ОГЛАВЛЕНИЕ

ОТ СОСТАВИТЕЛЕЙ

Настоящие номера 3 и 4 журнала «Электронные библиотеки» включают статьи, подготовленные сотрудниками и студентами кафедры программной инженерии Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета (ВШ ИТИС КФУ).

Статьи, представленные в данном тематическом сборнике, охватывают несколько основных направлений деятельности кафедры: цифровое образование и использование программных инструментов в образовательном процессе, виртуальная реальность и компьютерные игры, мобильные технологии.

В контексте цифрового образования авторы нескольких статей обращают внимание на применение мобильных технологий в обучении для решения узкоспециализированных задач, рассматривают использование модели студента для задач образовательной аналитики и адаптивного обучения, а также приводят подход к реализации модели совместного обучения, основанной на стратегии STAD. Авторы предлагают также подходы к индивидуализации образования, касающиеся автоматизированной генерации индивидуальной траектории и расписания, а также формирования учебных групп и проектных команд с учетом компетенций и личностных качеств обучающихся. Кроме того, в этих работах раскрыта зависимость достоверности оценки знаний от визуального представления вопросов проверочного тестирования и приведена схема работы инструмента проектирования адаптивных тестов для microlearning-приложений, рассмотрены вопросы определения сложности курса и анализа связей расписания с успеваемостью студентов, описана концепция программного инструмента для обучения языку программирования Java с элементами игрофикации и особенности внедрения микросервисной архитектуры в проектную работу студенческой лаборатории.

В статьях, посвященных виртуальной реальности и разработке компьютерных игр, описаны концепции программного инструмента для генерации сценарного прототипа игры на основе текстового описания, предложен метод виртуализации запахов, а также представлены математические подходы для реализации методов синхронизации действий игрока и виртуального аватара.

В работах, описывающих новые решения в области мобильных разработок, рассмотрены методы модификации визуальных интерфейсов на основе ин-

дивидуальных характеристик пользователя, описан программный инструмент для автоматической генерации кода визуального интерфейса на основе данных графического редактора, приведен алгоритм синхронизации сессий дополненной реальности на мобильных устройствах.

Кроме того, в тематическом сборнике представлены статьи с описанием программных инструментов, использующих технологию распределенных реестров для решения специализированных задач, способов конфигурирования веб-приложений на основе UML-диаграмм, а также подходов к извлечению заголовков из научных документов.

И. С. Шахова, А. Ф. Хасьянов

ЧАСТЬ 1

УДК 004.414.3

ГЕНЕРАЦИЯ ИНДИВИДУАЛЬНЫХ ОБРАЗОВАТЕЛЬНЫХ ТРАЕКТОРИЙ И РАСПИСАНИЯ ОБУЧЕНИЯ В ПАРАДИГМЕ ИНДИВИДУАЛИЗАЦИИ ОБРАЗОВАНИЯ

М. М. Абрамский[1], Э. Ф. Батырова[2], А. Р. Марданова[3], Т. А. Ахметзянова[4]

[1-4] *Высшая школа информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета*

[1]ma@it.kfu.ru, [2]elviraa3805@gmail.com, [3]aigulmardanova96@gmail.com, [4]tanya.push0494@gmail.com

Аннотация

Представлен подход к индивидуализации образования, основанный на автоматизированной генерации индивидуальной образовательной траектории и расписания, учитывающих особенности каждого обучающегося и его пожелания. Описан принцип действия разработанных инструментов генерации. Затронуты вопросы применения разработанных подходов и инструментов в высшем образовании.

Ключевые слова*: smart-образование, индивидуализация образования, индивидуальный учебный план, индивидуальная траектория, генетические алгоритмы, генерация расписания*

ВВЕДЕНИЕ

Согласно федеральной целевой программе развития образования одной из важных задач является «индивидуализация образовательных траекторий с учетом личностных свойств, интересов и потребностей обучающегося» [1].

Под *индивидуализацией* понимается организация учебного процесса, учитывающая индивидуальные особенности каждого студента. Это помогает создать оптимальную среду для максимальной реализации возможностей обучающегося [2].

Под *индивидуальной образовательной траекторией* в широком смысле понимается спроектированный на основе компетенций и характеристик обуча-

ющегося перечень дисциплин и проектов, который студент должен пройти для достижения наилучших для себя образовательных результатов.

В рамках решения задач составления индивидуальных траекторий все большее развитие получают концепция *Smart University («Smart-университет», «Умный университет»)* и ее неотъемлемая часть, посвященная непосредственно образованию, – *Smart Education (Smart-образование)*, предполагающая адаптивную реализацию образовательного процесса с применением интеллектуальных (*smart*) цифровых инструментов, осуществляющих индивидуализацию образовательной траектории с помощью средств машинного обучения, анализа данных, визуализации и др.

Реализация концепции Smart University, в том числе и Smart Education, является одной из стратегических инициатив Казанского (Приволжского) федерального университета (КФУ), что отражено в 4-м этапе Плана мероприятий по реализации программы повышения конкурентоспособности КФУ [3].

Концепция Smart-университета, описывающая, в том числе, вопросы кампуса, электронного образования, коллаборации университетов для объединения ресурсов, подробно описана в [4].

Концепция Smart Education в литературе понимается достаточно широко. Например, в работе [5] к Smart-образованию относят, в том числе, вопросы организации смешанной формы обучения, предполагающей синергию обучения в классе и online-обучения, вопросы организации исследовательской, познавательной и проектной деятельностей студентов, особенности механизмов поддержания актуальных компетенций студентов и преподавателей, соответствующих критериям рынка труда. В работе [6] достаточно подробно изучены вопросы технической реализации облачной системы управления образовательным контентом.

Не умаляя всех достоинств концепции Smart Education, в данной работе будем рассматривать именно вопросы индивидуализации и ее воплощения в образовательный процесс вуза с помощью специальных цифровых инструментов, при этом будут затронуты и смежные компоненты концепции.

Работа построена следующим образом. В первом разделе описан предлагаемый авторами подход к реализации индивидуализации в вузе. Во втором разделе разобраны вопросы генерации индивидуальных образовательных тра-

екторий, третий раздел касается механизмов сбора сведений, необходимых для генерации расписания обучения, а также алгоритма генерации.

1. ПРЕДЛАГАЕМЫЙ ПОДХОД К ИНДИВИДУАЛИЗАЦИИ ОБРАЗОВАНИЯ

В данной работе под *индивидуальным учебным планом* и *индивидуальной образовательной траекторией* будет подразумеваться одно и то же понятие. Отметим все же, что под индивидуальной образовательной траекторией понимается больше содержательная часть индивидуального образования, а под индивидуальным учебном планом – ее реализация в качестве формального документа. Однако в рамках данной работы это различие не является существенным.

Учебный план и расписание занятий являются неотъемлемой частью образовательного процесса. В рамках индивидуализации компетенции учебного плана, предусмотренные профильными ФГОСами, должны строго коррелировать с индивидуальной образовательной траекторией каждого студента. Вместе с этим эти же компетенции должны соответствовать требованиям рынка труда, что отражено во внедрении в высшее образование профессиональных стандартов (ФЗ №122 от 2 мая 2015 г. об обязательном использовании профессиональных стандартов при установлении требований к профессиональным компетенциям выпускников) [7].

Имеет место постоянная работа по адаптации учебного плана к актуальным требованиям рынка, иногда выходящим за пределы профессиональных стандартов. При внедрении индивидуализации происходит переход от общего учебного плана к индивидуальному для каждого студента, и работа по адаптации всех индивидуальных учебных планов, очевидно, является трудозатратной.

При генерации расписания на основе составленного индивидуального учебного плана необходимо учитывать дополнительную информацию: личные предпочтения студентов и преподавателей, совместимость их компетенций и интересов, характеристики аудиторного фонда. Данная задача также не может быть решена без разработки специальных информационных систем и цифровых инструментов, которые смогут решить ее автоматически.

Принципиальная схема работы разрабатываемых инструментов представлена на рисунках 1 и 2.

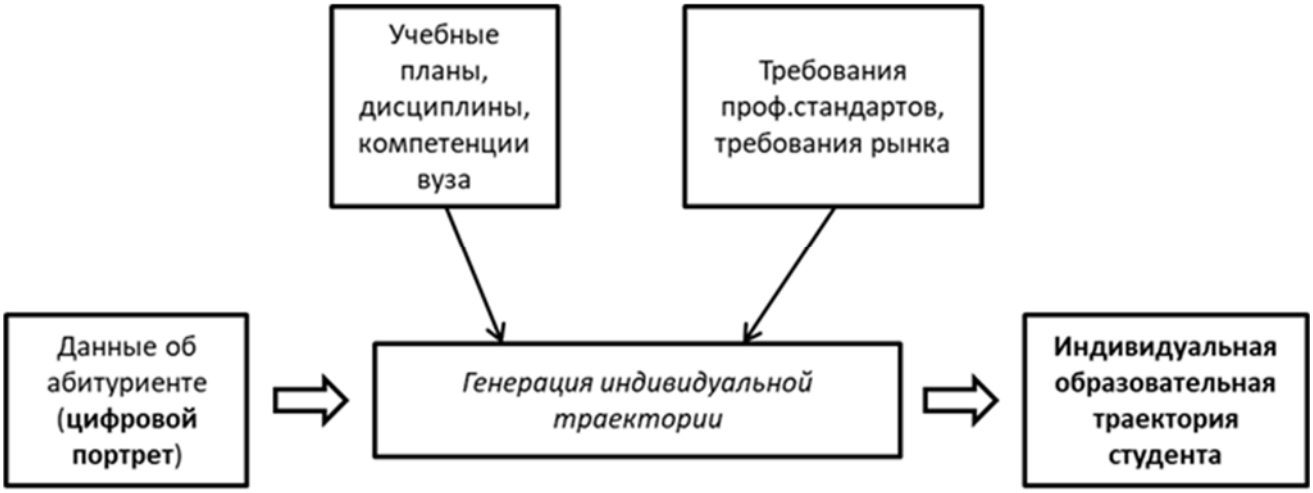


Рис. 1. Принцип работы модуля генерации индивидуальной образовательной траектории

Рис. 2. Принцип работы платформы сбора требований к расписанию и его генерации

Отметим, что вопросы устройства цифрового портрета обучающегося требуют отдельного рассмотрения, что будет сделано в последующих публикациях. На данный момент цифровым портретом студента можно считать хранимую в базе данных информацию, включающую перечень его оценок аттестата средней школы, баллы ЕГЭ, иные достижения и умения студента. Все данные портрета обучающегося хранятся в формате «строка/значение» – «компетенция/уровень владения», «предмет/оценка» или «умение/уровень».

2. РЕАЛИЗАЦИЯ ГЕНЕРАЦИИ ИНДИВИДУАЛЬНОЙ ОБРАЗОВАТЕЛЬНОЙ ТРАЕКТОРИИ ОБУЧАЮЩЕГОСЯ

Для построения индивидуальных учебных планов разработан алгоритм, который изображен на рис. 3.

Алгоритм начинается с обработки цифрового портрета абитуриента, в данном случае – перечня знаний, умений, навыков, компетенций и достижений. Также на начальном этапе алгоритма происходит извлечение компетенций из ФГОС и учебных планов, а также трудовых функций из профессиональных стандартов. В будущем планируется добавить в этот этап автоматизированную обработку вакансий и трендов рынка труда.

Следующим шагом производится вычисление корреляции элементов цифрового портрета с компетенциями учебного плана с целью обнаружения схожести этих значений. Сравнение производится с помощью алгоритма шинглов [8] – поиска копий и дубликата. Предварительно каждая строка очищается от знаков препинания и стоп-слов (предлогов, союзов, местоимений). Затем каждая строка представляется в виде множества *n*-грамм – последовательностей слов фиксированной длины *n* (или шинглов). С целью увеличения эффективности шинглы из текстовых подстрок преобразуются в числа – дактилограммы. Элементы двух множеств сравниваются по количеству схожих элементов. В алгоритме установлен лимит количества схожих элементов, при превышении которого делается вывод о соответствии навыка студента и компетенции в учебном плане.

Здесь сразу же стоит отметить, что в настоящее время абитуриент самостоятельно выбирает, по какой образовательной программе он хочет учиться, и, если его результаты единого государственного экзамена (ЕГЭ) соответствуют проходному баллу, то он автоматически поступает на выбранное направление. Соответствие между уровнем компетенций студента и возможностью учиться по выбранному направлению определяется лишь перечнем из 3 или 4 результатов ЕГЭ по общеобразовательным предметам, требуемым для данного направления.

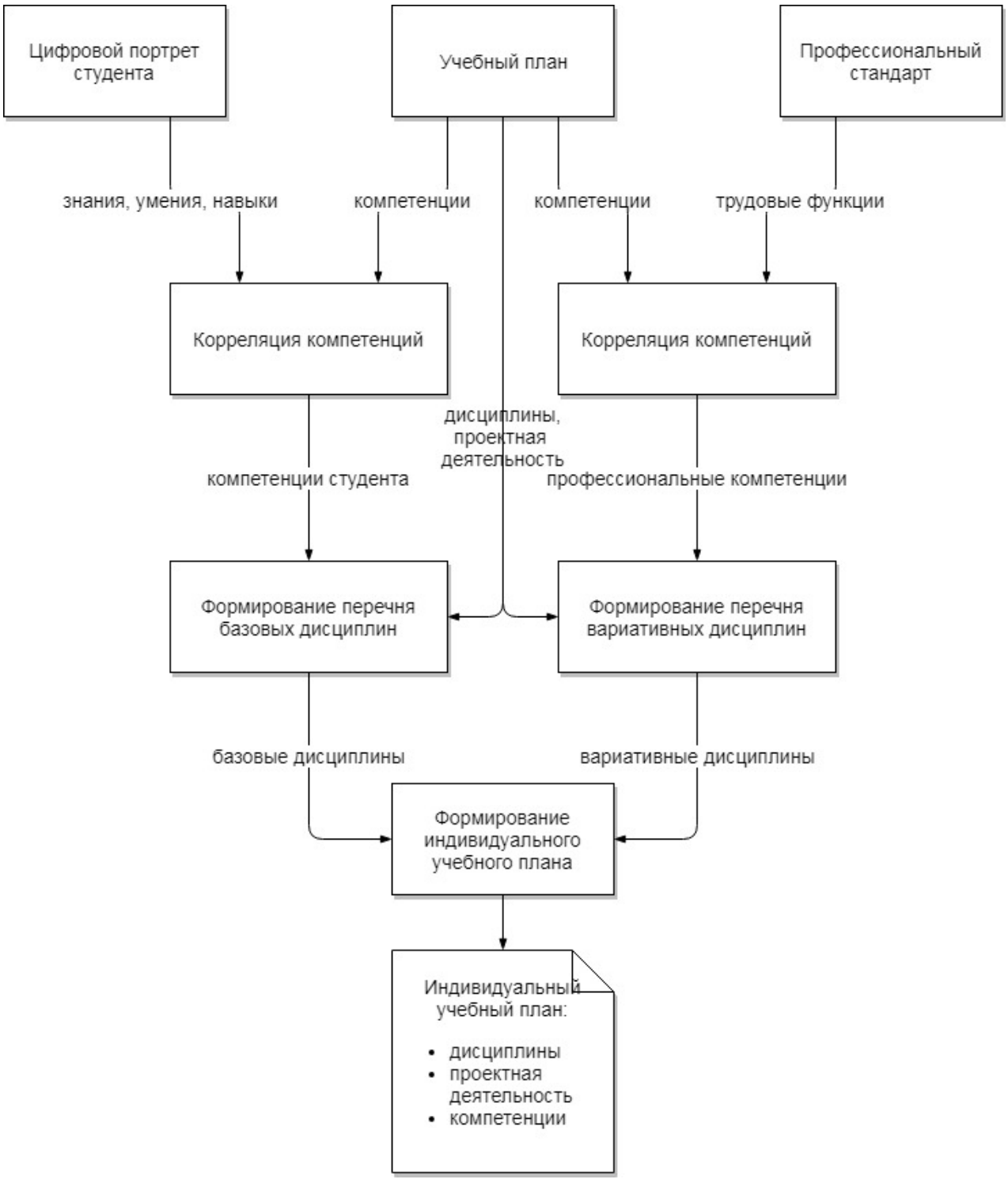


Рис. 3. Алгоритм построения индивидуальных учебных планов

В настоящее время стало очевидным, что оценки за ЕГЭ не являются исчерпывающим описанием компетенций абитуриента, особенно в рамках индивидуализации образования. В поиске более тесных связей между компетенция-

ми студента (чем он владеет) и компетенциями учебного плана (чем он может овладеть) и заключается описанный этап вычисления корреляции.

