерархии, но в остальном они похожи. Неокортекс состоит из шести слоёв: пять слоёв клеток и один неклеточный слой (есть несколько исключений, но это общее правило). Каждый слой в неокортикальной области имеет множество взаимосвязанных ячеек, расположенных в столбцах.

Области НТМ также состоят из слоя сильно взаимосвязанных клеток, расположенных в столбцах. Клетки региона в модели НТМ примерно эквивалентны нейронам в третьего слоя неокортекса. Хотя регион НТМ является эквивалентом только части области коры мозга, он вполне способен делать распознавания и предсказания в сложных потоках входных данных, что позволяет его использовать при решении многих задач [3].

NuPIC

Платформа Numenta для интеллектуальных вычислений (NuPIC) - это платформа машинного интеллекта, которая реализует алгоритмы обучения НТМ. NuPIC подходит для множества проблем, в частности обнаружения аномалий и прогнозирования потоков данных.

Кодовая база NuPIC Python содержит реализации кода на языке Python для НТМ. Через этот интерфейс пользователи могут указать, использует ли их код алгоритмы Python или более быстрые алгоритмы C++, используя привязки Python, предоставляемые в nupic.core. Помимо предоставления привязок Python к API-интерфейсу nupic.core, эта кодовая база также включает в себя клиентский API более высокого уровня, называемый Online Prediction Framework (OPF), который настроен на работу с предсказаниями, обнаружением аномалий и определением оптимальных параметров модели [4].

Аномалии

Аномалией в широком смысле называется отклонение от нормы, общей закономерности; неправильность.

Под обнаружением аномалий в данных понимается процесс нахождения таких данных, которые не соответствуют ожидаемому поведению. Эти несоответствующие образцы данных принято называть аномалиями, несоответствиями или отклонениями. Таким образом, аномалии (в данных) — это образцы данных, которые не подходят под точно определённое представление нормального поведения [2].

Можно выделить три вида аномалий:

1. Если точка, по отношению к другим данным, является аномальной, то её называют точечной аномалией.
2. Если точка сама по себе не является аномальной, но в контексте других данных ведёт себя аномально, то её называют контекстной аномалией (рисунок 2).
3. Если группа точек является аномалией только по отношению ко всем данным, то её называют коллективной аномалией (рисунок 3) [2].

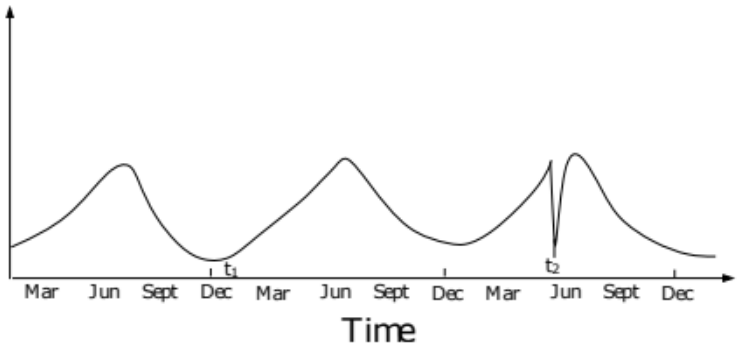


Рисунок 2 – Контекстная аномалия в температурном временном ряду [2]. Температура во время t_1 такая же, как и во время t_2, но находится в ином контексте и поэтому не рассматривается как аномалия

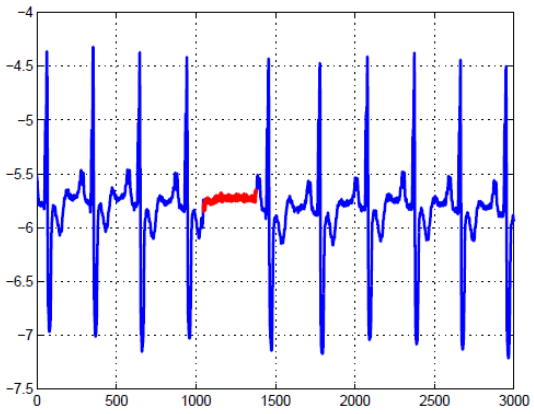


Рисунок 3 – Коллективная аномалия, связанная с преждевременной закупоркой артерии на электрокардиограмме человека

Обнаружение аномалий

Будем считать аномалией количество клеток, которые были в состоянии предсказания, но не перешли в активное состояние. На основе этого получим формулу (1) расчета коэффициента аномалии:

$$anomalyScore = \frac{|A_t| - |(P_{t-1} \cap A_t)|}{|A_t|}, \quad (1)$$

где A_t – Активные клетки в момент времени t, P_t – клетки в состоянии предсказания в момент времени t, $|A|$ – количество элементов в множестве A.

Результаты работы платформы на примере демо-задачи «Потребление энергии»

Входные данные (рисунок 4):

- timestamp – точка во времени;
- kw_energy_consumption – собственно затраты энергии.

	A	B	C	D	E	F	G
1	timestamp	kw_energy_consumption					
2	datetime	float					
3	T						
4	7/2/10 0:00	21.2					
5	7/2/10 1:00	16.4					
6	7/2/10 2:00	4.7					
7	7/2/10 3:00	4.7					
8	7/2/10 4:00	4.6					
9	7/2/10 5:00	23.5					
10	7/2/10 6:00	47.5					
11	7/2/10 7:00	45.4					
12	7/2/10 8:00	46.1					
13	7/2/10 9:00	41.5					
14	7/2/10 10:00	43.4					
15	7/2/10 11:00	43.8					
16	7/2/10 12:00	37.8					
17	7/2/10 13:00	36.6					
18	7/2/10 14:00	35.7					
19	7/2/10 15:00	38.9					
20	7/2/10 16:00	36.2					
21	7/2/10 17:00	36.6					
22	7/2/10 18:00	37.2					
23	7/2/10 19:00	38.2					
24	7/2/10 20:00	14.1					
25	7/2/10 21:00	5.1					
26	7/2/10 22:00	5					

Рисунок 4 – Снимок окна с входными данными потребления

Выходные данные (рисунок 5):

- timestamp – точка во времени;
- kw_energy_consumption – собственно затраты энергии;
- prediction – предсказание (прогноз);
- anomaly_score (оценка аномалии);
- anomaly_likelihood (вероятность аномалии).

	A	B	C	D	E	F
1	timestamp	kw_energy_consumption	prediction	anomaly_score	anomaly_likelihood	
2	2010-07-02 00:00:00	21.2	21.2	1.0	0.5	
3	2010-07-02 01:00:00	16.4	16.4	1.0	0.5	
4	2010-07-02 02:00:00	4.7	4.7	1.0	0.5	
5	2010-07-02 03:00:00	4.7	4.7	1.0	0.5	
6	2010-07-02 04:00:00	4.6	4.6	1.0	0.5	
7	2010-07-02 05:00:00	23.5	23.5	1.0	0.5	
8	2010-07-02 06:00:00	47.5	47.5	1.0	0.5	
9	2010-07-02 07:00:00	45.4	45.4	1.0	0.5	
10	2010-07-02 08:00:00	46.1	46.1	1.0	0.5	
11	2010-07-02 09:00:00	41.5	46.1	1.0	0.5	
12	2010-07-02 10:00:00	43.4	41.5	1.0	0.5	
13	2010-07-02 11:00:00	43.8	43.4	1.0	0.5	
14	2010-07-02 12:00:00	37.8	43.8	1.0	0.5	
15	2010-07-02 13:00:00	36.6	37.8	1.0	0.5	
16	2010-07-02 14:00:00	35.7	36.6	1.0	0.5	
17	2010-07-02 15:00:00	38.9	35.7	1.0	0.5	
18	2010-07-02 16:00:00	36.2	38.9	1.0	0.5	
19	2010-07-02 17:00:00	36.6	36.2	0.975	0.5	
20	2010-07-02 18:00:00	37.2	4.67	1.0	0.5	
21	2010-07-02 19:00:00	38.2	38.2	1.0	0.5	
22	2010-07-02 20:00:00	14.1	14.1	1.0	0.5	
23	2010-07-02 21:00:00	5.1	14.1	1.0	0.5	
24	2010-07-02 22:00:00	5.0	5.1	0.075	0.5	
25	2010-07-02 23:00:00	5.9	5.069999999999999	0.525	0.5	
26	2010-07-03 00:00:00	22.5	4.67	0.975	0.5	

Рисунок 5 – Снимок окна с выходными данными потребления

BigData в автомобильной индустрии

Известно, что современные автомобили имеют одно или несколько управляющих устройств (ЭБУ, ECU - электронный блок управления). Устройства обмениваются между собой через автомобильные шины передачи данных (примеры - K-line, CAN, LIN). Для

большинства устройств, связанных с двигателем и трансмиссией, данные, которые они передают, доступны на шине CAN (Controller Area Network), через диагностический разъем автомобиля и через разъем периферийных устройств, поскольку обычно таких шин две - высокоскоростная (0.5-1 Мбит/с) и низкоскоростная (125-250 Кбит/с).

В настоящий момент реализовано подключение к таким шинам и программное обеспечение сбора данных [4]. Результаты эксперимента сбора данных показали (рисунок 6), что для тестового автомобиля существует около 20 передатчиков, которые шлют разнородные данные, часто специфичные для каждого автомобиля.

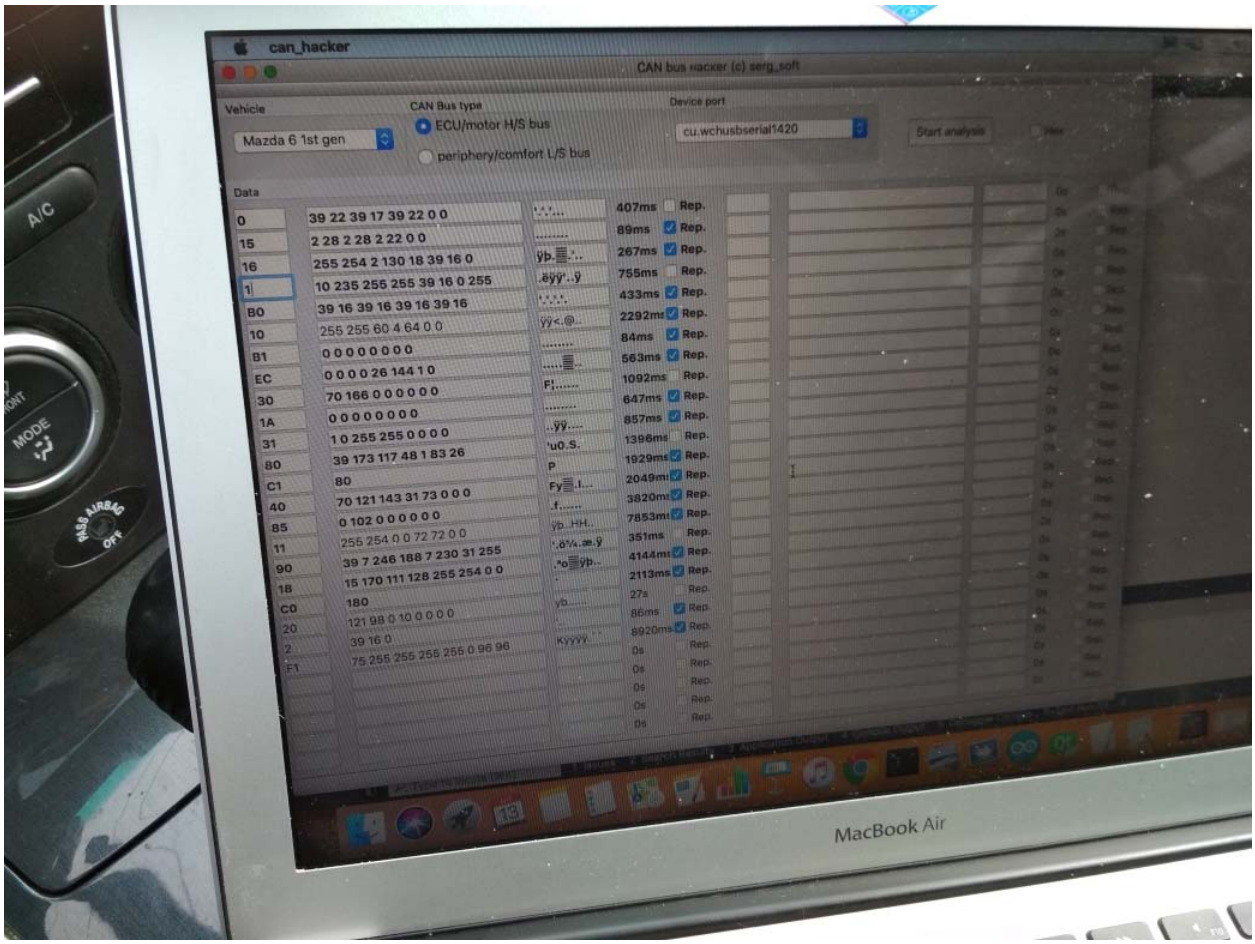


Рисунок 6 – Данные с CAN шины передачи данных автомобиля Mazda 6

Данные характеризуются номером передатчика (CAN ID), за которым следует, обычно, до 8 байт данных. Декодирование данных по типам их ID и содержанию осуществляется энтузиастами [5], однако такая информация в целом закрытая. Сборка и обработка данных с заданной скоростью во время движения представляет собой задачу обработки больших данных (bigdata).

Постановка задачи на исследование

Для задачи потребления электроэнергии, рассмотренной ранее, осуществляется обработка кортикальными алгоритмами только одной серии данных. В то время для задачи поиска аномалий в данных с автомобиля существует несколько серий данных разной длины. Поэтому необходимо поставить задачу - найти аномалию для всего набора данных, представляющего из себя датасет из нескольких серий данных, то есть по сути следует определить в каждый момент времени композитное состояние системы по её данным и осуществить предсказание аномалии для этого состояния. При этом природу самих данных

знать не обязательно (доказано [1], человеческий мозг обрабатывает разные типы данных единообразно).

Задача может решена следующими способами:

- хэширование данных по всем сериям, получение итогового значения и предсказание аномалий среди таких значений;
- параллельная работа по поиску аномалий во всех сериях и определение композитной аномалии;
- создание сети HTM как конвергенции из нескольких сетей HTM по каждой серии данных и поиск аномалий в этой сети.

В настоящее время выполняется исследование по данному вопросу.

Список литературы

1. Джефф Хоккинс. Об интеллекте. - Москва: Издательство Вильямс, 2007. – 240 с. - ISBN 978-5-8459-1139-1.
2. Chandola V., Banerjee A., Kumar V. Anomaly Detection: A Survey //ACM Comput. Surv. - 2009. - Jul. - Vol. 41. - no. 3. - P. 15:1–15:58.
3. HIERARCHICAL TEMPORAL MEMORY including HTM Cortical Learning Algorithms [Электронный ресурс]. - Режим доступа: https://numenta.org/resources/HTM_CorticalLearningAlgorithms.pdf
4. CAN hacker [Электронный ресурс]. - Режим доступа: https://github.com/SergeyStaroletov/can_hacker
5. Mazda CAN ID [Электронный ресурс]. - Режим доступа: http://opengarages.org/index.php/Mazda_CAN_ID

РАЗРАБОТКА АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ ТЕСТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ ПРОМЫШЛЕННЫХ МИКРОКОНТРОЛЛЕРОВ

Хворов К.Е., Цисык В.О. – студенты, Лукоянычев В.Г. – к.т.н., профессор
Алтайский государственный технический университет (г. Барнаул)

Введение

К разработке программного обеспечения в современном мире предъявляют два требования: скорость и качество. Если скорость разработки можно измерить в часах и зависит она от способностей разработчиков, то измерить качество программного обеспечения достаточно сложно. Какое ПО мы можем считать качественным?