Параллельно теми же средствами происходит аналогичное вычисление корреляции трудовых функций из профессиональных стандартов с компетенциями учебных планов. Трудовые функции определяются профессиональной деятельностью, которой студент желает заниматься после окончания университета. Таким образом, для трудовых функций подбираются профессиональные компетенции из учебных планов.

На выходе первого шага алгоритма имеется перечень компетенций, которые обучающийся (на основе своих текущих навыков) сможет сформировать, обучаясь в вузе. Отметим, что компетенции не обязательно должны соответствовать только одному учебному плану. Наоборот, корреляции с несколькими учебными планами позволят создать индивидуальный план, способствующий подготовке востребованного междисциплинарного специалиста.

Среди подходящих учебных планов выбирается учебный план, обозначающийся как базовый. Наличие базового учебного плана необходимо, поскольку по формальным правилам индивидуальный учебный план должен быть встроен в один из базовых учебных планов. Подразумевается, что из базового учебного плана в индивидуальный учебный план попадут базовые предметы, такие, как, например, философия, информатика, физика.

По профессиональным компетенциям подбираются дисциплины, которые могут их сформировать. Данные дисциплины называются вариативными. Отметим, что в рамках настоящей работы к вариативным дисциплинам мы относим также дисциплины по выбору. В будущих работах планируется возможность автоматического составления формата проектной деятельности для развития профессиональных компетенций.

Таким образом, по завершении работы алгоритма имеется перечень дисциплин, формирующих компетенции, учитывающие как потребности студента, так и требования профессионального стандарта. Вместе они представляют собой индивидуальный учебный план.

Модуль с предложенным выше алгоритмом разработан на языке программирования Java с использованием фреймворка Spring [9]. Для хранения

данных была использована СУБД PostgreSQL [10]. Для анализа русскоязычного текста была использована утилита MyStem [11].

3. СБОР СВЕДЕНИЙ ДЛЯ ГЕНЕРАЦИИ РАСПИСАНИЯ И АЛГОРИТМ ГЕНЕРАЦИИ

В качестве связующего звена между составленным индивидуальным планом и генерацией расписания была разработана специальная платформа, обеспечивающая алгоритм генерации расписания необходимыми данными – как путем ручного сбора пожеланий к расписанию со стороны студентов и преподавателей, так и с помощью автоматического создания пожеланий и требований к расписанию на основе анализа цифровых портретов студентов и преподавателей, индивидуальных траекторий, формальных ограничений, соображений эффективности и др. Пример страницы ввода пожелания представлен на рис. 4.

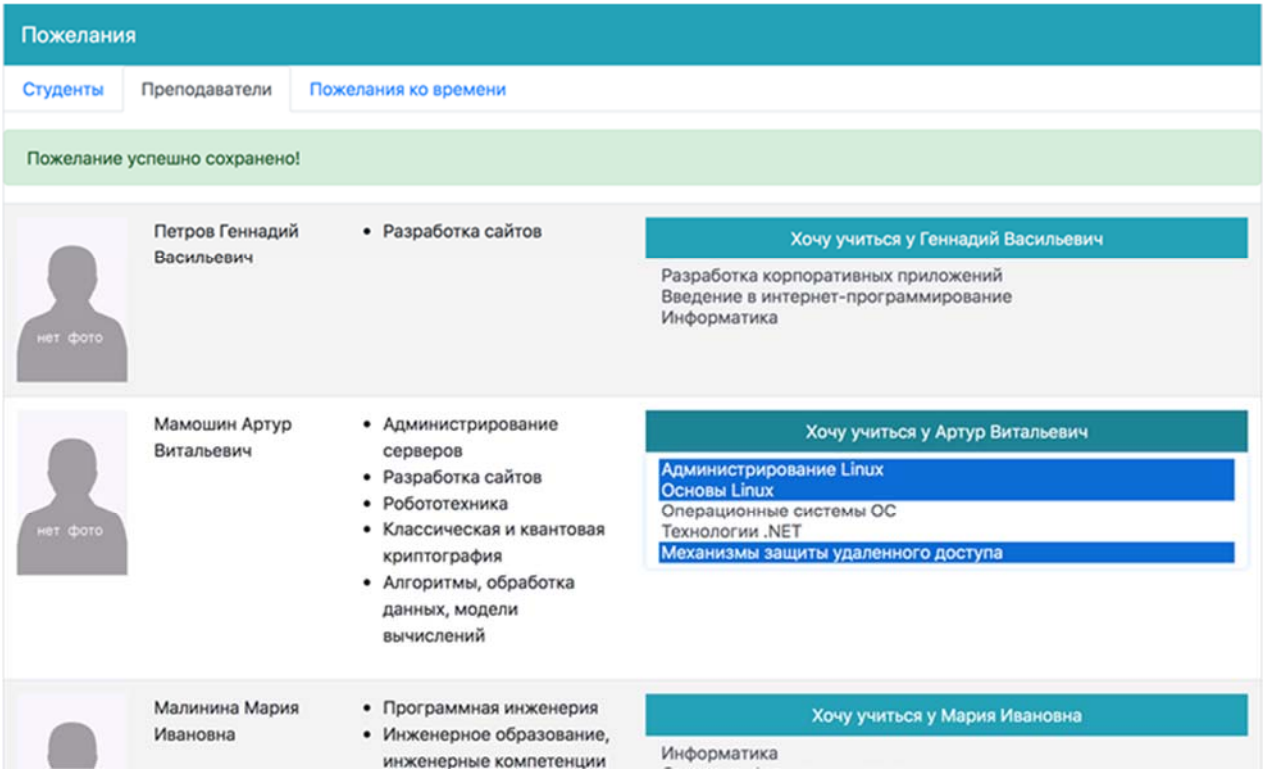


Рис. 4. Страница ввода пожеланий к обучению у конкретного преподавателя по конкретному предмету

Пожелание представляет собой условие к расписанию, выполнение которого либо обязательно, либо желательно для целей индивидуализации, и кото-

рое опирается на составные части цифровых портретов сущностей университета. Был составлен расширенный (в контексте индивидуализации) каталог возможных пожеланий и требований (рис. 5), звездочкой (*) отмечены пожелания и требования, для которых доступна автоматическая генерация.

1	Тип	Название
1	Студент – Студент (*)	Пожелание студента к студенту
2	Студент – Студент – Предмет (*)	Пожелание студента к студенту для определенного предмета
3	Студент – Преподаватель	Пожелание студента к преподавателю
4	Студент – Преподаватель – Предмет	Пожелание студента к преподавателю для определенного предмета
5	Преподаватель – Студент (*)	Пожелание преподавателя к студенту
6	Преподаватель – Студент – Предмет (*)	Пожелание преподавателя к студенту для определенного предмета
7	Пользователь – Начало занятий	Пожелание ко времени первой пары (не раньше, чем)
8	Пользователь – Начало занятий – Предмет	Пожелание ко времени первой пары (не раньше, чем) для определенного предмета
9	Пользователь – Конец занятий	Пожелание ко времени последней пары (не позже, чем)
10	Пользователь – Конец занятий – Предмет	Пожелание ко времени последней пары (не позже, чем) для определенного предмета
11	Преподаватель – Аудитория (*)	Пожелание преподавателя к аудитории для определенного предмета
12	Преподаватель – Оборудование	Пожелание преподавателя к наличию оборудования в аудитории для определенного предмета
13	Предмет – Оборудование	Требования к наличию оборудования в аудитории для определенного предмета
14	Предмет – Аудитория (*)	Пожелание к аудитории для определенного предмета

Рис. 5. Каталог пожеланий и требований. (*) – доступна автоматическая генерация

Приведем пример автоматической генерации пожелания/требования «Студент–студент». Предположим, что в рамках генерации необходимо собрать в группы студентов, которые имеют в своем цифровом портрете и индивидуальных траекториях схожие компетенции. Для оценки «схожести» была предложена достаточно простая формула:

$$similarityC=|PC1\cap C2|+|C1\cap PC2|+|C1\cap C2|,$$

где PC1 – множество личных компетенций первого студента (из цифрового портрета), PC2 – множество личных компетенций второго студента, C1 – множество развиваемых компетенций первого студента (из индивидуального учебного плана), C2 – множество развиваемых компетенций второго студента.

При включении параметра «автоматическая генерация» для данного требования в алгоритм генерации расписания попадут условия сочетания студентов по величине данного признака. Отметим, что данная формула легко поддается изменению на другую в рамках системы.

Оговоримся, что целью данной работы не является выработка критериев наиболее эффективного распределения студентов. К примеру, возможно будет

эффективным распределить иностранных студентов, владеющих английским языком, к преподавателю, который тоже им владеет. Или же наоборот, возможно, для развития их компетенций коммуникации на русском языке эффективнее будет распределить их к преподавателю, не владеющему английским языком.

Разрабатываемые инструменты позволяют делать гибкую настройку этих критериев, а их наполнение с целью сделать распределение эффективным с той или иной точки зрения отдается полностью пользователям-экспертам, управляющим алгоритмом генерации расписания.

Задача составления расписания неоднократно решалась для классических случаев, на рынке существует целая серия приложений, генерирующих расписание занятий [12–14]. Также осуществлялись попытки ее решения с учетом некоторых особенностей, например, пожеланий преподавателей [15]. Недостатком существующих решений является их исходное несоответствие принципу индивидуализации образования, а также разработанной выше модели.

В качестве способа реализации алгоритма генерации расписания в контексте индивидуализации нами был выбран генетический алгоритм, зарекомендовавший себя как надежный подход к решению задачи составления расписания [16]. Этот способ позволяет строить сложные конструкции путем генерации начальной популяции и скрещивания между элементами с целью последующего улучшения.

Схема разработанного алгоритма представлена на рис. 6. Первое поколение расписаний строится на основе сущностей университета, конфигурации расписания и системных ограничений.

Конфигурация расписания содержит такие параметры, как количество учебных дней в неделе, количество пар в день, максимальное число студентов в группе. Системные ограничения – это такие ограничения, без которых расписание не может быть построено. Существует три типа системных ограничений: преподаватель не может одновременно находиться на двух парах, студент не может одновременно находиться на двух парах, аудитория не может быть занята одновременно двумя парами. Системные ограничения также проверяются после каждой итерации алгоритма, за счет чего расписание всегда остается корректным.

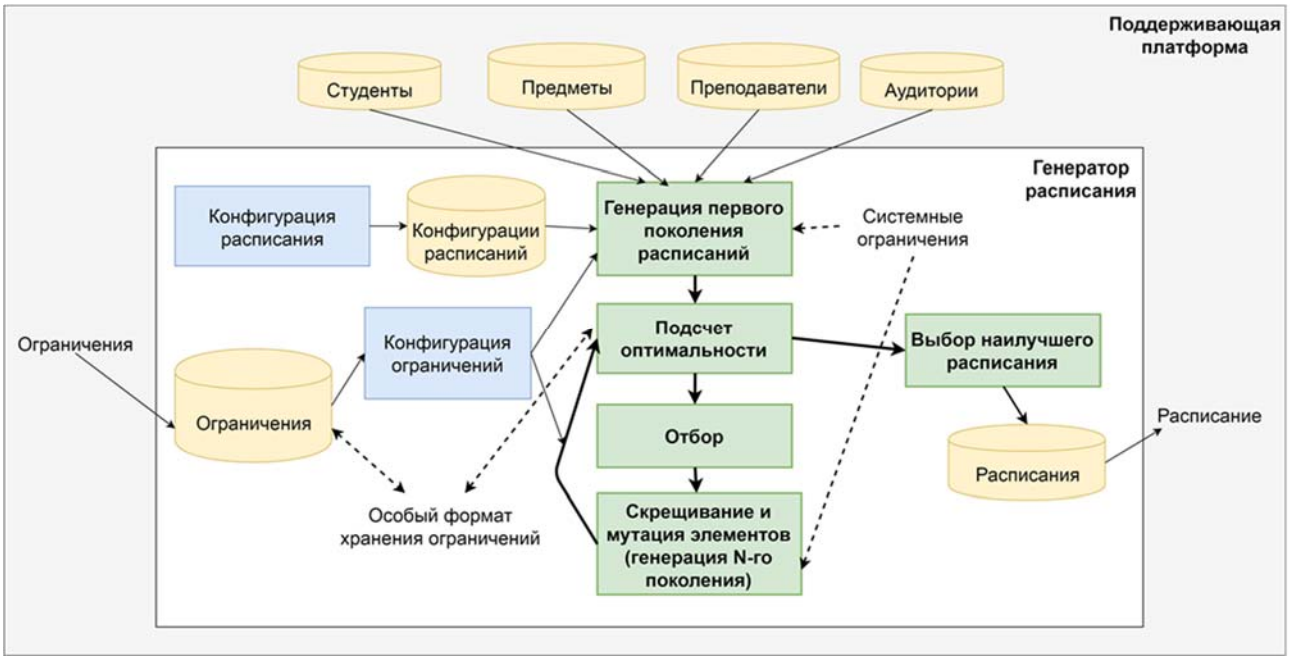


Рис. 6. Схема алгоритма генерации расписания. Под ограничениями понимаются все требования к расписанию, в том числе и пожелания

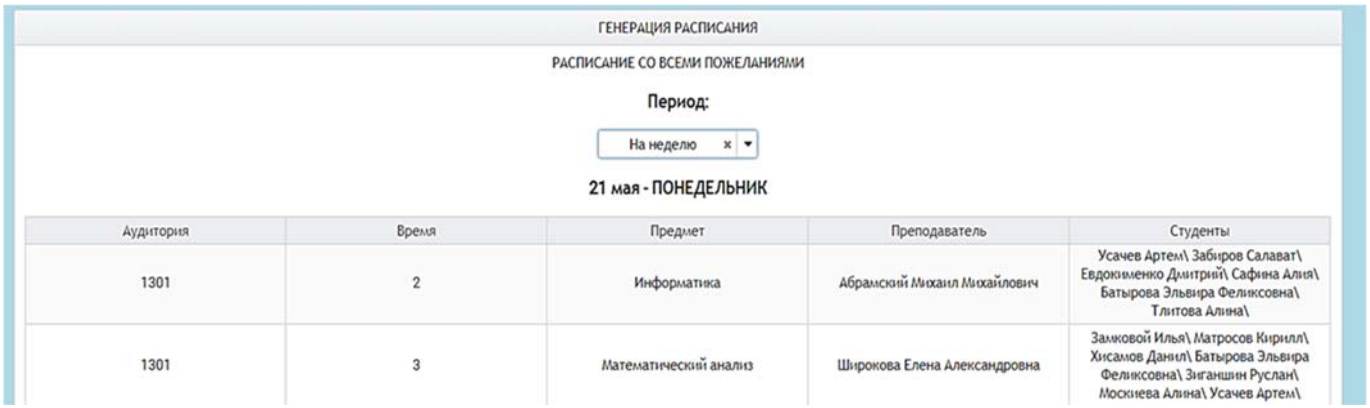
Подсчет оптимальности расписаний происходит с помощью таблицы, строки которой содержат участвующие в ограничении сущности и boolean-колонку, отображающую выполнение или невыполнение условия.

Тип	Включен	Всего	Выполнено
Студент - Студент - Предмет	☑	20	7
Студент - Преподаватель	☑	20	5
Преподаватель - Начало пар - Конец пар	☑	20	8

Рис. 7. Возможность управления ограничениями в процессе генерации расписания

В ходе работы на алгоритм можно влиять путем включения и выключения типов ограничений (рис. 7). Так можно сначала включить только важные типы ограничений, а когда они будут выполнены, добавить остальные. На данный момент реализованы выключения/включения ограничений по их типу, в будущем планируется проработать вопрос управления отдельными ограничениями.

По окончании работы генератор выбирает расписание с наибольшим процентом выполненных условий. Полученное расписание сохраняется в базу данных, откуда его можно выгрузить для просмотра (рис. 8).


ГЕНЕРАЦИЯ РАСПИСАНИЯ

РАСПИСАНИЕ СО ВСЕМИ ПОЖЕЛАНИЯМИ

Период:

На неделю

21 мая - ПОНЕДЕЛЬНИК

Аудитория	Время	Предмет	Преподаватель	Студенты
1301	2	Информатика	Абрамский Михаил Михайлович	Усачев Артем\ Забиров Салават\ Евдокименко Дмитрий\ Сафина Алия\ Батырова Эльвира Феликсовна\ Тлитова Алина\
1301	3	Математический анализ	Широкова Елена Александровна	Замковой Илья\ Матросов Кирилл\ Хисамов Данил\ Батырова Эльвира Феликсовна\ Зиганшин Руслан\ Москиева Алина\ Усачев Артем\

Рис. 8. Просмотр расписания. В колонке «время» указан номер пары по расписанию учебного заведения (для КФУ: 2 – 10:10, 3 – 11:50)

Было проведено тестирование модуля генерации расписания на шестидневной учебной неделе с шестью парами в день. Общее количество студентов в тесте – 144 (студентов в группе – максимум 25), количество преподавателей – 15, предметов – 15. Для сравнения каждый тип ограничения был протестирован отдельно (по 20 ограничений в каждом тесте). Наименьший процент выполнения получали пожелания от студентов, а наивысший – от преподавателей. Это объясняется тем, что учесть пожелания каждого студента из группы сложнее, чем пожелание одного преподавателя. Также был произведен общий тест из 140 ограничений различных типов, 80% из которых были выполнены. Это значит, что алгоритм способен учитывать значительную долю пожеланий и рекомендаций.

Платформа для сбора пожеланий и генерации требований, а также модуль генератора расписания были реализованы в виде веб-приложения. Клиентская часть написана на фреймворке Angular [17], что позволяет динамически следить за процессом генерации, а серверная – с использованием Spring Framework. Для хранения данных также использовалась СУБД PostgreSQL.

ЗАКЛЮЧЕНИЕ

Реализованы модуль для построения индивидуальных учебных планов, платформа для сбора пожеланий и требований к расписанию, а также модуль генерации расписания. Представлена модель индивидуализации образования, которую разработанные инструменты могут обеспечить. Проведены эксперименты по проверке работоспособности инструментов на тестовых данных.

В дальнейшем планируется развитие разработанных инструментов в сторону гибкости, удобства использования и большей функциональности. Будет поднят вопрос об эффективной визуализации индивидуальной образовательной траектории и расписания обучения. Будут подробнее рассмотрены цифровой портрет студента и механизмы его формирования.

Работа выполнена в рамках деятельности лаборатории Smart Education Lab Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета.

СПИСОК ЛИТЕРАТУРЫ

1. Концепция Федеральной целевой программы развития образования на 2016–2020 годы / Распоряжение Правительства Российской Федерации от 29.12.2014 № 2765-р. URL: http://government.ru/media/files/mIorxfXbbCk.pdf

2. *Cheng, Yin Cheong.* New Paradigm for Re-engineering Education: Globalization, Localization and Individualization. Springer, 2005. 26 p.

3. План мероприятий по реализации программы повышения конкурентоспособности ФГАОУ ВО «Казанский (Приволжский) федеральный университет» на 2013–2020 годы (4-й этап – 2018–2020 гг.). URL: http://kpfu.ru/portal/docs/F_1176208685/DK.KFU_4.etap._utverzh.pdf

4. Smart Universities. Concepts, systems and Technologies. V.L. Uskov, J.P. Bakker, R.J. Howlet, L.C. Jain (Eds.). Springer Inter. Publishing, 2018. 435 p.

5. *Тихомиров В.П., Днепровская Н.В.* Смарт-образование как основная парадигма развития информационного общества // Современные информационные технологии и ИТ-образование. 2015. Т. 1. № 11. С. 9–13.

6. *Ji-Seong Jeong, Mihye Kim and Kwan-Hee Yoo.* A Content Oriented Smart Education System based on Cloud Computing // Int. J. of Multimedia and Ubiquitous

Engineering. 2013. Vol. 8, No. 6. P. 313–328. URL: http://dx.doi.org/10.14257/ijmue.2013.8.6.31

7. Федеральный закон от 02.05.2015 г. № 122-ФЗ «О внесении изменений в Трудовой кодекс Российской Федерации и статьи 11 и 73 Федерального закона «Об образовании в Российской Федерации». URL: http://static.kremlin.ru/media/acts/files/0001201505020007.pdf

8. *Broder A.* On the resemblance and containment of documents // Compression and Complexity of Sequences (SEQUENCES'97). IEEE Computer Society, 1998. P. 21–29.

9. Spring Framework. URL: https://projects.spring.io/spring-framework

10. PostgreSQL: The World's Most Advanced Open Source Relational Database. URL: https://www.postgresql.org/

11. MyStem. Yandex URL: https://tech.yandex.ru/mystem/

12. Bullet Education Scheduling and Timetabling. URL: https://www.bulletsolutions.com/education-scheduling-timetabling/

13. Mimosa Scheduling Software. URL: http://www.mimosasoftware.com/

14. Snap Schedule. URL: https://www.snapschedule.com/Industry/Education/

15. «Официальный сайт программного продукта БИТ.ВУЗ.РАСПИСАНИЕ». URL: http://www.pulsar.ru/progs/1904/

16. *Lim C., Sim E.* Production Planning in Manufacturing / Remanufacturing Environment using Genetic Algorithm // GECCO 2005: Proc. of the 2005 Conference on Genetic and Evolutionary Computation, 2005. P. 2217–2218. URL: http://www.cs.bham. ac.uk/~wbl/biblio/gecco2005/docs/p2217.pdf

17. Официальный сайт фреймворка Angular 5. URL: https://angular.io/

GENERATION OF INDIVIDUAL EDUCATIONAL ROUTES AND LEARNING SCHEDULE IN INDIVIDUALIZATION PARADYGM

M. M. Abramskiy[1], E. F. Batyrova[2], A. R. Mardanova[3], T. A. Akhmetzyanova[4]

[1-4] *Higher School for Information Technologies and Intelligent Systems of Kazan (Volga Region) Federal University*

[1]ma@it.kfu.ru, [2]elviraa3805@gmail.com, [3]aigulmardanova96@gmail.com, [4]tanya.push0494@gmail.com

Abstract

An approach for individualization of education based on automatic generation of individual educational routes and schedule with respect to individual features and wishes of student. A working principle of developed instruments is shown. Question of the application of developed approaches and instruments in higher education are raised.

Keywords*: Smart Education, individualization of education, individual educational program, individual educational route, genetic algorithms, schedule generation*

REFERENCES

1. Kontseptsiya Federal'noj tselevoj programmy razvitiya obrazovaniya na 2016-2020 gody / Rasporyazhenie Pravitel'stva Rossijskoj Federatsii ot 29.12.2014 No 2765-r. URL: http://government.ru/media/files/mlorxfXbbCk.pdf

2. *Cheng Yin Cheong*. New Paradigm for Re-engineering Education: Globalization, Localization and Individualization. Springer, 2005. 26 p.

3. Plan meropriyatij po realizatsii programmy povysheniya konkurentosposobnosti FGAOU VO «Kazanskij (Privolzhskij) federal'nyj universitet» na 2013–2020 gody (4-j ehtap – 2018–2020 gg.). URL: http://kpfu.ru/portal/docs/F_1176208685/ DK.KFU_4.etap._utverzh.pdf

4. Smart Universities. Concepts, systems and Technologies. V.L. Uskov, J.P. Bakker, R.J. Howlet, L.C. Jain (Eds.). Springer Inter. Publishing. 2018. 435 p.

5. *Tikhomirov V.P., Dneprovskaya N.V.* Cmart-obrazovanie kak osnovnaya paradigma razvitiya informatsionnogo obshhestva // Sovremennye informatsionnye tekhnologii i IT-obrazovanie. 2015. T. 1. № 11. S. 9–13.

6. *Ji-Seong Jeong, Mihye Kim and Kwan-Hee Yoo.* A Content Oriented Smart Education System based on Cloud Computing // Int. J. of Multimedia and Ubiquitous Engineering. 2013. Vol. 8, No. 6. P. 313–328. URL: http://dx.doi.org/10.14257/ijmue.2013.8.6.31

7. Federal'nyj zakon ot 02.05.2015 g. № 122-FZ «O vnesenii izmenenij v Trudovoj kodeks Rossijskoj Federatsii i stat'i 11 i 73 Federal'nogo zakona «Ob obrazovanii v Rossijskoj Federatsii». URL: http://static.kremlin.ru/media/acts/files/0001201505020007.pdf

8. *A. Broder.* On the resemblance and containment of documents // Compression and Complexity of Sequences (SEQUENCES'97). IEEE Computer Society, 1998. P. 21–29.

9. Spring Framework. URL: https://projects.spring.io/spring-framework

10. PostgreSQL: The World's Most Advanced Open Source Relational Database. URL: https://www.postgresql.org/

11. MyStem. Yandex URL: https://tech.yandex.ru/mystem/

12. Bullet Education Scheduling and Timetabling. URL: https://www.bulletsolutions.com/education-scheduling-timetabling/

13. Mimosa Scheduling Software. URL: http://www.mimosasoftware.com/

14. Snap Schedule. URL: https://www.snapschedule.com/Industry/Education/

15. Ofitsial'nyj sajt programmnogo produkta BIT.VUZ.RASPISANIE. URL: http://www. pulsar. ru/progs/1904/

16. *Lim C., Sim E.* Production Planning in Manufacturing / Remanufacturing Environment using Genetic Algorithm // GECCO 2005: Proc. of the 2005 Conference on Genetic and Evolutionary Computation, 2005. P. 2217–2218. URL: http://www.cs.bham. ac.uk/~wbl/biblio/gecco2005/docs/p2217.pdf

17. Официальный сайт фреймворка Angular 5. URL: https://angular.io/

СВЕДЕНИЯ ОБ АВТОРАХ

АБРАМСКИЙ Михаил Михайлович – старший преподаватель кафедры программной инженерии Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета.

Mikhail Mikhailovich ABRAMSKIY – senior lecturer at Department of Software Engineering of Higher School of ITIS KFU

email: ma@it.kfu.ru

БАТЫРОВА Эльвира Феликсовна – бакалавр Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета.

Elvira Felixovna BATYROVA – bachelor of Higher School of ITIS KFU.

email: elviraa3805@gmail.com

МАРДАНОВА Айгуль Рустамовна – бакалавр Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета.

Aygul Rustamovna MARDANOVA – Bachelor of Higher School of ITIS KFU.

email: aigulmardanova96@gmail.com

АХМЕТЗЯНОВА Татьяна Алексеевна – магистр программной инженерии Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета.

Tatiana Alekseevna AKHMETZYANOVA – Master of Software Engineering of Higher School of ITIS KFU

email: tanya.push0494@gmail.com

Материал поступил в редакцию 31 июля 2018 года

УДК 004.415.2+004.414.38

ПРИМЕНЕНИЕ ТЕХНОЛОГИЙ МОБИЛЬНОГО ОБУЧЕНИЯ ДЛЯ ФОРМИРОВАНИЯ КОМПЕТЕНЦИЙ ПРОЕКТНОГО МЕНЕДЖЕРА

М. М. Абызов[1], И. С. Шахова[2]

Высшая школа информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета

[1]mishabuzov@gmail.com, [2]is@it.kfu.ru

Аннотация

Представлен обзор способов измерения прогресса разработки программного продукта в рамках гибкой методологии SCRUM, а также дано описание разработки программного инструмента, отслеживающего текущее состояние проекта по временным характеристикам. Анализируя их, такой инструмент способен подсказать проектному менеджеру, на что именно ему следует обратить внимание в текущей проектной обстановке, и помочь в выборе действий для достижения эффективных результатов.

***Ключевые слова**: управление проектами, проектный менеджер, обучение проектных менеджеров, мобильное приложение, SCRUM, User Story, Story Point, Sprint, Sprint Backlog, Agile reporting, Mobile Learning*

ВВЕДЕНИЕ

На сегодняшний день внедрение гибких методологий разработки в процесс производства является весьма распространенной практикой. По данным исследований всемирной организации Project Management Institute, в 2017 году почти ¾ компаний так или иначе использовали гибкие методологии разработки в процессе управления своими проектами [1].

Ключевым принципом гибких методологий разработки является то, что они позволяют проводить периодическую корректировку действий, которая может привести к устойчивым и эффективным результатам [2]. В то же время, по данным статистики за 2017 год, примерно 97 млн. долларов на каждый инвестированный миллиард теряются в результате недостаточно эффективного процес-

са управления проектами [1]. Именно поэтому обучению проектных менеджеров следует уделять особое внимание.

Успех или неудача проекта во многом зависят от способности проектного менеджера правильно расставлять приоритеты между бюджетом проекта, его качеством и временем разработки, а также вовремя разрешать конфликты, возникающие в ходе жизненного цикла проекта [3]. В течение всего цикла разработки программного продукта уровень важности каждого из приоритетов может несколько раз изменяться. Всё это отражается и на временных рамках. Например, переоценка графика может оказаться фатальной для проекта, когда бюджет становится решающим фактором, и наоборот. Аналогично требования к качеству продукта должны быть изменены в определенных ситуациях, чтобы соответствовать реальности.

Чтобы правильно оценить ситуацию при довольно быстро меняющихся приоритетах, проектному менеджеру, безусловно, необходим опыт. Начинающим менеджерам может быть крайне трудно оценить текущую обстановку и принять решение, благодаря которому разработка программного продукта будет завершена точно в срок, и сам программный продукт при этом будет соответствовать основным требованиям заказчика.

КЛЮЧЕВЫЕ ОСОБЕННОСТИ МЕТОДОЛОГИИ SCRUM

Согласно исследованию, проведенному Agile Survey, наиболее распространенным методом среди гибких методологий разработки является Scrum (используемый 66% из 6042 опрошенных респондентов) [4]. В рамках того же опроса было установлено, что потеря контроля в управлении проектом является одной из самых серьезных проблем всех гибких методологий. Поэтому постоянное наблюдение за происходящими событиями в проекте имеет решающее значение для обеспечения видимости, контроля и адаптации.

Далее будут рассмотрены общие принципы и ключевые характеристики методологии Scrum. В идеале команда разработки в рамках методологии Scrum не должна превышать 9 человек – с большим числом участников, как правило, трудно поддерживать связь, что снижает эффективность данной методологии [5]. Каждый спринт может длиться не более 1 месяца и начинается с планового собрания, на котором обсуждаются задачи предстоящего спринта. Их выбирают

из общего списка задач, называемого Product Backlog. После этого участники команды задают время, необходимое для реализации проекта. В конце каждого спринта происходят обзор всех выполненных задач и демонстрация результата. Также в Scrum широко распространена практика ежедневных собраний, в ходе которых обсуждаются текущий прогресс и возможные препятствия при реализации задач.

КРИТЕРИИ ЭФФЕКТИВНОСТИ РЕЗУЛЬТАТОВ

Существуют хорошо известные критерии измерения прогресса в рамках гибкой методологии Scrum – это скорость и объем оставшейся работы [4]. Далее эти критерии будут рассмотрены на примерах соответствующих им графиков.

1) *Диаграмма скорости выполнения проекта*

Под скоростью выполнения подразумевается объем работы, который необходимо выполнить в текущем спринте [4]. Она выражается в Story Point. Как правило, 1 Story Point соответствует 1 рабочему дню. Планируемая скорость выполнения задач оценивается командой в начале каждого спринта и рассчитывается так, чтобы успеть выполнить все User Story (функциональные требования), запланированные на спринт. Фактическую же скорость можно вычислить в конце спринта путем суммирования Story Point для всех User Story, принятых владельцем продукта.

Рассмотрим, к примеру, диаграмму, отражающую скорость выполнения задач спринта на примере проекта создания веб-сайта для словенской издательской компании.

Результаты исследования проекта, отраженного на диаграмме Рис. 1, показали, что фактическая скорость выполнения спринта оказывалась всегда меньше запланированной [4]. Для первых спринтов это может быть объяснено отсутствием опыта в рамках нового проекта, поэтому и планируемая скорость вначале оценивалась, просто считая рабочий день (т. е. Story Point) равным 6 часам эффективной работы. Но самая низкая скорость оказалась в пятом спринте, когда в команду были добавлены два новых разработчика, которые по задумке должны были увеличить объем выполненной работы, но вместо этого привели проект к срыву, снизив производительность других членов команды. Существенное различие между запланированной и фактической скоростями в спринте 6 объясняется переоценкой реальных возможностей команды: вместо того, чтобы

приспособить оценку к фактическим достижениям в предыдущих спринтах, команда поддалась давлению приближающегося срока и обещала предоставить больше функциональности, чем это было реально возможно.