В большинстве случаев для контроля качества можно использовать принцип “если работает для *этой* ситуации, то продукт хороший, иначе - плохой”. Иначе говоря, проверив работоспособность ПО на определенном количестве тестов, можно судить о его качестве.

При большом количестве тестов возникает желание автоматизировать процесс тестирования. К счастью разработчиков, существует большое количество тестирующих инструментов, которые позволяют это сделать, избавляя программистов от повторения монотонных действий при каждом релизе.

Однако, если мы разрабатываем ПО для встраиваемых микросистем, то наши возможности окажутся куда более скромными. Микроконтроллеры обладают рядом ограничений, которые часто просто не позволяют использовать обычные системы тестирования. С ростом популярности микроконтроллеров проблема тестирования ПО становится актуальной.

Анализ проблемы

Для проверки качества программного обеспечения традиционно используют наборы тестов, которые запускаются при обновлении версии программы. Следует выделить особенности в тестировании ПО для микроконтроллеров:

1. Ограниченное количество памяти;
2. ограниченные аппаратные возможности;
3. физическое воздействие важно;
4. большое количество подключаемых модулей.

Подобные ограничения не позволяют разработчикам запускать полноценные тестирующие фреймворки: микроконтроллеры слишком слабы для них. Существует ряд подходов к тестированию, которые позволяют обойти указанные особенности:

1. Использование эмуляторов;
2. тестирование при помощи сервиса непрерывной интеграции;
3. тестирование при помощи хостовой машины;
4. тестирование программного кода на более мощной аппаратной платформе;
5. использование “легковесных” тестирующих фреймворков.

Данные подходы имеют свои области применения, преимущества и недостатки. Какой подход применить зависит от возможностей железа и желания разработчиков настраивать тестирующую систему: каких-либо готовых, “из коробки”, продуктов не существует.

Предлагаемый вариант решения

Мы предлагаем использовать тестирующий стенд двух устройств: тестируемое и тестирующее. В качестве тестируемого выступает некоторый микроконтроллер, а в качестве тестирующего может выступать другой микроконтроллер либо машина.

На тестируемое устройство заливается прошивка с тестами. В зависимости от того, как тест был пройден, в памяти устройства устанавливаются соответствующие флаги. Тестирующее устройство взаимодействует с тестируемым и возвращает некоторый результат. Сам стенд собирает информацию о тестировании и отправляет ее в систему непрерывной интеграции.

Таких стендов может быть довольно много, и каждый может иметь отдельную конфигурацию и свой репозиторий исходного кода. Примерная схема решения представлена ниже.

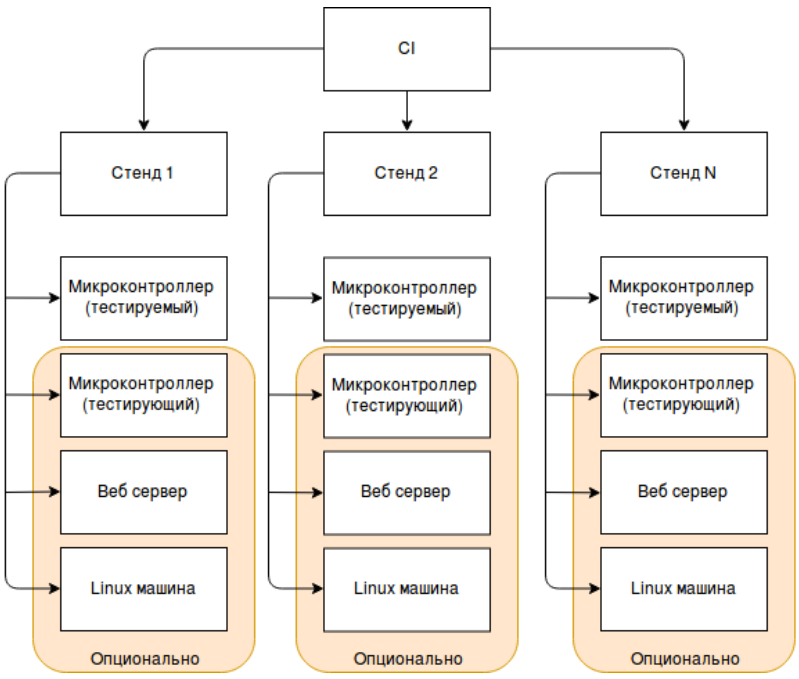


Рисунок 1 - Схема решения

Итерацию тестирования можно представить следующими пунктами:

1. Исходный код загружается в систему контроля версий
2. Система непрерывной интеграции автоматически запускает сборку прошивок для тестируемого и тестирующего устройства на определенном стенде.
3. *Написанное ПО* после компиляции загружает прошивки в соответствующие устройства
4. Выполняются тесты на устройствах
5. *Написанное ПО* собирает результаты тестирования с устройств
6. Система непрерывной интеграции анализирует результаты тестирования и формирует отчет о результатах тестирования.

В рамках статьи были описаны основные ограничения, возникающие при разработке программного обеспечения для микроконтроллеров, и приведены основные методы тестирования программного обеспечения. Кроме того, был предложен собственный вариант системы тестирования.

Список литературы

1. CI for Embedded Systems [Электронный ресурс]. - Режим доступа: https://jamesmunns.com/blog/hardware-ci-overview/
2. CI for Embedded - Common Techniques [Электронный ресурс]. - Режим доступа: https://docs.google.com/spreadsheets/d/1ScSDfn9v73TBaGpuiGfpPTqnBeTvNd22TzApMt_E0qE/edit#gid=0
3. Grenning J.W. Test-Driven Development for Embedded C: The Pragmatic Bookshelf. - 2010.
4. TDD and Embedded Programming [Электронный ресурс]. - Режим доступа: https://www.element14.com/community/message/103355/l/tdd-and-embedded-programming#103355

ПРОЕКТИРОВАНИЕ АДАПТИВНОЙ СИСТЕМЫ УПРАВЛЕНИЯ ПРОМЫШЛЕННЫМИ РОБОТАМИ

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

Введение

В современном мире роботы используются повсеместно и в широком спектре задач. Они могут выглядеть как манипулятор на сборочной линии, автомобиль или веб-страница, которая просит вас распознать объекты на фотографиях [1,2,7]. Роботы отлично справляются с монотонными задачами, не устают, а качество их работы невероятно высоко. Подобные «сотрудники» подходят для монотонных и высокоточных работ куда лучше людей. Очевидно, что подобные машины нашли применение в первую очередь в промышленности.

Автоматизация производственного процесса не позволяет до конца исключить человека. Как минимум кто-то должен следить за работой и предпринять действия в случае сбоя. Если это будет делать машина, то по какому алгоритму она бы работала? Как производителю учесть все возможные ситуации? Программное обеспечение, которое управляет роботом куда важнее, чем ее физическое воплощение. Создавать робота, который строго следует одному алгоритму и не может от него отступить — это не выгодно и не позволяет задействовать все его потенциальные возможности. Производители современных роботов стремятся создать продукт, способный так или иначе адаптироваться к изменяющимся условиям работы. Как следствие, робот будет более универсальным, сможет выполнять свою работу эффективнее и качественнее.

Адаптивные системы управления

Качество ПО для роботизированной системы тем выше, чем больше априорной информации о самом объекте и условиях его работы известно [1]. При создании системы трудно описать ее характеристики изначально. Например, динамические характеристики летательных аппаратов существенно зависят от температуры, режима полета, ветра и других параметров. Предсказать подобные условия невозможно, однако автопилот должен на них адекватно среагировать.

В связи с этим возникает проблема построения адаптивной системы, которая не требует полной информации об объекте и условиях его функционирования. Эффект адаптации достигается за счет накопления и обработки информации о поведении объекта в процессе его работы. Это позволяет снизить влияние неопределенности на качество управления.

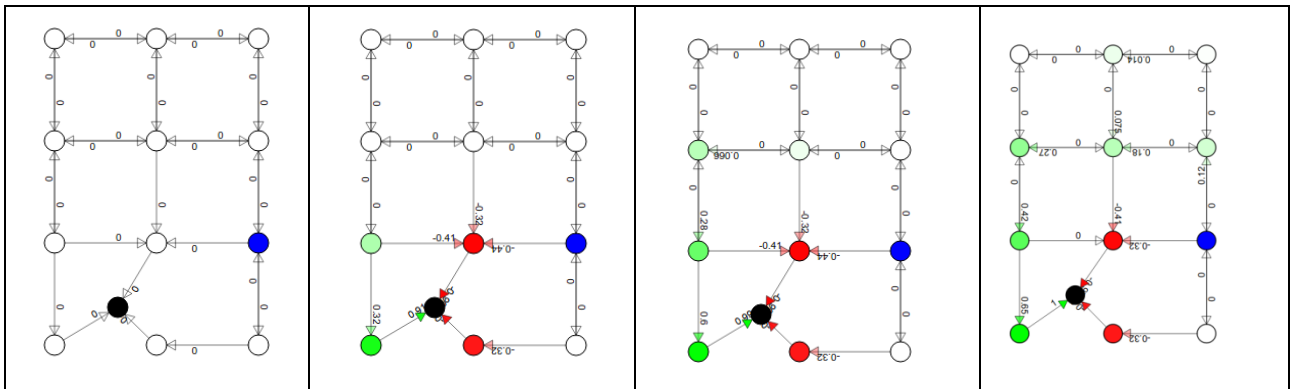
Обучение с подкреплением

Обучение с подкреплением - один из методов обучения системы (*агента*). Метод предназначен длят обучения модели, которая не имеет сведений об окружающей его среде, но имеет возможность производить какие-либо действия в ней. Агент воздействует на среду, а среда воздействует на агента: взаимодействуя со средой, агент получает вознаграждение за свои действия. *Агент* должен *действовать* в *окружении* так, чтобы максимизировать некоторый *выигрыш*. Агент повторяет процесс обучения до тех, пока выигрыш не перестанет изменяться в пределах заданной точности.

Q-Learning

В данной работе реализован простой, но являющийся базовым для понимания обучения с подкреплением, табличный алгоритм Q-Learning [3-5]. В распоряжении агента имеются некоторое множество действий $A(a_1, a_2 \dots a_n)$. Агент имеет возможность определить, в каком состоянии он находится в данный момент. Действия агента влияют на среду, и за них от среды он получает то или иное вознаграждение **R(a,s)**. Среда представляет из себя множество возможных состояний $S(s_1, s_2 \dots s_n)$ и *матрицу переходов* R(s, a, s') между ними. Матрица содержит вероятности достижения состояния **s'**, если в состоянии **s** было выполнено действие **a**.

Задачей агента является найти наилучшую стратегию. В данном случае она будет описываться Q-значениями (Q-Values), которые определяют полезность **Q(a,s)** выполняемого действия **a** в соответствующем состоянии **s**. Представить обобщенно окружающую среду можно в виде графа, у которого состояния представлены вершинами, а переходы — ребрами. На рисунке 1 изображен фрагмент эпизода обучения.



Начальное состояние системы	Состояние системы после нескольких итераций обучения	Состояние системы после десяти итераций обучения	Обученная система

Рисунок 1 - Последовательность этапов обучения системы

Алгоритм Q-Learning

1. Начать с $Q_0(s,a)$ для всех a и s';
2. для $k = 1,2\ldots$ пока значения не сойдутся:
3. выбрать действие a, получить действие s';
4. если s' - конечное, то:
   1. $target = R(s,a,s')$;
   2. выбрать новое состояние s';
5. иначе:
   1. $target = R(s,a,s') + \gamma \max_{a'} Q_k(s',a')$;
   2. $Q_{k+1}(s,a) \leftarrow (1-\alpha)Q_k(s,a) + \alpha[target]$;
   3. $s \leftarrow s'$.

То, как агент выбирает действия, важно. Если он выбирает действия случайно, то будут рассмотрены все варианты действий, и агент получит полную информацию о системе. Однако, часто действия будут ошибочными и не будут приносить положительной награды. Если действовать жадно и выбирать только те действия, где награда максимально, то не будут рассмотрены все переходы. Одним из вариантов решения служит эпсилон-жадный алгоритм: с вероятностью ε выбирается случайное действие, а с вероятностью 1 - ε действие выбирается по жадному алгоритму [3].

Вычисляемые значения для каждого состояния после определенного числа итераций перестанут измениться в пределах выбранной точности вне зависимости от того, как агент выбирает действия.

Таким образом, создан каркас для дальнейшей модульной разработки адаптивной системы управления роботами, а так же представлен один из алгоритмов обучения. Визуализация позволяет в общем случае увидеть процесс обучения.

Систему можно существенно расширить и дополнить, добавив в нее различные методы обучения.

Список литературы

1. Теория автоматического управления: Учеб. для вузов по спец. «Автоматика и телемеханика». В 2-х ч. Ч. II. Теория нелинейных и специальных систем автоматического управления. / А. А. Воронов, Д. П. Ким, В. М. Лохин и др.; Под ред. А. А. Воронова. - 2-е изд., перераб. и доп. - М.: Высш. шк., 1986. - 504 с.
2. Deep Reinforcement Learning: Pong from Pixels [Электронный ресурс]. - Режим доступа: http://karpathy.github.io/2016/05/31/rl/
3. Deep RL Bootcamp [Электронный ресурс]. - Режим доступа: https://sites.google.com/view/deep-rl-bootcamp/lectures / Deep RL Bootcamp
4. UC Berkeley CS 188 [Электронный ресурс]. - Режим доступа: http://ai.berkeley.edu/home.html / UC Berkeley CS188 Intro to AI -- Course Materials
5. Викиверситет [Электронный ресурс]. - Режим доступа: https://ru.wikiversity.org/wiki/Q-learning/ / Q-learning

РАЗРАБОТКА ТРЕБОВАНИЙ К АРМ ТЕРАПЕВТА

Сухоруков В.Е. – студент, Астахова А.В. – к.э.н., доцент
Алтайский государственный технический университет (г. Барнаул)

С развитием IT-направления совершенствуются системы организационного управления предприятий и организаций, в том числе, организаций здравоохранения. Автоматизация работ по регистрации, учету, анализу, отчетности в данной области, несомненно, способствуют улучшению качества медицинских услуг и облегчению работы врача. Кроме того, современные информационно-коммуникационные технологии позволяют сократить затраты времени пациентов поликлиник и обеспечить им комфортные условия взаимодействия с медицинским учреждением.

Одним из примеров современной автоматизации может служить внедрение в медицинских учреждениях нашего края программного продукта с достаточно странным названием "АРМ поликлиника". Заметим, что АРМ – автоматизированное рабочее место – может быть у специалиста, а поликлиника не является специалистом.

Данный продукт предназначен для облегчения выполнения должностных обязанностей врача - терапевта.

Однако, практика использования данного АРМ показывает, что врачи поликлиник при использовании названного программного продукта сталкиваются с рядом неудобств. Основными из них являются:

- отсутствие разделения выдаваемой информации в зависимости от пользователя (например, для врача и статиста выдается одна и та же информация);
- наличие избыточной информации, которая на практике редко востребована;
- наличие достаточного количества функциональных возможностей, которые не реализованы в настоящее время, но отображены на экране монитора;
- несколько программных блоков реализованы локально (не связаны с основной программой «АРМ поликлиника»), хотя алгоритмически информационные связи с АРМ очевидны (например, «Электронный больничный», «Диспансеризация» и другие).

В соответствии с вышесказанным можно сделать вывод, что шаги в направлении автоматизации работы медицинских сотрудников в нашем крае сделан, но проект требует значительных доработок.