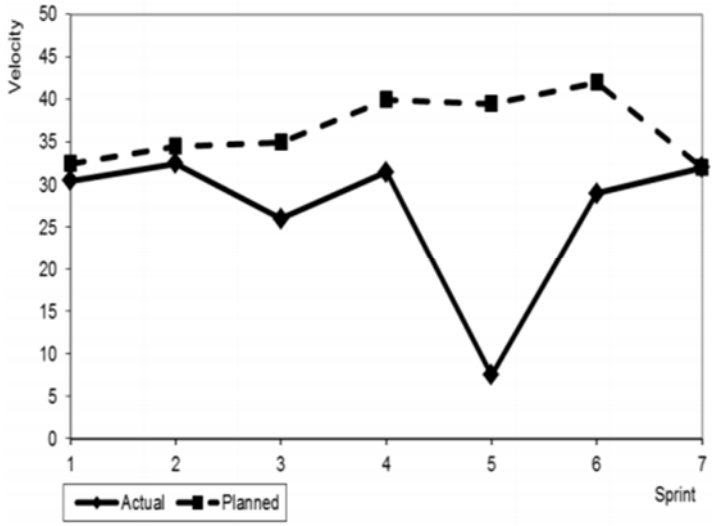


Рис. 1. Диаграмма скорости выполнения спринтов. Отражает действительную и запланированную скорости [4]

2) *Диаграмма сгорания для проекта*

Диаграмма сгорания, представленная на Рис. 2, сигнализирует о том, что у команды не получилось завершить весь проект в течение 7 спринтов, как ожидалось изначально. Основной причиной этого, как правило, являются новые требования, которые изначально не были зафиксированы в Product Backlog, но, тем не менее, постоянно добавлялись заказчиком в течение всего проекта [4]. В таком случае необходимо пересмотреть функционал проекта, чтобы выделить основные моменты и завершить проект в более приемлемые сроки. Используя такой подход, издательская компания пересмотрела содержимое Product Backlog и успешно выпустила проект после 9 спринтов. Следует отметить, что описанный подход вполне можно считать оправданным: куда лучше пересмотреть требования и выпустить продукт во вновь установленные сроки, чем рисковать не выпустить его вовсе. Тем более, что, согласно исследованиям в области разработки программного обеспечения, почти 65% функционала выпускаемых программ используются в действительности редко, а часть не используется вообще [6].

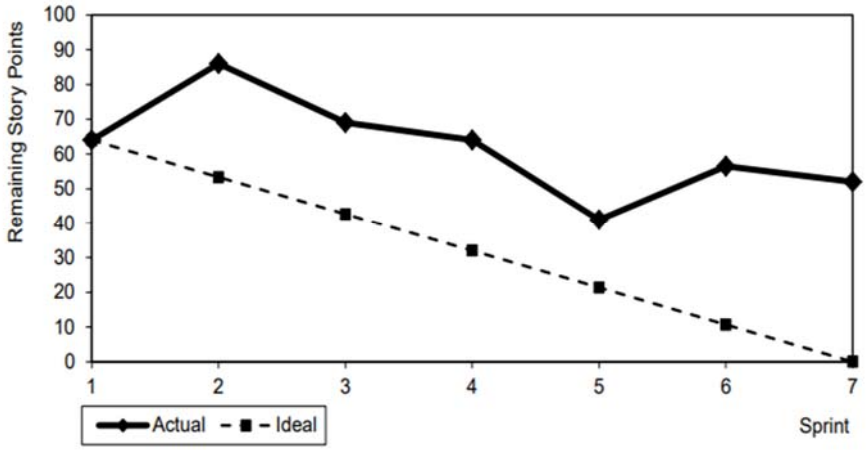


Рис. 2. Диаграмма сгорания для проекта к началу 7-го спринта [4]

3) *Диаграмма сгорания для спринта*

Эта диаграмма очень похожа на диаграмму сгорания для всего проекта, но вместо общей картины в ней отображено количество работы, которую необходимо выполнить, чтобы завершить текущий спринт. На горизонтальной линии отображается количество дней спринта, в то время как вертикальная линия показывает количество оставшихся рабочих часов. Диаграмма обновляется каждый день, суммируя оценки для всех задач. Текущая линия тренда отражает, сумеет ли команда выполнить все задачи до конца спринта. В отличие от диаграммы на Рис. 2, диаграмма сгорания для текущего спринта на Рис. 3 отражает более детально то, что команда выполнила почти все поставленные перед ней задачи в рамках текущего спринта [4].

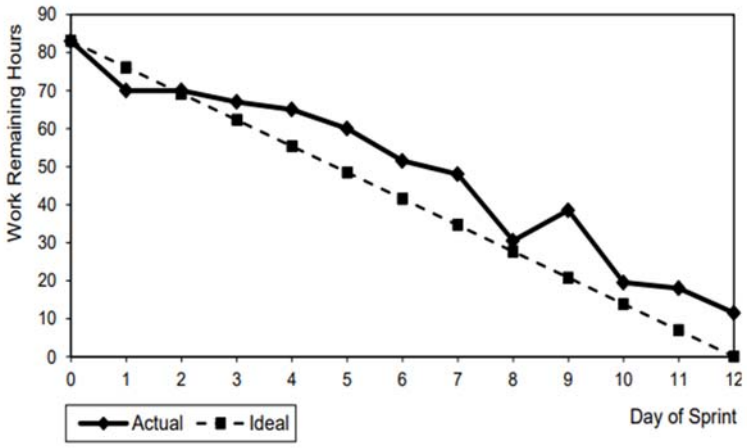


Рис. 3. Пример диаграммы сгорания для спринта [4]

На основе анализа данной диаграммы проектный менеджер может сделать и другие выводы о том, как повысить эффективность работы своей команды.

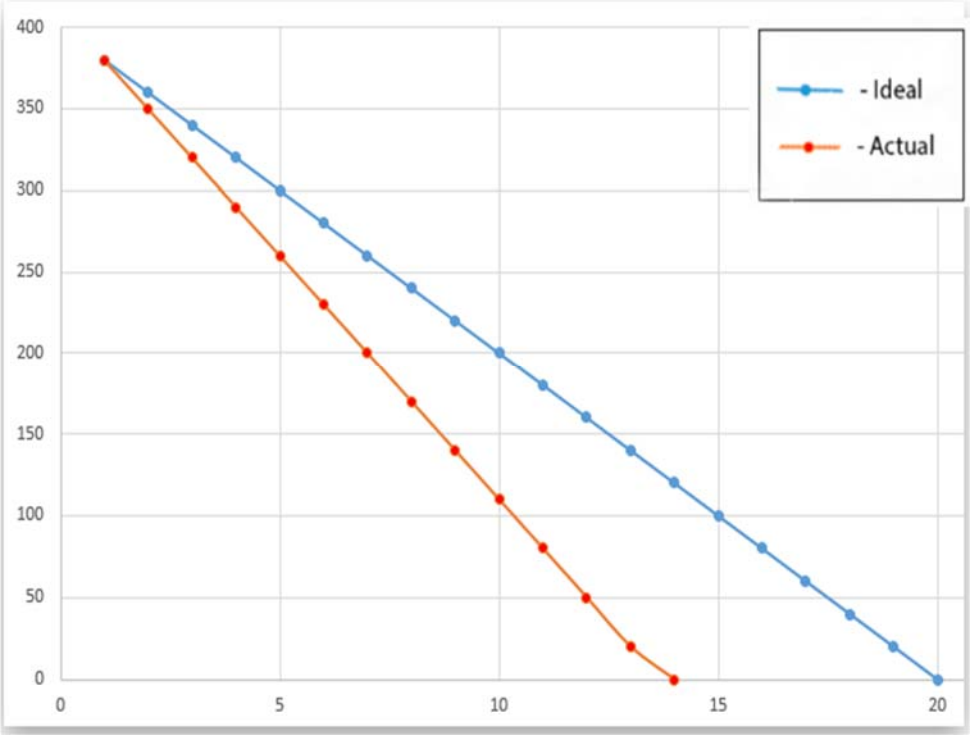


Рис. 4. Диаграмма сгорания спринта – завершен намного раньше срока [7]

Например, на диаграмме Рис. 4 отчетливо видно, что команда завершила работу по спринту раньше заявленного срока. Конечно, такой результат является позитивным для проекта в целом, однако же по нему можно судить и о ряде имеющихся проблем в команде [7]:

- оценка предстоящей работы была выполнена неправильно, команда сильно перестраховалась;
- в ходе выполнения спринта не были добавлены новые задачи; как правило, именно это и рекомендуется сделать проектному менеджеру в ситуации, изображенной на Рис. 4.

Ещё одним, более негативным вариантом развития событий является ситуация, когда команда сильно не успевает выполнять текущие задачи. Тогда диаграмма сжигания задач спринта может выглядеть примерно так, как показано на Рис. 5.

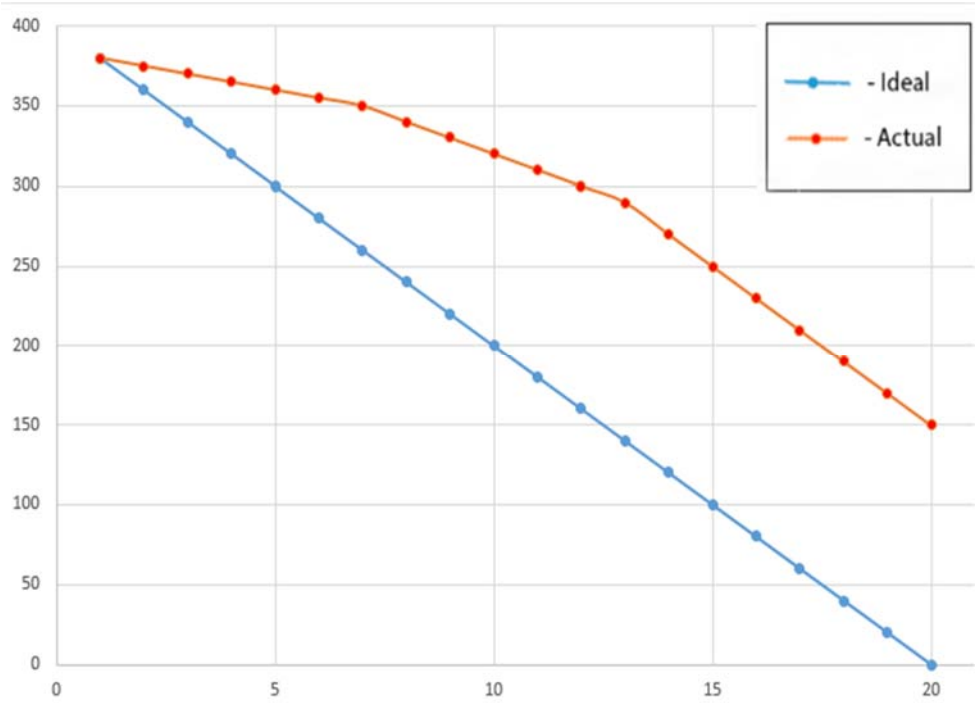


Рис. 5. Диаграмма сгорания спринта – сильное отставание в графике выполнения задач [7]

Данная диаграмма свидетельствует о недостаточной эффективности ежедневных собраний команды в рамках методологии Scrum. Возможно, здесь имело место постоянное добавление задач во время спринта, либо же часть задач была выполнена не полностью [7]. В любом случае менеджеру проекта следует уделить особое внимание ситуации, о которой свидетельствует график на Рис. 5.

Отметим, что любое отклонение вверх графика реального выполнения задач от запланированного графика выполнения задач спринта, согласно одному из постулатов методологии Scrum [5], уже является проблемой: её обязательно нужно обсуждать и решать в рамках проектных встреч.

4) *Диаграмма совокупного потока*

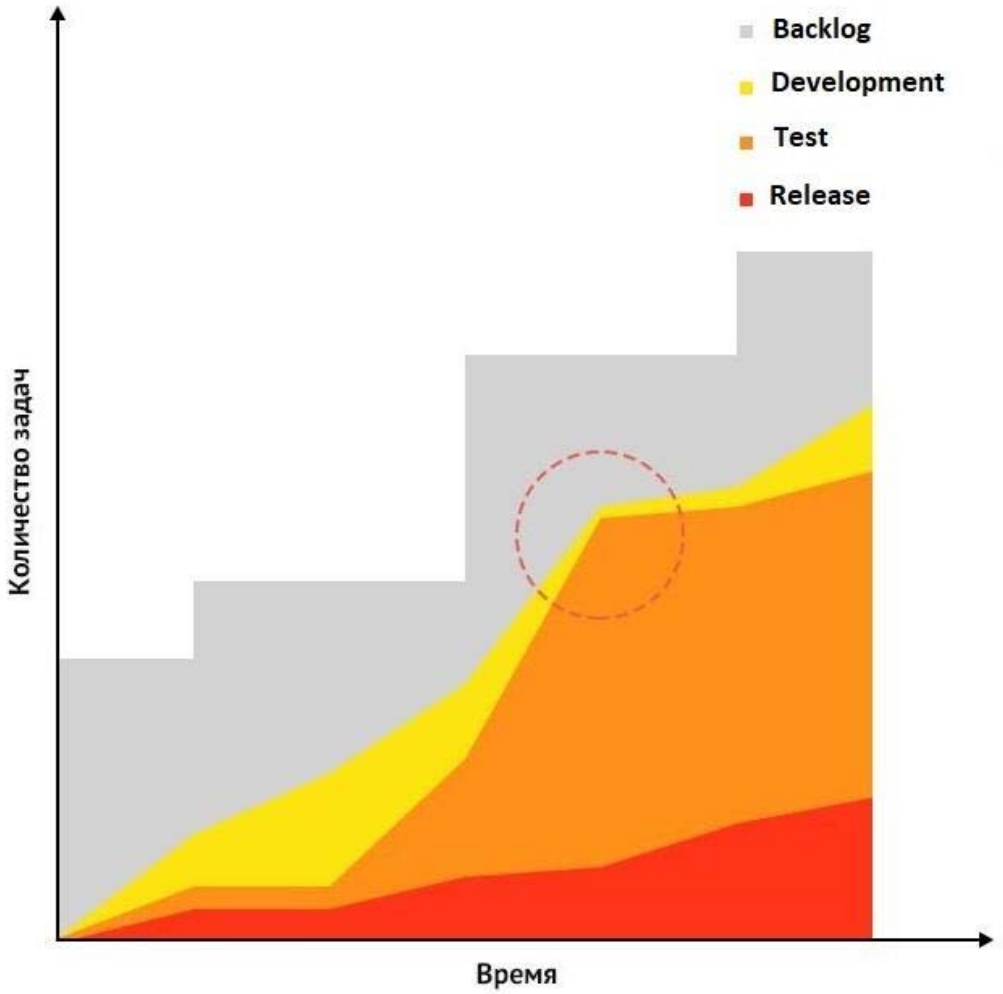


Рис. 6. Диаграмма совокупного потока [8]

На Рис. 6 приведен пример диаграммы совокупного потока. Она представляет собой график, визуализирующий количество задач с тем или иным статусом во времени [8]. Линии, иллюстрирующие разные статусы, либо приближаются друг к другу, либо отдаляются. Если одна линия начинает почти полностью перекрывать другую, то это говорит о том, что в работе над проектом образуются простои. Например, на данном графике изображена ситуация, когда разработчики не успевают обеспечить задачами отдел тестирования. В этом случае менеджеру необходимо повысить производительность команды разработки либо сократить количество новых задач – по ситуации.

ИСПОЛЬЗОВАНИЕ ПОКАЗАТЕЛЕЙ ЭФФЕКТИВНОСТИ В СИСТЕМАХ УПРАВЛЕНИЯ ПРОЕКТАМИ

Для работы над проектом менеджеры, как правило, используют различные системы управления с многопользовательским доступом для распределения текущих задач и отслеживания статуса их выполнения [9]. Такие системы представляют собой пользовательский интерфейс с различным набором инструментов, в том числе и широко применяемых в гибких методологиях разработки, как, например, Scrum- и Kanban-доски.

Как уже было показано выше, имея статистику задач по проекту, данные по срокам их исполнения, связанности задач друг с другом, данные об исполнителях и т. д., можно отразить приблизительную динамику выполнения проекта, позволяющую на основе предыдущих показателей предсказать, завершится ли текущий спринт в установленные для него сроки или часть задач останется незавершенной.

Некоторые системы, как, например, Atlassian Jira, предлагают такой функционал автоматического построения графиков [10]. Однако же начинающий проектный менеджер далеко не всегда способен понять, на что именно нужно обратить внимание на данный момент, исходя из имеющихся показателей. Поэтому было решено разработать программный инструмент, который сообщал бы начинающим менеджерам проектов о наиболее важных отклонениях, возникающих в ходе разработки программного продукта.

Принцип работы и архитектура приложения

Созданное приложение имеет клиент-серверную архитектуру, и все новые данные по мере их создания сразу же отправляются на сервер, где хранятся в удаленной базе данных. Между клиентом и сервером ведется синхронизация, поэтому если произошло какое-либо значимое для проекта событие, то клиент сразу же будет оповещен.

Само приложение представляет собой мобильный клиент для системы управления проектами. В нем реализованы многие стандартные функции, необходимые проектному менеджеру: создание новых задач, спринтов, проектов, их редактирование, мониторинг активности, модуль документации и некоторые другие. Кроме всего упомянутого, приложение имеет модуль статистики, показывающий графики с текущим состоянием проекта, а также модуль оповещений,

целью которого является уведомить проектного менеджера, когда состояние проекта начинает отклоняться от запланированного, и посоветовать ему предпринять какие-либо меры в зависимости от сложившейся ситуации.