Разрабатываемый автором данных тезисов IT-проект по заявке одной из поликлиник Барнаула планируется для опытной эксплуатации, по результатам которой администрация поликлиники планирует разработать пожелания и требования для централизованной доработки имеющегося программного обеспечения «АРМ поликлиника».

Разрабатываемый автором данных тезисов проект "АРМ терапевта" представляет собой программу, охватывающую множество врачебных обязанностей:

- запись на первичный или повторный прием к терапевту или узкому специалисту, а также возможность вызова врача на дом;
- ведение амбулаторных карт всех пациентов (жалобы, анализы, направления, медикаменты, рекомендации, сведения о прививках, сведения о противопоказаниях и прочее);
- ведение личного блока амбулаторной карты (паспортные данные, место жительства, номер СНИЛСа, номер медицинского полиса, место работы и/или учёбы (если есть));
- выписка направлений на анализы;
- работа с больничным листом (открытие, продление, закрытие);
- назначение лечения пациенту и прочих рекомендаций;
- прикрепление результатов сторонних обследований;
- регистрация информации о госпитализации (направление и выписка);

- ведение диспансеризации взрослого населения;
- формирование запросов и отчётов.

Названные возможности существенно дополняют возможности эксплуатируемого в настоящее время программного обеспечения.

Анализ функциональных требований и соответствующих деловых документов, используемых для реализации функции, позволил выявить требования к информационной модели предметной области, выявить такие сущности, как:

- персональные данные,
- протокол посещения,
- больничный лист,
- диспансеризация,
- прививочный сертификат,
- госпитализация, талоны на день,
- расписание на день,
- и др.

Для реализации базы данных проектируемого АРМ предусмотрены следующие основные справочники:

- методика обследования,
- группа инвалидности,
- группа крови,
- организация,
- компания-страховщик,
- код нарушения режима,
- лекарства,
- и др.

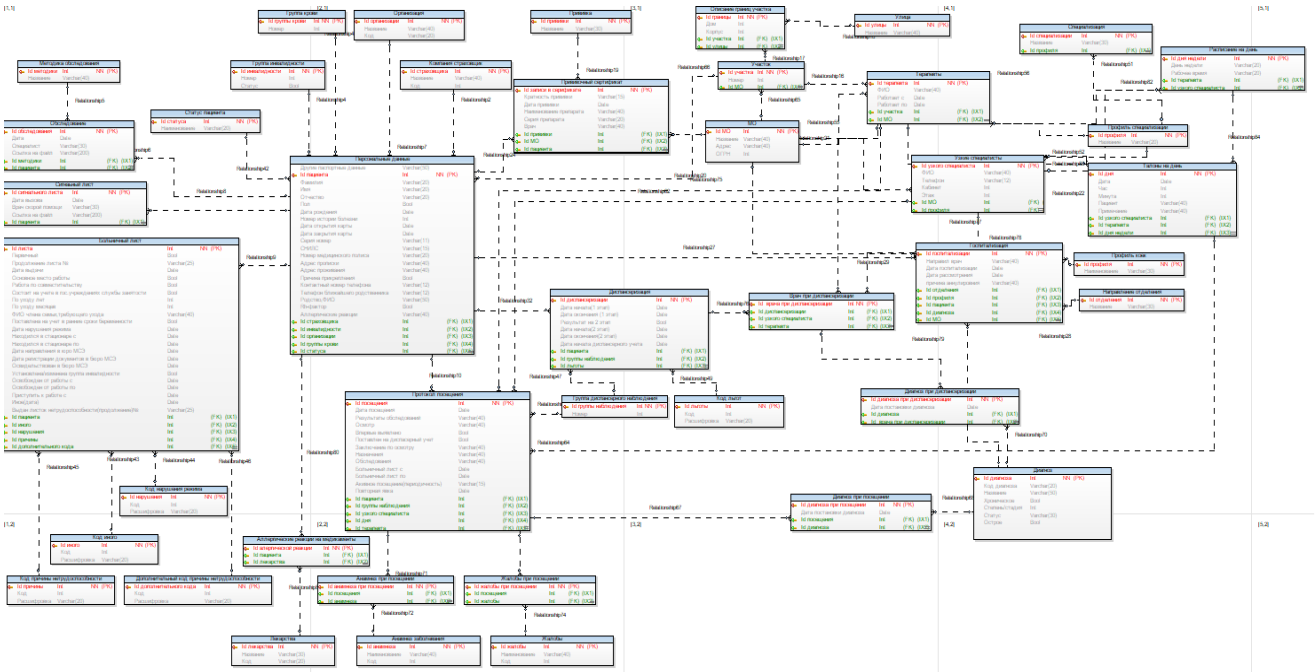


Рисунок 1 – Фрагмент полной атрибутивной модели данных

При работе врача в рамках АРМ требуется добавлять, редактировать, удалять информацию о пациентах и врачах, связывать сотрудника медицинского учреждения и услугу, которую он предоставляет, и выполнять ряд других типовых действий, связанных с

хранением и поиском оперативный и нормативно-справочной информации. Для хранения и поиска данных целесообразно использовать базу данных (см. рис.1).

Проектом предусматривается, что клиентское приложение будет обращаться напрямую к серверу базы данных с установленной СУБД.

Реализации приложения планируется в рамках многоуровневой клиент-серверной архитектуры (см. рис. 2)

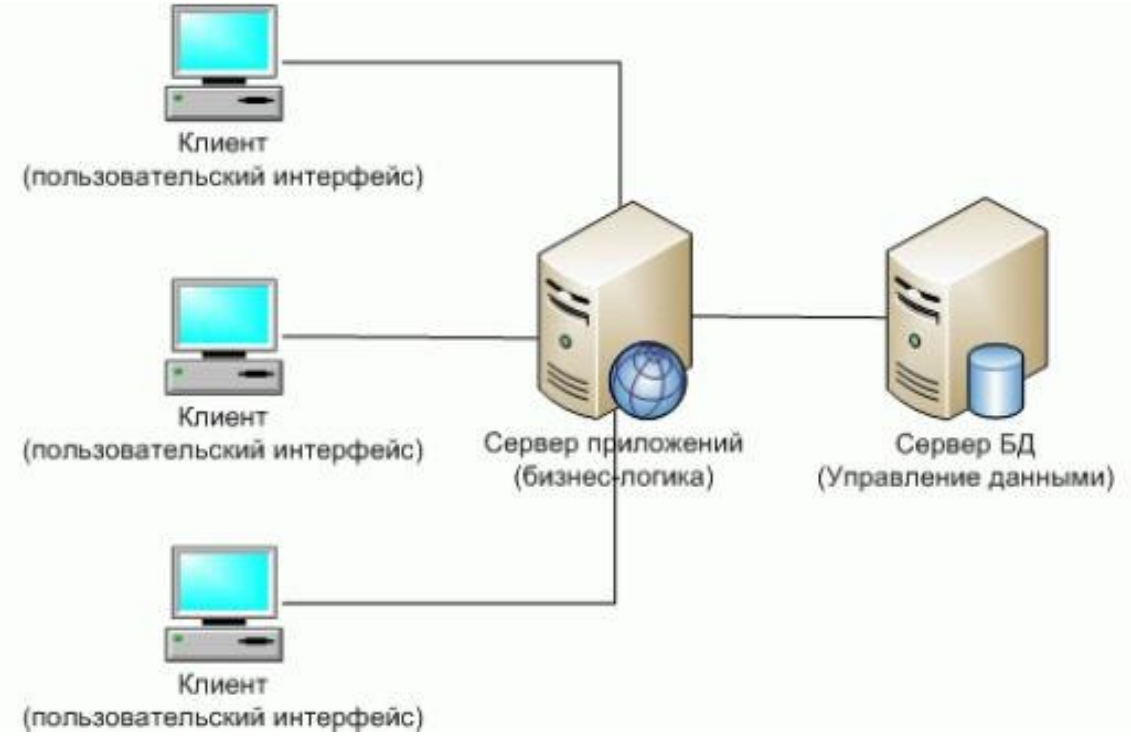


Рисунок 2 – Архитектура «Многоуровневый клиент-сервер»

Данная архитектура для реализации описанных выше требований к АРМ достаточно эффективна, поскольку в ней можно скрыть от пользователей все детали реализации серверного ПО, разрешив врачу обращаться только к тем пунктам меню, которые непосредственно связаны с автоматизируемыми должностными обязанностями. Кроме того, такую систему достаточно просто адаптировать к Web-технологиям: удобнее разработать html-формы для доступа пользователей лишь к определенным атрибутам базы данных, чем ко всем данным.

Список литературы

1. Пирогов В.Ю. Информационные системы и базы данных. Организация и проектирование. – СПб.: БХВ-Петербург, 2009. – 528 с.
2. Орлик С. Многоуровневые модели в архитектуре клиент-сервер [Электронный ресурс]. – Режим доступа: http://citforum.ru/database/kbd97/22.shtml
3. Арунянц Г.Г., Столбовский Д.Н., Калинкин А.Ю. Информационные технологии в медицине и здравоохранении. – Феникс, 2009. – 384 с.
6. Machinelearning.ru [Электронный ресурс]. – Режим доступа: www.machinelearning.ru/wiki/index.php?title=Обучение_с_подкреплением / Обучение с подкреплением
7. Youtube.com [Электронный ресурс]. – Режим доступа: https://www.youtube.com/watch?v=R9OHn5ZF4Uo / How Machines Learn

ВОПРОСЫ ИСПОЛЬЗОВАНИЯ ТЕХНОЛОГИИ IDEF0 ПРИ РАЗРАБОТКЕ ТРЕБОВАНИЙ К АВТОМАТИЗАЦИИ УЧЕБНОГО ПРОЦЕССА

Макашова Е.А. – студент, Астахова А.В. – к.э.н., доцент
Алтайский государственный технический университет (г. Барнаул)

Повышение эффективности управления учебной деятельностью ВУЗа во многом определяется качеством используемых в управлении компьютерных технологий. Проектирование и реализация проекта информационной системы (подсистемы) учебного процесса ВУЗа или другой учебной организации требует предварительной разработки модели управленческих процессов. Именно моделирование процессов является основой для выработки требований к информационному и к прикладному программному обеспечению информационной системы.

При разработке модели следует учитывать, что она должна отображать основные механизмы и принципы взаимосвязи различных подпроцессов в рамках исследуемого бизнес-процесса.

Для решения подобных задач моделирования сложных систем была разработана методология IDEF0. С ее помощью можно эффективно отображать и анализировать модели деятельности широкого спектра сложных систем в процессе разбиения на подсистемы. Также разработчик сам выбирает глубину описания процессов, что позволяет не перегружать модель лишней информацией.

В основе методологий семейства IDEF0 лежат четыре основных понятия.

Первым понятием является функциональный блок. Он изображается в виде прямоугольника и представляет собой конкретную функцию в рамках системы. Название функционального блока формулируется отглагольного прилагательного.

Каждая из четырех сторон функционального блока имеет своё определенное значение, при этом:

1) верхняя сторона имеет значение «Управление»;
2) левая сторона имеет значение «Вход»;
3) правая сторона имеет значение «Выход»;
4) нижняя сторона имеет значение «Механизм».

Вторым понятием является понятие интерфейсной дуги или стрелки. Они отображают элемент системы, который обрабатывается функциональным блоком или оказывает влияние на функцию. Изображается интерфейсная дуга в виде однонаправленной стрелки, которая имеет свое название.

Каждый процесс происходит по определенным правилам и выдает соответствующий результат. Поэтому функциональный блок должен иметь хотя бы одну управляющую и одну исходящую стрелку.

Третьим понятием является декомпозиция. Принцип декомпозиции применяется при разбиении сложного процесса на составляющие его функции. Уровень детализации процесса определяется непосредственно разработчиком модели.

Декомпозиция позволяет постепенно и структурировано представлять модель системы в виде иерархической структуры отдельных диаграмм, что делает ее менее перегруженной и легко усваиваемой.

В процессе декомпозиции, функциональный блок, который в контекстной диаграмме отображает систему как единое целое, подвергается детализации на другой диаграмме. Получившаяся диаграмма второго уровня содержит функциональные блоки, отображающие главные подфункции функционального блока контекстной диаграммы и называется дочерней по отношению к нему. В свою очередь, функциональный блок – предок называется

родительским блоком по отношению к дочерней диаграмме, а диаграмма, к которой он принадлежит – родительской диаграммой.

Последним из понятий IDEF0 является глоссарий, который описывает сущность данного элемента.

Методология моделирования бизнес-процессов IDEF0 реализуется, в частности, в свободно распространяемом программном продукте Ramus. С помощью данной программы можно описать бизнес-процессы, а также создать систему классификации и кодирования, всех объектов, которые используются в бизнес-процессе организации (предприятия) и увязывать эту систему с графическими моделями бизнес-процессов. Также Ramus обладает гибкими возможностями построения отчётности бизнес-процессов и системе классификации и кодирования, позволяет просматривать содержимое проектов через веб-интерфейс.

Рисунок 1 – Контекстная диаграмма моделируемого процесса

Пример такой модели разработан автором данных тезисов в рамках учебного проекта по разработке и моделированию реализации образовательно-пространственной среды в организациях, осуществляющих профессионально-педагогическую деятельность (на примере инклюзивной школы).

Разработанная функциональная модель позволяет выделить требования, как к информационному обеспечению исследуемой предметной области, так и к автоматизируемым функциям соответствующей системы управления учебным процессом.

Основные результаты построения модели с использованием языка IDEF0 приведены на рисунках 1 и 2. На рисунке 1 представлена контекстная диаграмма для задачи управления учебным процессом на примере управления учебным процессом для обучающихся с ограниченными возможностями зрения (ОВЗ) с учетом основой цели образования.

Как следует из диаграммы рис.1, в качестве входной информации, на основе которой выполняется управление учебным процессом для обучающихся с ОВЗ, используются:

1) данные о наборе обучающихся с ОВЗ;
2) информация о фактическом предметном пространстве в учебном заведении для обучающихся с ОВЗ.

Управляющие параметры модели определяются нормативными правовыми и нормативно-организационными документами, регламентирующими образовательную деятельность в рассматриваемом учебном заведении. Это:

1) ФГОС НОО ОВЗ;
2) устав инклюзивной школы.

В роли «Механизмов управления» выступают:

1) преподавательский состав;
2) информационная система;
3) администрация;
4) сотрудники.