Виды оповещений

Выше был представлен способ анализа состояния выполнения проекта на основе нескольких диаграмм. По данным для их построения и о сроках выполнения задач можно выделить ключевые моменты в отклонении промежуточных результатов проекта от изначально запланированных. Рассмотрим некоторые из них:

1) ***Недооценка времени выполнения критических задач***

Задачи с высоким для проекта приоритетом должны выполняться в первую очередь, т. к. именно от их реализации напрямую зависит главный функционал проекта. Поэтому особо важно следить за статусом выполнения таких задач и как можно быстрее реагировать на возникающие препятствия в их реализации.

Таким образом, если задача с высоким приоритетом, запланированная на текущий спринт, будет по каким-либо причинам не выполнена в запланированные сроки, приложение уведомит об этом проектного менеджера и посоветует ему рассмотреть эту проблему на ежедневной встрече. Механизм работы уведомления для этого случая представлен на блок-схеме Рис. 7, где **n** – количество задач в спринте, **taskList** – коллекция задач спринта.

2) ***Отставание команды от графика выполнения текущего спринта***

Распределение задач в команде может быть неравномерным – играют роль многие факторы, например, опыт члена команды. Одному разработчику можно делегировать сразу несколько задач на текущий спринт, а затем добавлять новые или, наоборот, передать задачу для выполнения от одного разработчика другому. Вполне возможна ситуация, что у одного члена команды ближе к концу спринта окажется задач больше, чем он рассчитывает выполнить. Тогда часть его задач нужно передать другим разработчикам. Если такая ситуация возникает сразу у всей команды разработки, то это свидетельствует о явной недооценке ситуации по спринту в целом.

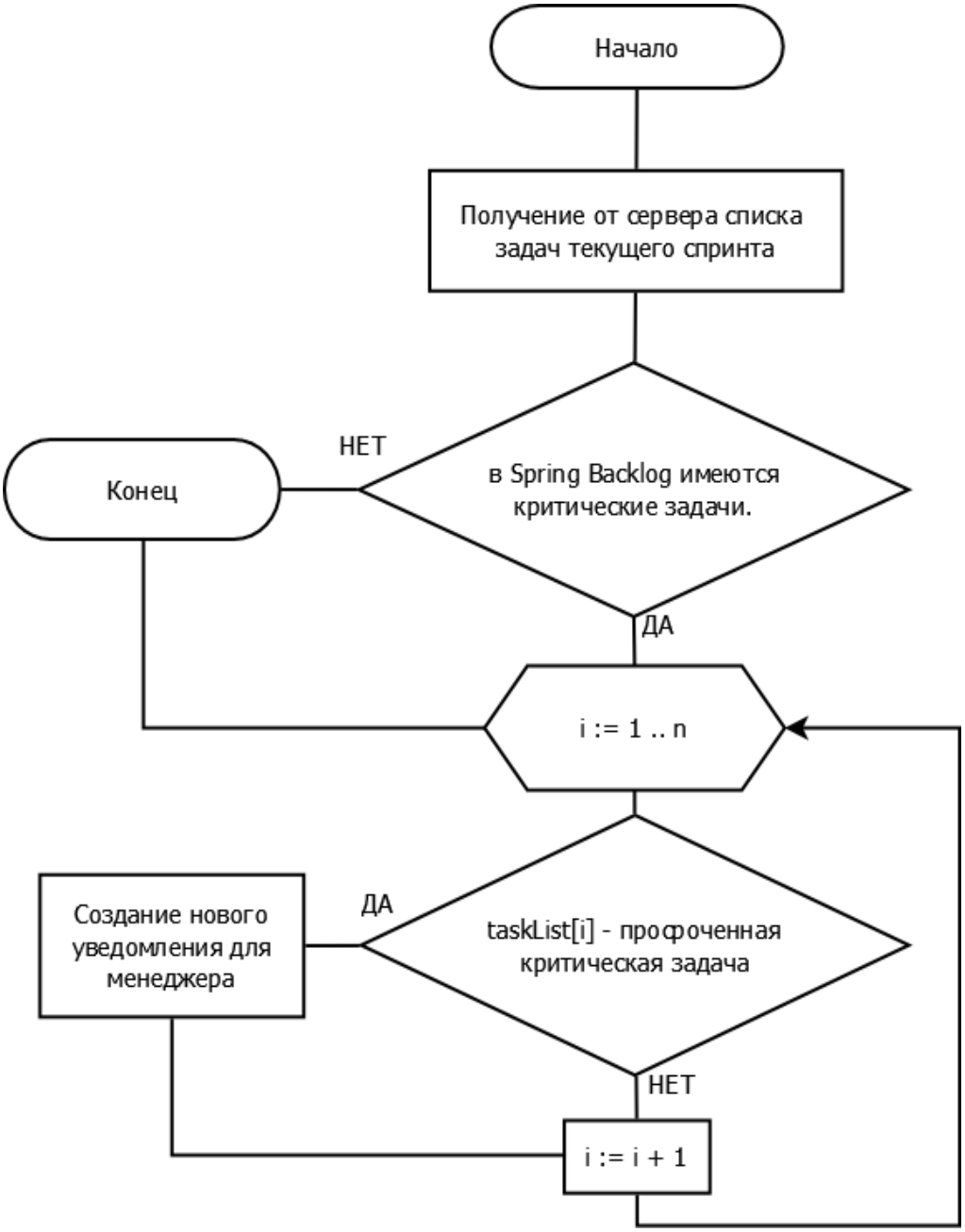


Рис. 7. Блок-схема создания уведомлений о невыполненных критических задачах

Таким образом, приложение уведомит менеджера в следующих случаях:

- если у члена команды сумма Story Point всех невыполненных задач окажется больше, чем осталось дней до завершения спринта;
- если больше, чем у 50% членов команды (в этом случае перераспределение всех задач между участниками уже маловероятно) сумма Story Point всех невыполненных задач окажется больше, чем осталось дней до завершения

спринта, приложение уведомит проектного менеджера о необходимости предпринять меры для ускорения процесса разработки, например, пересмотреть приоритет задач, чтобы успеть выполнить наиболее важные, увеличить ресурсы или производительность работы текущих исполнителей.

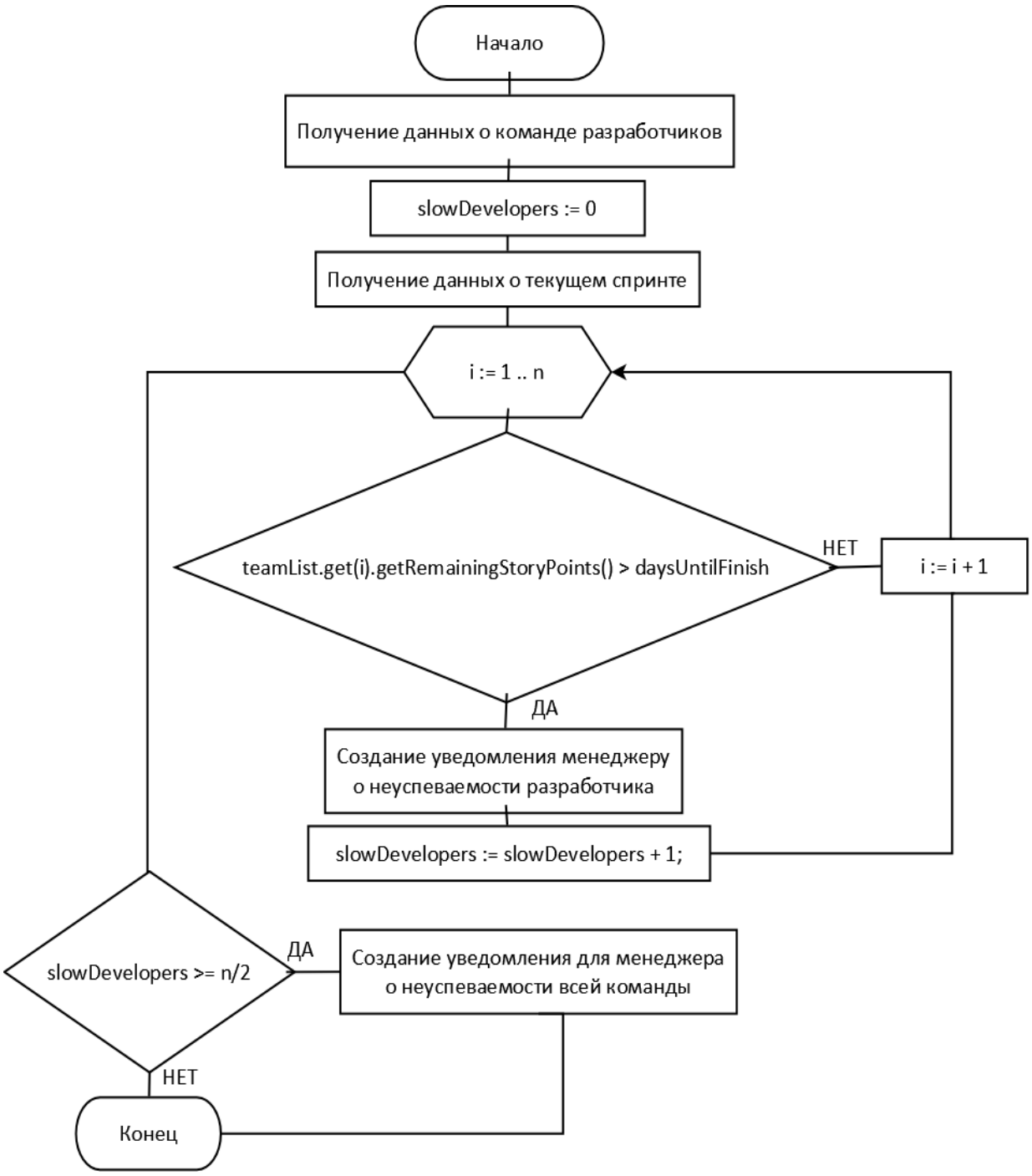


Рис. 8. Блок-схема создания уведомлений об отставании разработчиков от запланированного графика

Механизм создания уведомления для случаев отставания разработчиков от запланированного графика представлен на блок-схеме Рис. 8, где **n** – число разработчиков в команде, **teamList** – коллекция с данными о разработчиках, **getRemainingStoryPoints()** – сумма Story point всех оставшихся задач у конкретного разработчика, **slowDevelopers** – количество неуспевающих разработчиков, **daysUntilFinish** – число дней до конца текущего спринта.

3) ***Переоценка текущего спринта***

Если 70% всех задач, включая критические, будет выполнено к середине спринта или же все баги и User Story будут закрыты ещё до конца спринта, то это может свидетельствовать о переоценке командой поставленных задач. В этом случае приложение напомнит менеджеру о необходимости на ежедневной встрече расспросить о возможности добавления новых задач в Sprint Backlog. Блок-схема Рис. 9 демонстрирует принцип создания уведомлений для менеджера о возможности добавления новых задач.

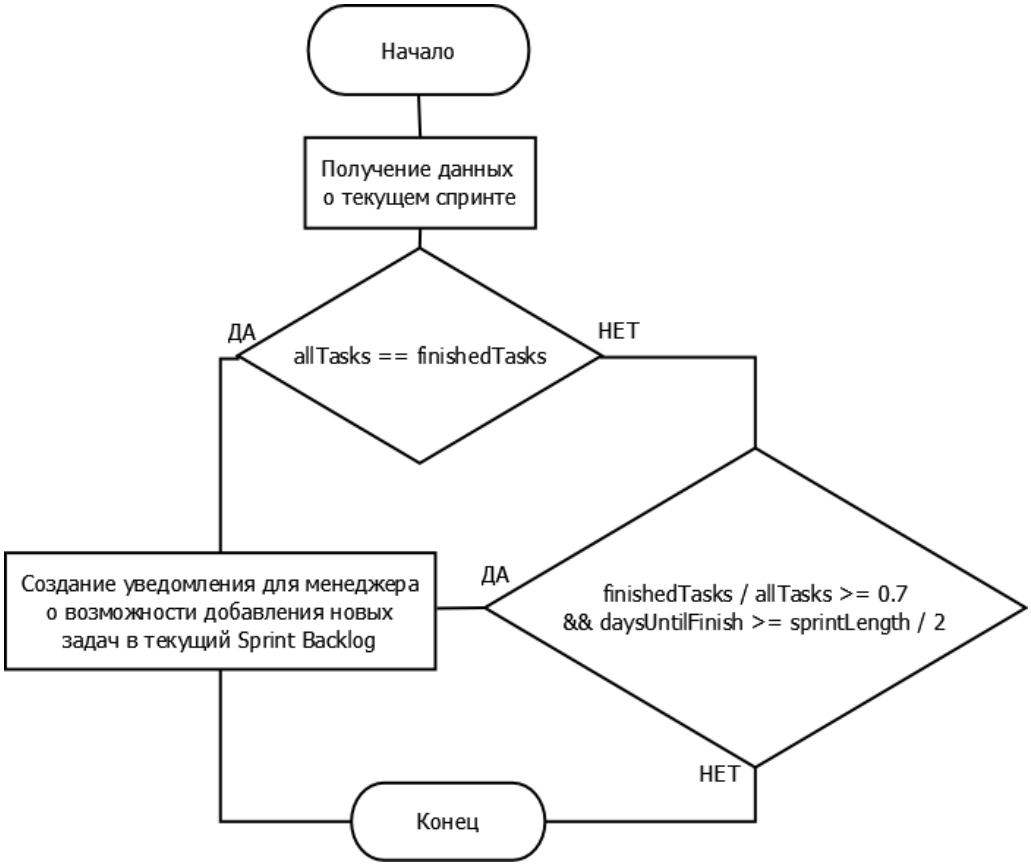


Рис. 9. Блок-схема создания уведомлений о возможности пополнения текущего Sprint Backlog

На данном рисунке **sprintLength** – количество дней в текущем спринте, **daysUntilFinish** – число дней до конца спринта, **allTasks** – число задач в текущем Sprint Backlog, **finishedTasks** – число выполненных задач текущего Sprint Backlog.

4) ***Изменение состава команды разработки***

Изменения в команде разработки во многих случаях могут сопровождаться резким падением производительности всей команды в целом, причем неважно, будут ли это уход члена команды или добавление новых участников. Один из таких случаев, когда добавление новых разработчиков снизило производительность команды в целом, был рассмотрен в статье ранее. К новичкам в проекте необходим индивидуальный подход. Перед менеджером в данном случае стоит задача посвятить их в особенности проекта, при этом сделав так, чтобы их появление не привело к сильному падению производительности работы всей команды.

Рис. 10. Блок-схема создания уведомлений для контроля новых членов команды разработки

- При появлении новых членов команды приложение покажет менеджеру уведомление о необходимости приставить к ним наставника, разбирающегося в проекте, а также перед каждым ежедневным митингом на протяжении первого спринта с новыми участниками у менеджера будет появляться уведомление о необходимости расспросить новичков о текущем прогрессе в понимании проекта и его технологий. Механизм создания уведомлений для данного случая представлен на блок-схеме Рис. 10, где **sprintsFinished** – количество завершенных спринтов.
- Если же участников в проекте становится меньше 3 или больше 9 (рекомендуемые количества членов команды разработки в рамках методологии Scrum [5]), то проектный менеджер получит уведомление о недостатке людей в команде (вследствие чего есть риск не успеть выполнить все задачи) либо о слишком большом числе разработчиков (вследствие чего могут возникнуть проблемы с коммуникацией между ними). Блок-схема для этой ситуации представлена на Рис. 11, где **developersCount** – количество разработчиков в команде.