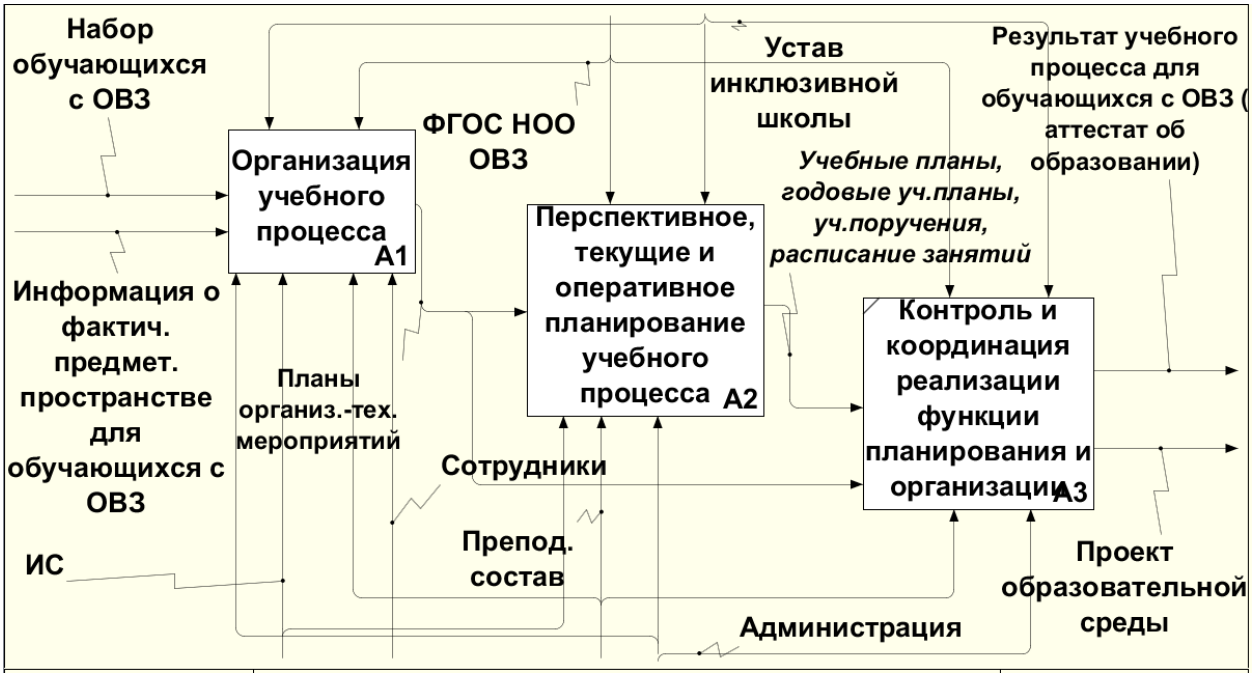


Рисунок 2 – Диаграмма декомпозиции первого уровня

Результатом в системе управления данным учебным заведением (и соответственно – результатом моделирования процессов) должны быть:

1) аттестат об образовании обучающегося, востребованный на рынке современных образовательных услуг;
2) проект образовательной среды, в рамках которой должно осуществляться образование.

Диаграмма декомпозиции первого уровня для рассматриваемого процесса приведена на рисунке 2.

Очередность выполнения функций для решения рассматриваемой задачи следующая:

1) организация учебного процесса (блоки А1). В качестве исходной информации выступают набор обучающихся с ОВЗ и информация о фактическом предметном пространстве. Выходной информацией служат планы организационно-технических мероприятий;
2) перспективное, текущие и оперативное планирование учебного процесса (блок А2). На базе организационно-технических мероприятий создаются учебные планы и расписание занятий;
3) контроль и координация реализации функций планирования и организации (блок А3).

Декомпозиция второго и третьего уровней позволяет детально представить процессы, реализующие укрупненные функции, отображенные на диаграмме рисунка 2, результатом которых являются требования к информационным потокам системы и автоматизируемым управленческим функциям и работам.

Следует отметить, что выполнение подобных проектов еще на этапе вузовской подготовки специалистов способствуют внедрению компетентностного подхода к образованию, позволяя обучающимся глубже понять не только теоретические основы автоматизации исследуемых предметных областей, но и почувствовать связь теории с практикой.

Список литературы

1. Назначение и состав методологии IDEF0 [Электронный ресурс]. - Режим доступа: https://sites.google.com/site/anisimovkhv/learning/pris/lecture/tema6/tema6_2
2. Моделирование бизнес-процессов [Электронный ресурс]. - Режим доступа: http://www.economic-s.ru/index.php/business-software/ramus-modelirovanie-biznes-protsessov/

ИСПЫТАНИЯ СИСТЕМЫ УПРАВЛЕНИЯ ДВИЖЕНИЕМ КОЛЕСНОГО ТРАКТОРА

Ненайденко А.С. - аспирант, Байбасаров Р.Р. - студент
Алтайский государственный технический университет (г. Барнаул)

Важнейшей задачей сельскохозяйственного производства является обеспечение дальнейшего роста производительности труда при сохранении высокого качества сельскохозяйственных работ. Достигается это путем повышения эффективности использования колесных машин на основе улучшения устойчивости их движения и управляемости. Одним из наиболее эффективных средств повышения устойчивости и управляемости является внедрение систем точного земледелия, позволяющих осуществлять управление движением без участия водителя (или с его минимальным участием). Стоит отменить, что в настоящий момент на нашем рынке можно найти такие устройства только импортного производства.

В АлтГТУ им. И.И. Ползунова проводятся исследования целью, которых является создание системы управления движением колесной сельскохозяйственной машины для отечественной техники, не уступающей по своим характеристикам импортным системам и имеющую более приемлемую стоимость для сельхозпроизводителей [1].

Программно-аппаратная схема взаимодействия элементов системы управления

Важной задачей при разработке системы управления является проработка программно-аппаратной реализации и алгоритма управления, обеспечивающего рабочее движение по задаваемой траектории. Схематично программно-аппаратные части разрабатываемой системы управления представлены на рисунке 1.

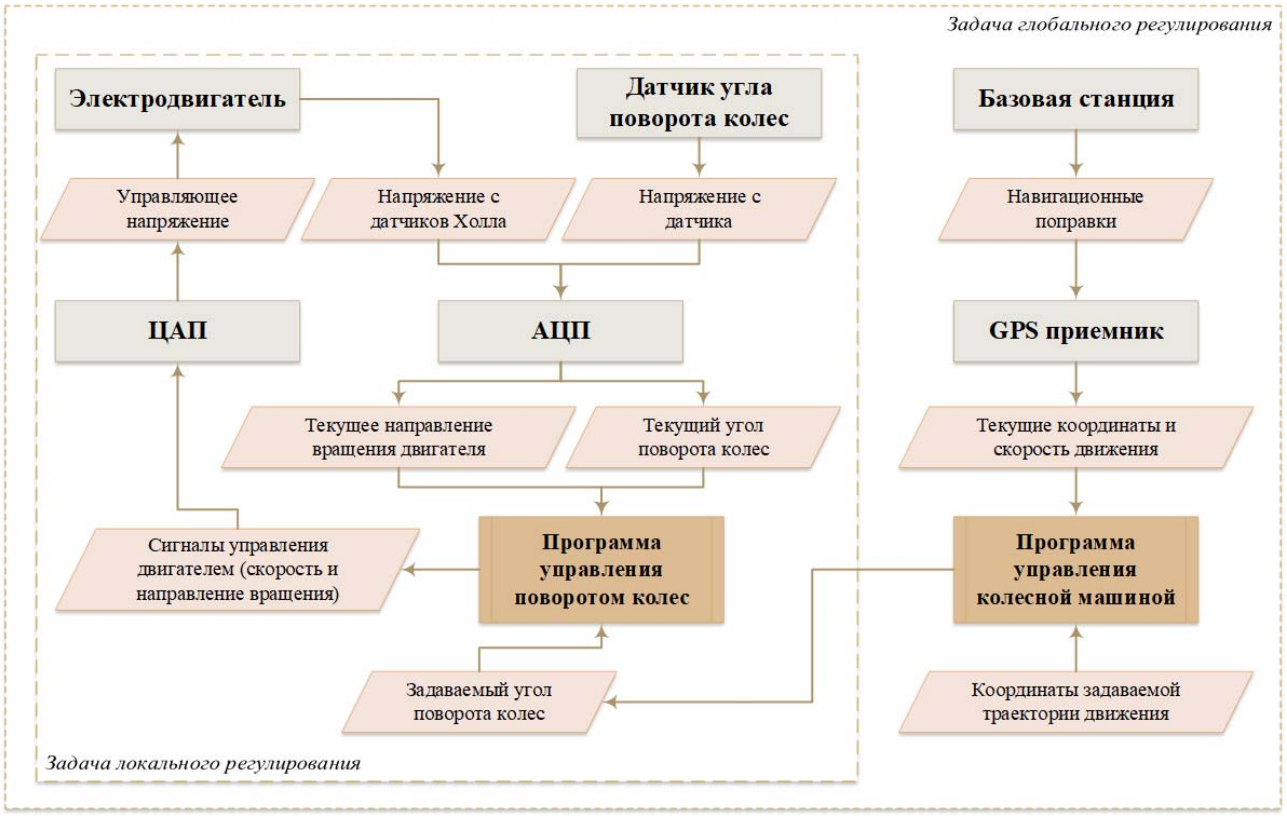


Рисунок 1 – Схема взаимодействия программно-аппаратных частей системы управления

Алгоритм управления условно можно разбить на две составные части – глобальное и локальное регулирование.