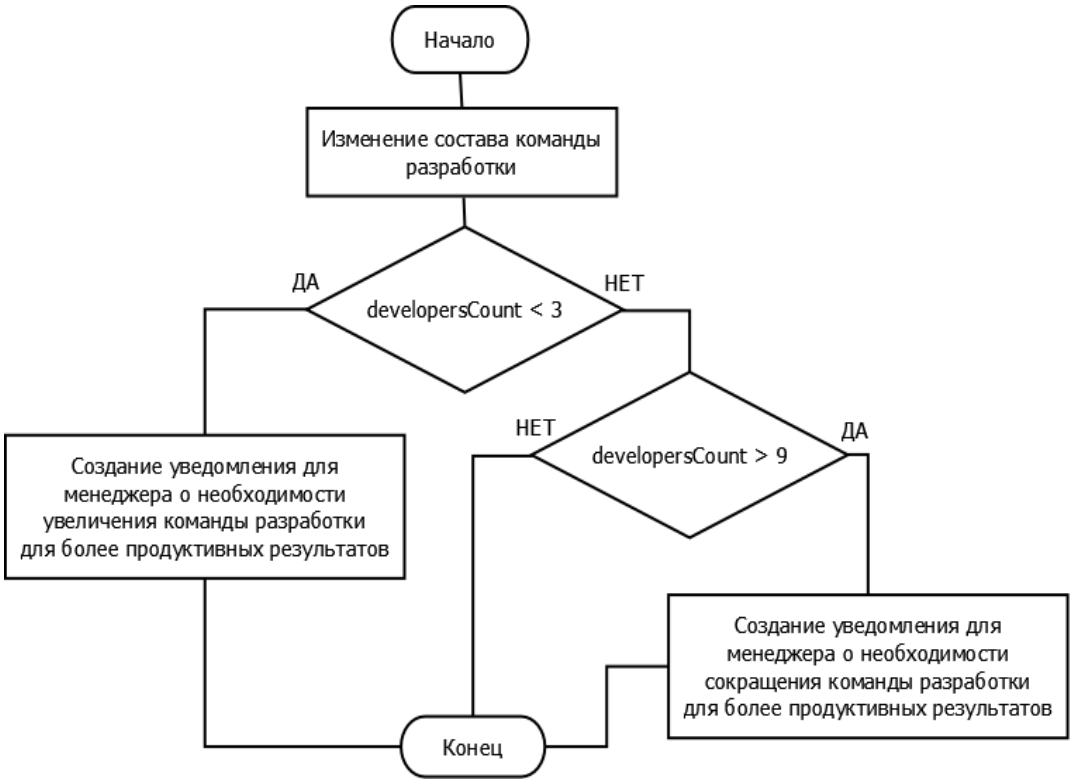


Рис. 11. Блок-схема, иллюстрирующая процесс создания уведомлений о неоптимальном количестве членов команды разработки

5) ***Простои в команде разработки***

При рассмотрении диаграммы совокупного потока в предыдущем разделе была отмечена ситуация простоя отдела разработки. В приложении это отслеживается путем сравнивания количества User Story и багов у разработчиков и тестировщиков за весь спринт (либо за весь процесс разработки продукта).

Таким образом, если по ходу спринта количество задач у подразделений начинает совпадать либо отличаться на незначительное число (в приложении за такое число было взято количество человек в команде разработки), то менеджер получит уведомление о простое в команде разработчиков и о том, что необходимо повысить их производительность.

Блок-схема создания уведомлений о простое разработчиков представлена на Рис. 12, где **developersCount** – количество разработчиков в команде, **developersTasks** – общее количество решаемых задач, **testTasks** – общее количество задач для тестировщиков.

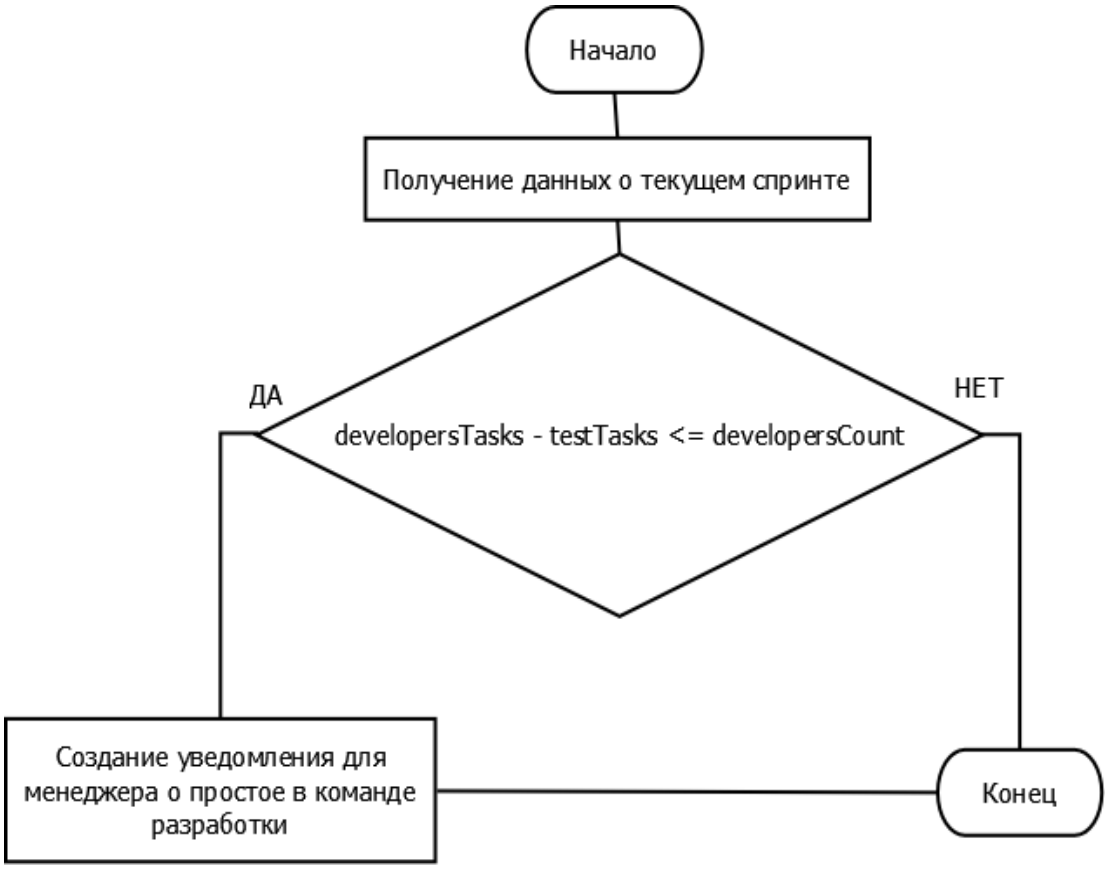


Рис. 12. Блок-схема создания уведомлений о простое команды разработчиков

6) ***Переполнение Product Backlog***

В случае изменения заказчиком требований в Product Backlog могут быть добавлены новые задачи либо же могут быть изменены требования к уже существующим. Такая ситуация рассматривалась в предыдущем разделе и оказалась весьма критичной для проекта: потребовалось ещё 2 незапланированных спринта для завершения разработки.

Таким образом, в этих случаях приложение сразу будет уведомлять менеджера:

- о необходимости изучить новые требования и задачи, как только они будут добавлены;
- если добавление новых требований повлекло за собой существенное увеличение Product Backlog более чем на 20%, то менеджеру в такой ситуации будет предложено сделать полную переоценку первоочередного функционала, увеличить ресурсы либо повысить производительность работы текущих исполнителей.

На Рис. 13 изображена блок-схема создания уведомлений в приложении о добавлении новых задач в Product Backlog, где **beginBacklogCount –** количество задач в Product Backlog к началу очередного спринта, **currentBacklogCount –** количество задач в Product Backlog на момент синхронизации приложения с удаленным сервером.

ЗАКЛЮЧЕНИЕ

В статье проанализированы лишь некоторые данные, на основании которых менеджер проектов может судить об эффективности процесса разработки программного продукта. Приведены примеры визуализации этих данных на графиках, а также разобран механизм уведомления в мобильном приложении, работающий на основе анализа рассмотренных проектных данных. Такое приложение в будущем при активном его внедрении и расширении способно помочь начинающим менеджерам в процессе их обучения.

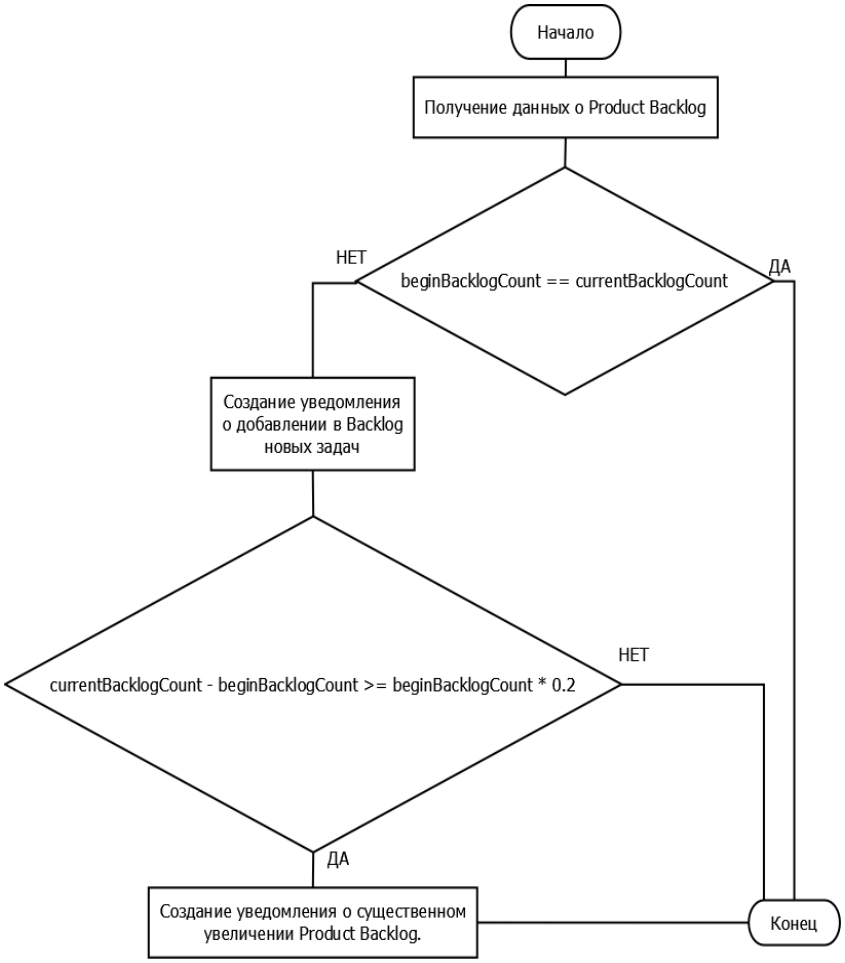


Рис. 13. Блок-схема создания уведомлений о добавлении в Product Backlog новых задач

СПИСОК ЛИТЕРАТУРЫ

1. Success Rates Rise. Transforming the high cost of low performance // Обзор 9-го глобального исследования в области управления проектами. Project Management Institute, 2017. URL: https://www.pmi.org/-/media/pmi/documents/public/pdf/learning/thought-leadership/pulse/pulse-of-the-profession-2017.pdf

2. *Clinton J. Woodward, Andrew Cain, Shannon Pace, Allan Jones and Joost Funke Kupper.* Helping Students Track Learning Progress Using Burn Down Charts // 2013 IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE), 26–29 August 2013, Bali Dynasty Resort, Kuta, Indonesia.

3. *Paul O. Gaddis.* The Project Manager // Harvard Business Review (May–June 1959).

4. *Mahnic V., Zabkar N.* Measuring Progress of Scrum-based Software Projects // Elektronika i elektrotechnika, ISSN 1392-1215. 2012. V. 18, No 8.

5. *Ken Schwaber and Jeff Sutherland.* The Scrum Guide™ // The Definitive Guide to Scrum. URL: http://www.scrumguides.org/ docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf#zoom=100

6. *Victor Szalvay.* An Introduction to Agile Software Development // Copyright © 2004 Danube Technologies, Inc. URL: http://www.danube.com/docs/Intro_to_ Agile.pdf

7. Система управления проектами “Scrum time”. URL: https://ru.scrum-time.com/infobase/burndown-chart.php

8. Scrum – студия “Сибирикс”. URL: https://blog.sibirix.ru/2014/11/27/ graphs/

9. *Alok Mishra, Deepti Mishra.* Software Project Management Tools: A Brief Comparative View // ACM SIGSOFT Software Engineering Notes, 2013. V. 38, No 3, May.

10. Система управления проектами “Jira Software”. URL: https://ru.atlassian. com/software/jira/features.org

PROJECT MANAGER’S COMPETENCY BUILDING USING MOBILE LEARNING TECHNOLOGIES

M. M. Abyzov[1], I. S. Shakhova[2]

Higher School of Information Technologies and Intelligent Systems at Kazan Federal University

[1]mishabuzov@gmail.com, [2]is@it.kfu.ru

Abstract

An overview of the methods for measuring progress of software development within flexible SCRUM methodology is given, as well as a description of a software tool development that monitors current status of project by time characteristics. The tool based on the characteristics can notificate project manager about events he have to pay attention in current project situation and help him to achieve effective results.

Keywords: *project management, project manager, training of project managers, mobile application, SCRUM, User Story, Story Point, Sprint, Sprint Backlog, Agile reporting, Mobile Learning.*

REFERENCES

1. Success Rates Rise. Transforming the high cost of low performance // Обзор 9-го глобального исследования в области управления проектами. Project Management Institute, 2017. URL: https://www.pmi.org/-/media/pmi/documents/public/pdf/learning/thought-leadership/pulse/pulse-of-the-profession-2017.pdf

2. *Clinton J. Woodward, Andrew Cain, Shannon Pace, Allan Jones and Joost Funke Kupper.* Helping Students Track Learning Progress Using Burn Down Charts // 2013 IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE), 26–29 August 2013, Bali Dynasty Resort, Kuta, Indonesia.

3. *Paul O. Gaddis.* The Project Manager // Harvard Business Review (May–June 1959).

4. *Mahnic V., Zabkar N.* Measuring Progress of Scrum-based Software Projects // Elektronika i elektrotechnika, ISSN 1392-1215. 2012. V. 18, No 8.

5. *Ken Schwaber and Jeff Sutherland.* The Scrum Guide™ // The Definitive Guide to Scrum. URL: http://www.scrumguides.org/ docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf#zoom=100

6. *Victor Szalvay.* An Introduction to Agile Software Development // Copyright © 2004 Danube Technologies, Inc. URL: http://www.danube.com/docs/Intro_to_Agile.pdf

7. Project management system “Scrum time”. URL: https://ru.scrum-time.com/infobase/burndown-chart.php

8. Scrum – studiya “Sibiriks”. URL: https://blog.sibirix.ru/2014/11/27/ graphs/

9. *Alok Mishra, Deepti Mishra.* Software Project Management Tools: A Brief Comparative View // ACM SIGSOFT Software Engineering Notes, 2013. V. 38, No 3, May.

10. Issue tracking system “Jira Software”. URL: https://ru.atlassian.com /software/jira/features.org

СВЕДЕНИЯ ОБ АВТОРАХ

АБЫЗОВ Михаил Михайлович – студент Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета, андроид-разработчик.

Mikhail Mikhailovich ABYZOV – student of the Higher School of Information Technologies and Intelligent Systems at Kazan Federal University, Android developer.

email: mishabuzov@gmail.com

ШАХОВА Ирина Сергеевна – ассистент кафедры программной инженерии Высшей школы информационных технологий и интеллектуальных систем Казанского федерального университета. Сфера научных интересов – цифровые образовательные системы, индивидуализация образования, мобильное обучение.

Irina Sergeevna SHAKHOVA – teacher of the Higher School of Information Technologies and Intelligent Systems at Kazan Federal University. Research interests include digital educational systems, individualization of education, mobile learning.

email: is@it.kfu.ru

Материал поступил в редакцию 31 мая 2018 года

УДК 004.93+004.5

СИНХРОНИЗАЦИЯ СЕССИЙ ДОПОЛНЕННОЙ РЕАЛЬНОСТИ В НАТИВНЫХ МОБИЛЬНЫХ ПРИЛОЖЕНИЯХ

Д. А. Евдокименко[1], Р. Г. Ханов[2], И. С. Шахова[3]

Высшая школа информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета

[1]evdodima@gmail.com, [2]rinat@khanov.com, [3]is@it.kfu.ru

Аннотация

Представлена реализация алгоритма синхронизации сессий дополненной реальности в мобильных iOS-приложениях, позволяющего создавать такие сессии с несколькими участниками для их совместного взаимодействия с одними и теми же виртуальными объектами.

***Ключевые слова**: дополненная реальность, augmented reality, AR, ARKit, сессия дополненной реальности, синхронизация сессий дополненной реальности*

ВВЕДЕНИЕ

Дополненная реальность – это среда с прямым или косвенным дополнением физического мира цифровыми данными в режиме реального времени при помощи компьютерных устройств [1]. На мобильных устройствах технологии дополненной реальности могут быть использованы, например, для отображения маячков или иконок на карте в реальном времени при наведении камеры на местность.

В iOS 11 компанией Apple представлен новый фреймворк ARKit, позволяющий применять технологию дополненной реальности (AR, augmented reality) в iOS-приложениях. ARKit, используя информацию со встроенных датчиков устройства, таких, как акселерометр, гироскоп, магнитометр, а также анализируя изображение, полученное с камеры, позволяет определять положение устройства в пространстве относительно других объектов – реальных или виртуальных. Помимо отображения виртуальных объектов данный фреймворк способен распознавать в реальном мире горизонтальные поверхности (начиная с версии 1.5,

и вертикальные). Это позволяет, например, размещать виртуальные объекты на распознанных поверхностях реального мира, благодаря чему виртуальные объекты остаются зафиксированными в конкретных точках на поверхностях реального мира и выглядят более естественными. Он также позволяет, например, проводить измерения расстояний или площадей на данных поверхностях.

Стандартные методы в ARKit предоставляют разработчику информацию о расположении устройства относительно начальной точки отсчета и распознанных горизонтальных и вертикальных плоскостей в реальном мире. В текущей версии ARKit все эти данные хранятся внутри одной сессии дополненной реальности (AR-сессии) на устройстве пользователя, а система координат на каждом устройстве находится в разных точках, в которых сессия была начата. Данные ограничения не позволяют делиться данными сессии с другими устройствами, а также корректно размещать виртуальные объекты из сессий других устройств, так как системы координат расположены по-разному. Таким образом, в текущей версии ARKit не представляется возможным создавать сессии дополненной реальности с несколькими участниками для их совместного взаимодействия с одними и теми же объектами дополненной реальности. Отсутствие возможности создания сессий дополненной реальности с несколькими участниками сильно ограничивает области, в которых технологии дополненной реальности могли бы применяться на мобильных устройствах. В таких сферах, как образование, медицина, проектирование, а также развлечения, дополненная реальность может быть применена гораздо эффективнее, если в процесс взаимодействия с виртуальными объектами будет вовлечено несколько пользователей [2].

В статье представлены реализация алгоритма передачи и синхронизации данных различных AR-сессий между несколькими устройствами и обеспечение совместной работы этих устройств внутри одной сессии дополненной реальности. Для реализации данного алгоритма используются нативные для iOS инструменты и технологии: язык программирования Swift 4.0, фреймворк ARKit 1.5 и системный фреймворк SceneKit для 3D графики, а также системный фреймворк Multipeer Connectivity для организации передачи данных между устройствами.

ПЕРЕДАЧА ДАННЫХ

Объединение устройств в сеть для передачи данных

Multipeer Connectivity Framework – это нативный для системы iOS фреймворк, который позволяет передавать данные, потоки и файлы между iOS-устройствами, находящимися рядом. Для этого используются WiFi сети, прямое WiFi соединение или Bluetooth [3]. Процесс работы фреймворка состоит из двух стадий: стадия обнаружения, во время которой происходит поиск устройств, с которыми будет осуществляться передача данных, и стадия взаимодействия (сессии), во время которой и происходит передача данных.

Для объединения устройств в сеть нужно реализовать метод advertiser(didRecieveInvationFromPeer:), который вызывается у делегата MCNearbyServiceAdvertiser, когда устройство получает приглашение присоединиться к сети. Внутрь этого метода передается функция invitationHandler, которая принимает в качестве параметра булевскую переменную, являющуюся ответом на приглашение (принимается приглашение или нет). Также нужно реализовать метод browser(foundPeer:), который вызывается у делегата MCNearbyServiceBrowser, когда найдено новое устройство. В данном методе нужно пригласить устройство в сеть. Для этого вызывается метод invitePeer. Для начала работы фреймворка нужно вызвать методы startAdvertising и startBrowsing у соответствующих объектов. Далее новые устройства будут находить друг друга и объединяться в сеть автоматически.

Передача данных о расположении устройств

В ARKit для отображения виртуальных объектов и взаимодействия с ними используется фреймворк SceneKit. Информацию о расположении устройства и его ориентации на сцене можно получить из объекта pointOfView, который находится на сцене и обновляется фреймворком ARKit, – этот объект отображает, где сейчас на сцене находится пользователь (наблюдатель).

Расположение устройства хранится в виде трех координат (X,Y,Z) с помощью структуры SCNVector3 [4]. Ориентация устройства (его углы поворота) также хранится в виде SCNVector3, где X, Y и Z – углы поворота вокруг соответствующих осей. Положение этих осей координат показано на Рис. 1, вектор (X, Y, Z).

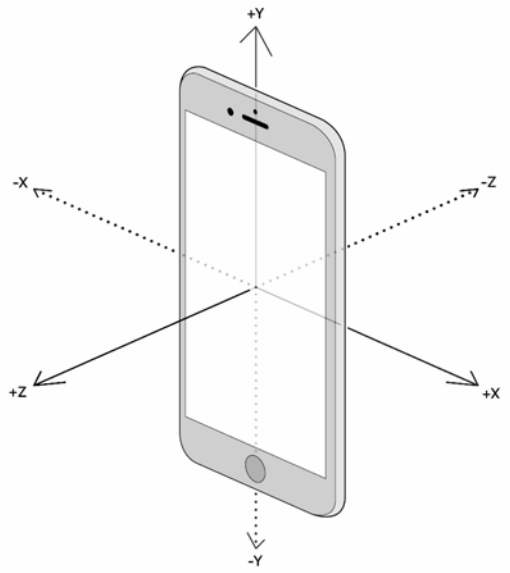


Рис. 1. Расположение осей координат, определяющих поворот устройства

Таким образом, чтобы однозначно определять положения устройств в пространстве, достаточно передавать два вектора: вектор расположения и вектор поворота.

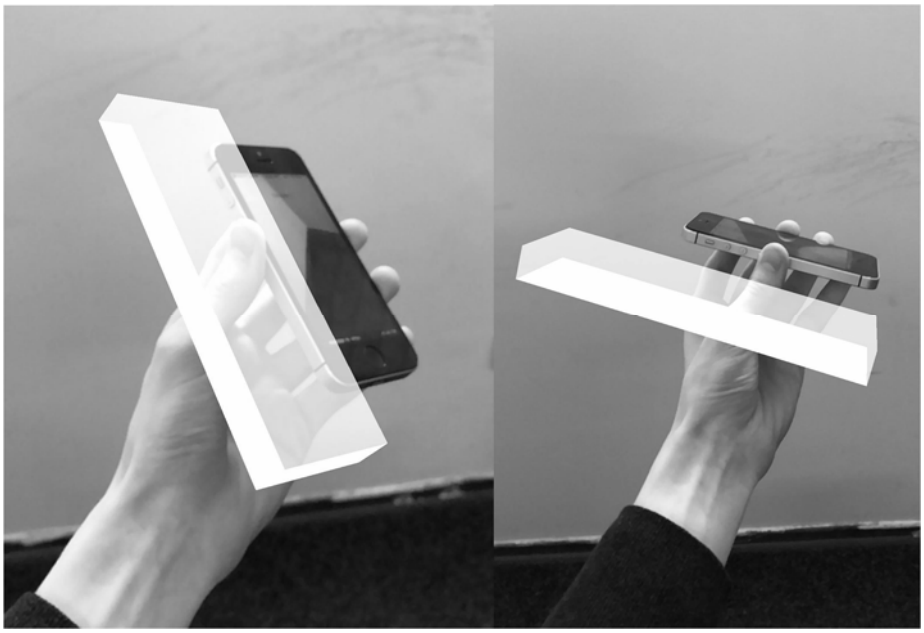


Рис. 2. Пример визуализации положения устройства

Так как данные о положении устройств обновляются постоянно, удобнее и быстрее всего для их передачи использовать потоки данных. В Multipeer Connectivity используются стандартные для языка Swift потоки ввода/вывода Input/OutputStream [5]. Для удобства положения всех устройств на сцене визуализируются в виде параллелепипедов с соотношением сторон, похожим на со-

отношение сторон телефона. В тот момент, когда из потока считываются данные о положении устройства, параллелепипед перемещается в точку с полученными координатами и поворачивается соответственно полученному вектору поворота. На Рис. 2 показаны примеры визуализации положений устройств с помощью данного параллелепипеда.

Передача данных об объектах на сцене

Чтобы обеспечить полную синхронизацию сцен на разных устройствах, необходимо, помимо положений на сцене самих устройств, передавать информацию обо всех объектах сцены и их обновлениях. В SceneKit основным объектом сцены является узел (SCNNode). Основными параметрами узла являются его расположение, ориентация и геометрия. Геометрия в свою очередь имеет определенные вид (сфера, плоскость, куб и т. д.), размер, а также материал [6]. В данном проекте в качестве объектов сцены, которые будут синхронизироваться между устройствами, выступают распознанные с помощью ARKit горизонтальные плоскости, значит, вид и материал геометрии для всех объектов будут одинаковыми.

ARKit предоставляет информацию о распознаваемых плоскостях и позволяет их визуализировать. Для этого существуют три метода делегата ARSCNViewDelegate:

- renderer(didAdd node:) – вызывается, когда была распознана новая плоскость;
- renderer(didUpdate node:) – вызывается, когда для существующей плоскости были обновлены данные (например, размер);
- renderer(didRemove node:) – вызывается, когда ранее распознанная плоскость удаляется (например, если произошло ее слияние с более крупной плоскостью).

Таким образом, можно выделить три вида событий, с помощью которых происходит обновление сцены: add, update и remove. Все эти события так или иначе изменяют состояние сцены и ее объектов.

Удобным и быстрым способом обмена информацией о состоянии сцены является передача по сети данных событий. Такое решение не нагружает сеть постоянной передачей однотипной информации, а также является знакомым и

понятным для разработчиков, так как представляет собой реализацию шаблона проектирования Observer [7].

В Multipeer Connectivity имеется возможность передавать по сети объекты типа Data – это значит, что любой объект, реализующий интерфейс Codable, может быть закодирован с помощью Encoder и передан по сети другим устройствам. В данном случае таким объектом будет объект класса Event, содержащий информацию о передаваемом событии. На Рис. 3 приведена блок-схема алгоритма обработки данных о событии, получаемых устройством.

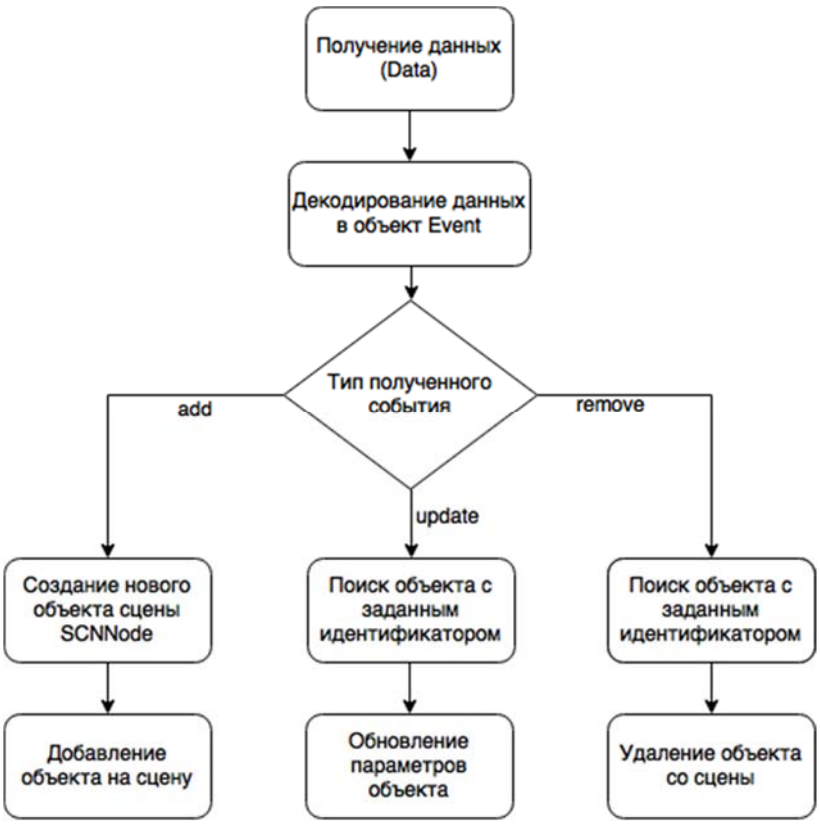


Рис. 3. Блок-схема алгоритма обработки событий

В ARKit удобно визуализировать распознанные плоскости с помощью узлов с геометрией SCNPlane, которая представляет собой плоскость, размер которой можно ограничить по ширине и высоте [8]. В качестве материала плоскости используется сетка из гексагонов (см. Рис. 4).

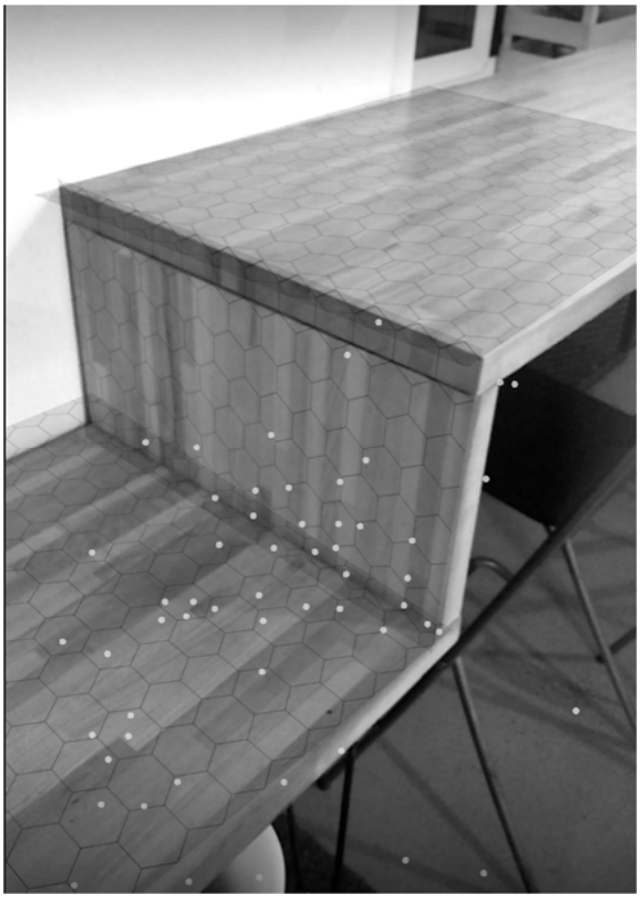


Рис. 4. Визуализация распознанных плоскостей

СИНХРОНИЗАЦИЯ СИСТЕМ КООРДИНАТ

Распознавание и локализация изображений

Начиная с версии 1.5, в ARKit появилась возможность распознавать и локализовывать (определять расположение и ориентацию в пространстве) заданные изображения [9]. Информация о распознанных изображениях, как и информация о распознанных плоскостях, будет передаваться приложению с помощью методов ARSCNViewDelegate. Пример работы этой технологии представлен на Рис. 5.

Синхронизация систем координат двух устройств

Данная технология может быть применена не только для распознавания статичных изображений в реальном мире, но и для таких изображений, расположение которых изменяется. Это позволяет, например, расположить заданную картинку на экране другого мобильного телефона или планшета, что позволит,

распознавая данное изображение, получать информацию о расположении устройства в пространстве и его ориентации. Чтобы такое изображение было корректно распознано на экране устройства, нужно правильно задать его физические размеры и, соответственно, отобразить его на экране таким образом, чтобы оно соответствовало заданным размерам. Также изображение должно соответствовать следующим условиям [10]:

Рис. 5. Распознавание и локализация изображений

- не должно иметь центральную симметрию (например, изображение круглой мишени будет плохо распознаваться, так как не ясно, где у него верх, а где низ);
- не должно иметь однородных цветовых областей;
- должно иметь высокую контрастность;
- должно быть хорошо текстурировано.

Когда устройства объединяются в сеть для передачи данных, их системы координаты не синхронизированы: начало координат сессии дополненной реальности в каждом устройстве находится в том месте, где был телефон, когда началась сессия, а поворот системы координат – такой же, как у устройства в момент начала сессии. Поэтому недостаточно просто передавать информацию о

расположении объектов и устройств на сцене, так как в разных устройствах одни и те же объекты сцены будут расположены в разных местах относительно объектов реального мира. Поэтому, прежде чем начинать взаимодействие пользователей с дополненной реальностью, необходимо синхронизировать системы координат их устройств.

Для синхронизации систем координат сессий дополненной реальности подходит технология, описанная ранее: она позволяет определить реальное расположение устройства, с которым происходит синхронизация, и сопоставить его реальные координаты с теми, которые присылает данное устройство. Далее, зная реальные координаты устройства и те координаты, которые получены от данного устройства, нужно переместить систему координат устройства таким образом, чтобы эти координаты совпали.