Локальное регулирование обеспечивает реализацию текущего задаваемого значения угла поворота управляемых колес. На вход программы управления поворотом колес посредством АЦП поступают значения задаваемого угла поворота, текущее значение угла поворота колес с датчика угла поворота и направление вращения электродвигателя. На основании этих параметров с помощью алгоритма ПИД-регулирования происходит расчет величины управляющего напряжения, которое необходимо передать с помощью ЦАП-а на электродвигатель для реализации необходимого угла поворота.

Глобальное регулирование должно по отклонению от задаваемой траектории определять угол поворота управляемых колес (или угла слома рамы), обеспечивающий движение по задаваемой траектории. На вход программы управления движением сельскохозяйственной машины поступают координаты задаваемой траектории движения, текущие координаты и проекции скорости на неподвижные координатные оси. Происходит вычисление необходимого угла поворота колес, который в свою очередь передается в программу локального регулирования. Текущие координаты и проекции скоростей поступают с навигационного приемника, который работает в связке с базовой корректирующей станцией для обеспечения приемлемой точности определения текущего местоположения.

Апробация системы управления

Ранее на языках высокого уровня C++ и C# было разработано программное обеспечение, обеспечивающее слаженное взаимодействие аппаратных составляющих системы, а также реализующее алгоритмы программ локального [2] и глобального регулирования. Полученная система управления была протестирована в лабораторных условиях на экспериментальном стенде «рулевое управление – передняя подвеска легкового автомобиля». При этом реальная колесная машина была заменена ее математической моделью [3]. После успешных

лабораторных испытаний было принято решение о готовности системы управления для апробации на реальной колесной машине.

Эксперимент проходил на тракторе МТЗ-80.1, принадлежавший управлению административно-хозяйственной работы АлтГТУ. На рисунках 2–4 представлены комплект оборудования и аппаратура для проведения испытаний.

Рисунок 2 – Электродвигатель с системой крепления на рулевое колесо

Рисунок 3 – Крепление датчика угла поворота и антенны навигационного приемника

Рисунок 4 – Блок управляющей аппаратуры, содержащий ЦАП/АЦП, блок управления двигателем и преобразователь напряжения

Во время проведения эксперимента сначала записывались координаты траектории движения трактора, затем данная траектория должна быть реализована с использованием разработанной системы управления, без участия водителя. На рисунке 5 представлены результаты движения по прямой задаваемой линии.

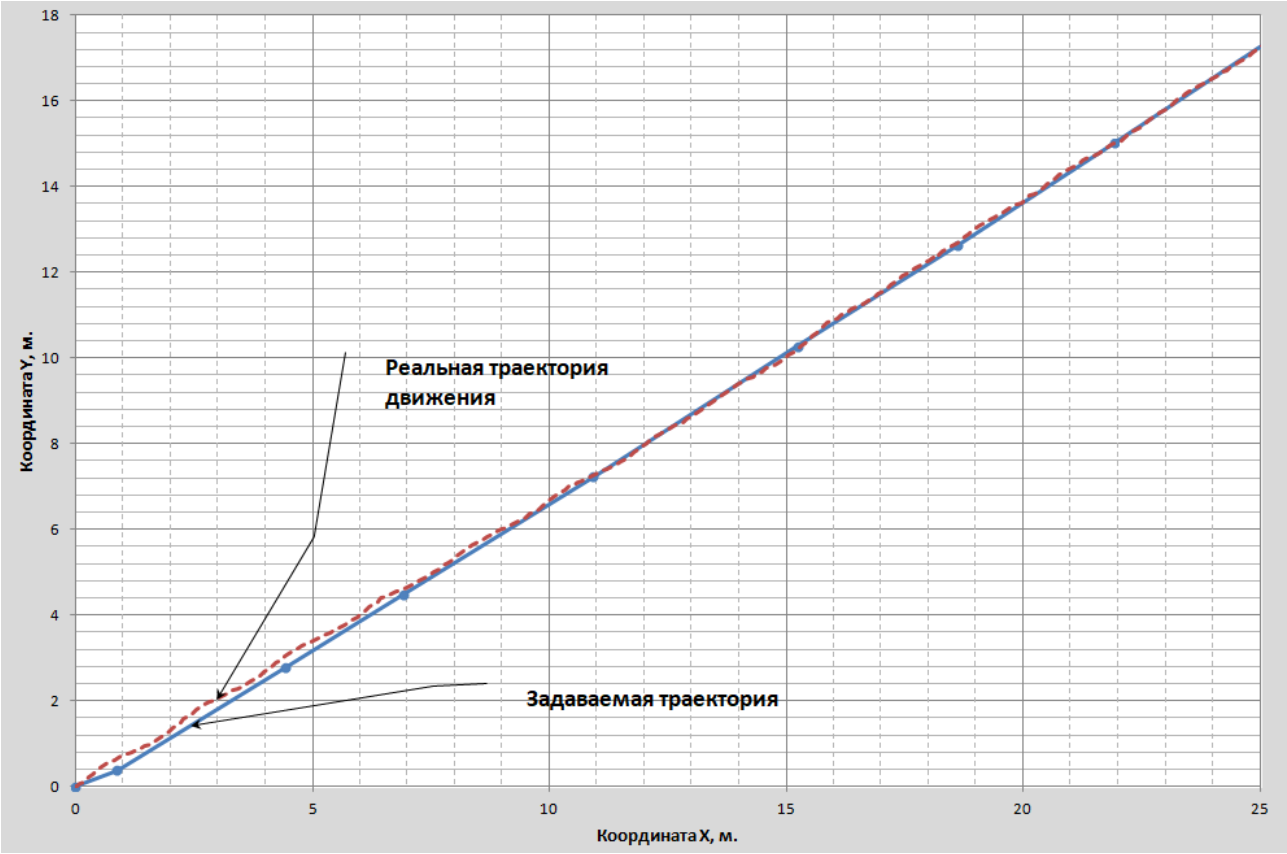


Рисунок 5 – Результаты движения по прямолинейной траектории

Основные выводы

На основании анализа результатов проведенного эксперимента можно сделать вывод об успешной работоспособности создаваемой системы управления. При движении по задаваемой прямолинейной траектории при средней скорости 0,6 м/с отклонения от задаваемой траектории не превысили 0,2 м. С увеличением средней скорости движения колесного трактора до 1,2 м/с отклонения увеличились до 0,4 м.

Из-за тугого рулевого управления и значительных люфтов, система управления не смогла обеспечить качественное движение по задаваемой криволинейной траектории. В ходе эксперимента было установлено, что при значительном сопротивлении повороту рулевого колеса крепление электродвигателя к рулевой колонке функционирует недостаточно надежно. В дальнейшем планируется усовершенствовать конструкцию крепления, а также провести испытания рабочего движения трактора в полевых условиях.

Список литературы

1. Поддубный В.И., Павлюк А.С., Шапошников Ю.А., Ковалев И.М. Управление движением колесного трактора с использованием спутниковых радионавигационных систем // Тракторы и сельхозмашины. 2016. - № 2. - С. 46–49.
2. Поддубный В.И., Ненайденко А.С., Валекжанин А.И. Разработка ПИД-регулирования для реализации задаваемого закона изменения угла поворота колес сельскохозяйственной машины // Ползуновский вестник. 2017. - № 1. - С. 63–67.
3. Ненайденко А.С., Поддубный В.И. Математическое моделирование движения колесной машины в горизонтальной плоскости // Вестник КрасГАУ. 2018. - № 3. - С. 72–77.