Таким образом, для синхронизации систем координат двух устройств подходит следующий алгоритм:

- объединение устройств в сеть для передачи данных;
- начало передачи данных о расположении и ориентации устройств между собой;
- определение главного устройства, с которого будет производиться распознавание изображения, и второстепенного, на экране которого будет показано изображение для распознавания;
- распознавание главным устройством изображения, показанного на экране второстепенного;
- получение на главном устройстве координат распознанного изображения и, соответственно второго устройства;
- нахождение различия в углах поворота систем координат.
- поворот системы координаты главного устройства на вычисленный угол;
- нахождение вектора разности положений систем координат устройств;
- перемещение начала системы координат главного устройства на вычисленный вектор.

В результате работы данного алгоритма начало системы координат главного устройства поворачивается согласно повороту системы координат второстепенного устройства, а начало системы координат помещается в ту же точку, где находится начало системы координат второстепенного устройства.

Обобщенный алгоритм синхронизации

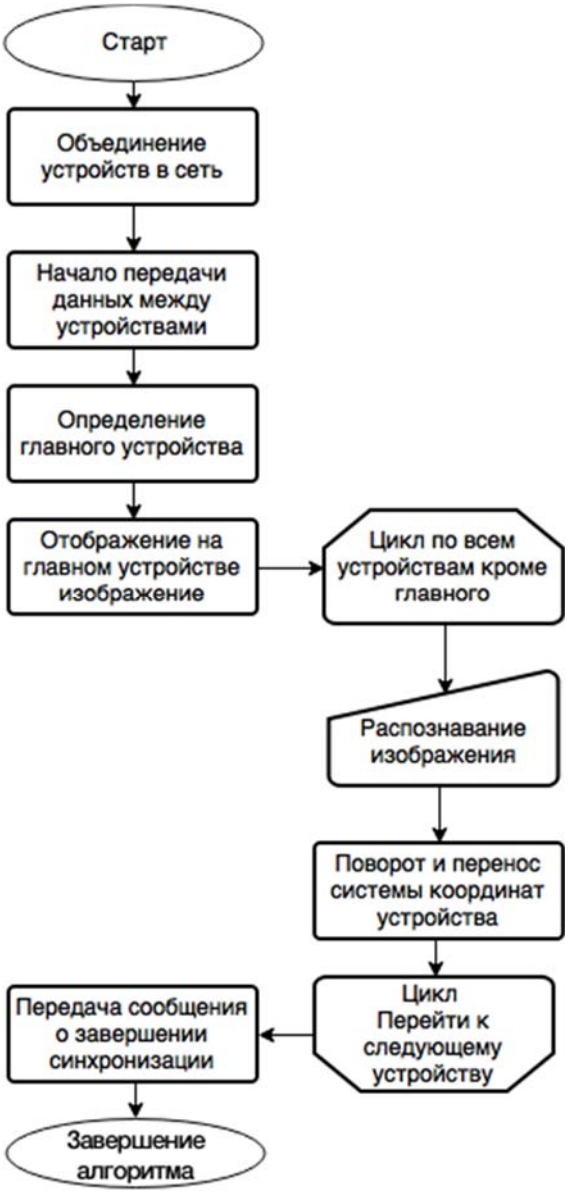


Рис. 6. Блок-схема обобщенного алгоритма синхронизации

Описанный алгоритм подходит только для синхронизации двух устройств, однако его можно обобщить на случай любого числа устройств. Для этого необходимо выделить главное устройство, к системе координат которого будут приведены системы координат остальных устройств. Затем, применив алгоритм

синхронизации из предыдущего параграфа для каждого второстепенного устройства, путем распознавания изображения, показанного на экране главного устройства, нужно переместить систему координат этого устройства, чтобы она соответствовала системе координат главного. На Рис. 6 приведена блок-схема работы обобщенной версии алгоритма.

ЗАКЛЮЧЕНИЕ

Построенный алгоритм выполняет следующие функции:

- объединение устройств в сеть для передачи данных;
- синхронизация информации о положении всех устройств на сцене;
- синхронизация информации о распознаваемых плоскостях.

Таким образом, данный алгоритм позволяет находящимся рядом устройствам работать внутри одной AR-сессии. На Рис. 7 приведен пример работы алгоритма для двух устройств.

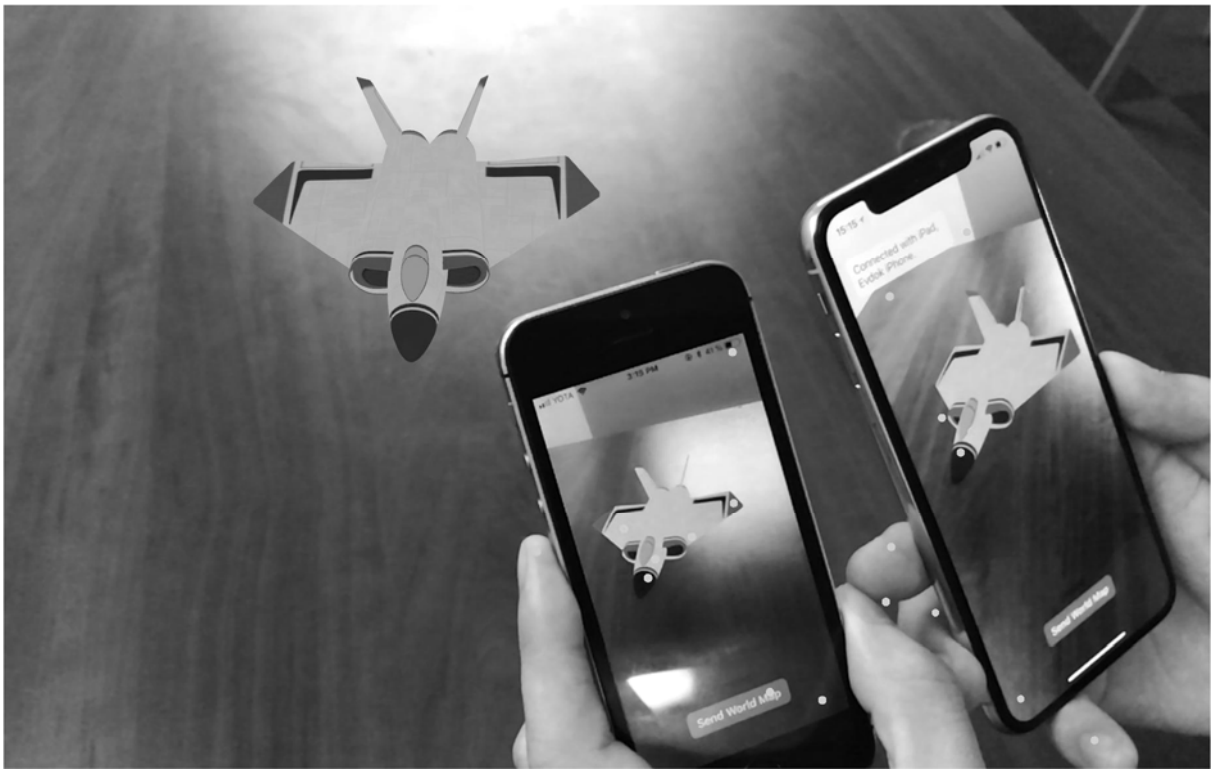


Рис. 7. Пример работы алгоритма

Описанный алгоритм имеет широкий спектр применения, так как позволяет реализовать многопользовательские AR-приложения – игры, приложения для совместного просмотра 3D-контента и взаимодействия с ним, навигационные

приложение и другие. Описанный алгоритм может быть выделен в отдельную библиотеку, что позволит разработчикам, подключая данную библиотеку с помощью CocoaPods [11], без проблем объединять устройства для совместной работы в дополненной реальности.

СПИСОК ЛИТЕРАТУРЫ

1. История VR/AR. URL: https://rb.ru/story/vsyo-o-vr-ar/

2. Towards Massively Multi-User Augmented Reality on Handheld Devices. URL: https://pdfs.semanticscholar.org/05be/d977601c84ae581a9a8b2 054ce484b342 e10.pdf

3. Multipeer Connectivity Framework iOS 7. URL: https://habrahabr.ru/company/touchinstinct/blog/198814/

4. SceneKit. URL: https://www.raywenderlich.com/83748/ beginning-scene -kit-tutorial

5. Класс Stream – документация Apple. URL: https://developer. apple.com/documentation/foundation/stream

6. Структура сцены в SceneKit. URL: https://www.invasivecode.com/ weblog/scenekit-tutorial-part-1/?doing_wp_cron=1513691954. 507189998908996582031 25

7. Шаблон проектирования Observer. URL: http://design-pattern.ru/ patterns/observer.html

8. Свойства геометрии SCNPlane. URL: http://spec-zone.ru/RU/OSX/ documentation/SceneKit/Reference/SCNPlane_Class/index.html

9. Распознавание изображений в ARKit. URL: https://developer. apple.com/documentation/arkit/recognizing_images_in_an_ar_experience

10. Требования к распознаваемому изображению. URL: https://developer. apple.com/documentation/arkit/recognizing_images_in_an_ar_experience

11. Менеджер зависимостей Cocoapods. URL: https://cocoapods.org

AUGMENTED REALITY SESSIONS SYNCHRONIZATION ALGORITHM IN MOBILE APPLICATIONS

D. A. Evdokimenko[1], R. G. Khanov[2], I. S. Shakhova[3]

Higher School of Information Technologies and Intelligent Systems at Kazan (Volga region) Federal University

[1]evdodima@gmail.com, [2]rinat@khanov.com, [3]is@it.kfu.ru

Abstract

Implementation of augmented reality session synchronization algorithm in mobile iOS applications. Algorithm allows to create augmented reality sessions with several participants for their joint interaction with the same virtual objects.

Keywords: *augmented reality, AR, ARKit, augmented reality session, augmented reality session synchronization*

REFERENCES

1. History of VR/AR. URL: https://rb.ru/story/vsyo-o-vr-ar/
2. Towards Massively Multi-User Augmented Reality on Handheld Devices. URL: https://pdfs.semanticscholar.org/05be/d977601c84ae581a9a8b2054ce484b342e10.pdf
3. Multipeer Connectivity Framework iOS 7. URL: https://habrahabr.ru/company/touchinstinct/blog/198814/
4. SceneKit. URL: https://www.raywenderlich.com/83748/beginning-scene-kit-tutorial
5. Class Stream – *Apple Documentation.* URL: https://developer. apple.com/ documentation/foundation/stream
6. SceneKit scene structure. URL: https://www.invasivecode.com/weblog/scenekit-tutorial-part-1/?doing_wp_cron=1513691954. 5071899890899658203125
7. Observer design pattern. URL: http://design-pattern.ru/ patterns/observer.html
8. SCNPlane geometry properties. URL: http://spec-zone.ru/RU/OSX/ documentation/SceneKit/Reference/SCNPlane_Class/index.html
9. Image recognition in ARKit. URL: https://developer. apple.com/documentation/arkit/recognizing_images_in_an_ar_experience

10. Requirements for the recognized image. URL: https://developer.apple.com/documentation/arkit/recognizing_images_in_an_ar_experience

11. Cocoapods dependencies manager. URL: https://cocoapods.org

СВЕДЕНИЯ ОБ АВТОРАХ

ХАНОВ Ринат Гафурович – студент Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета, направление «Прикладная информатика».

Rinat Gafurovich KHANOV, student of the Higher School of Information Technologies and Intelligent Systems of Kazan Federal University.

email: rinat@khanov.com

ЕВДОКИМЕНКО Дмитрий Андреевич – студент Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета, направление «Прикладная информатика».

Dmitriy Andreevich EVDOKIMENKO, student of the Higher School of Information Technologies and Intelligent Systems of Kazan Federal University.

email: evdodima@gmail.com

ШАХОВА Ирина Сергеевна – ассистент кафедры программной инженерии Высшей школы информационных технологий и интеллектуальных систем Казанского федерального университета. Сфера научных интересов - цифровые образовательные системы, индивидуализация образования, мобильное обучение.

Irina Sergeevna SHAKHOVA – teacher of the Higher School of Information Technologies and Intelligent Systems at Kazan Federal University. Research interests include digital educational systems, individualization of education, mobile learning.

email: is@it.kfu.ru

Материал поступил в редакцию 28 мая 2018 года

УДК 378.1+004.891+519.226.3

ОБРАЗОВАТЕЛЬНАЯ АНАЛИТИКА И АДАПТИВНОЕ ОБУЧЕНИЕ С ИСПОЛЬЗОВАНИЕМ МОДЕЛИ СТУДЕНТА В ИНТЕЛЛЕКТУАЛЬНЫХ ОБУЧАЮЩИХ СИСТЕМАХ

М. В. Каяшев[1], Д. Ю. Макаров[2], А. А. Марченко[3]

[1-3] *Высшая школа информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета*

[1]mishauni13@gmail.com, [2]den.mkr@gmail.com, [3]anton.a.marchenko@gmail.com

Аннотация

Для поддержки адаптивного обучения и образовательной аналитики в интеллектуальных обучающих системах необходимо собирать и обрабатывать данные об успеваемости студентов и их индивидуальных характеристиках. Это можно реализовать с помощью модели студента. Анализ подходов к моделированию студента показал оптимальным применение нескольких типов моделей, исходя из требований, составленных для разрабатываемой обучающей системы. Были выбраны и объединены в одну модель три подхода: оверлейный, сеть Байеса, моделирование ошибочных знаний. Использование оверлейной модели позволяет строить индивидуальные траектории обучения студентов. Сети Байеса реализуют компетентностный подход в обучении. Модель ошибок отслеживает ошибочные знания студентов и помогает им исправить их на ранних стадиях. Модель студента, объединяющая в себе данные подходы, является подходящей для реализации персонализированного обучения, позволяет преподавателю отслеживать успеваемость студентов по различным характеристикам, а также дает возможность легко представить в системе карту тем, знаний, компетентности студентов в различных областях в виде графа, что является удобным и понятным представлением.

Ключевые слова: *интеллектуальная обучающая система, модель студента, компетенция, адаптивное обучение, образовательная аналитика, оверлейная модель, байесовская сеть, доменная модель*

ВВЕДЕНИЕ

В настоящее время информационные технологии все больше внедряются в образовательный процесс, создаются различные обучающие системы, целью которых является повышение эффективности и качества процесса обучения, а также поддержки адаптивности, т. е. персонализированного подхода, который уже стал трендом в образовании. При адаптивном подходе система может предоставлять материал в виде, удобном для студентов, исходя из их индивидуальных особенностей, и при постоянном отслеживании характеристик обучаемого и поведения определять траекторию его дальнейшего движения в рамках курса или учебного плана в целом. Кроме персонификации процесса обучения важным является предоставление показателей успеваемости обучаемого в удобном виде для преподавателя и самого студента, а также отчетности о каждом шаге студента для того, чтобы данный процесс был всегда виден, понятен и мог контролироваться преподавателем.

В рамках разрабатываемой интеллектуальной обучающей системы необходимо осуществить поддержку названных подходов, которые будут построены на основе модели студента, выбору и проектированию которой нужно уделить достаточное количество времени, так как модель студента является одной из главных частей обучающей системы.

В следующих частях статьи будут выделены основные требования, необходимые для поддержки аналитики и адаптивности в обучающей системе, и будет предложена подходящая коллаборация нескольких моделей студента с их описанием.

ТРЕБОВАНИЯ К МОДЕЛИ СТУДЕНТА

На данный момент существуют обучающие системы, направленные на персонализацию обучения с поддержкой образовательной аналитики. В каждой из них заложена своя модель студента, имеющая как преимущества, так и недостатки [1, 2]. Чтобы определить, какая модель обучаемого наиболее подходит для разрабатываемой системы, необходимо составить требования к ней. Основные выделенные требования приведены ниже.

Модель студента должна:

- собирать детальную информацию о знаниях обучаемых, выявлять качество усвоенного материала по каждой теме;
- учитывать связи между отдельными темами, заданиями;
- предоставлять возможность построения наглядных графиков процесса обучения и результатов студента на основе информации, содержащейся в модели;
- анализировать успеваемость обучающегося и допущенные ошибки в том или ином задании;
- уметь предлагать персональные и актуальные для обучаемого задания, исходя из данных его модели (уровень знаний, навыков);
- содержать информацию о степени сформированности компетенций (умений) студента;
- уметь перестраивать (изменять) план обучения студента.

Согласно классификации моделей студента, предложенной Г. А. Атановым [3], нами проанализировано 13 типов моделей и 74 конкретных представителя в существующих обучающих системах. Было замечено, что ни один отдельный тип модели не может в полной мере обеспечить выполнение всех требований, поэтому необходимо провести объединение нескольких подходов моделирования обучаемого. С точки зрения реализации адаптивного обучения в системе и поддержки персонализированного подхода к студенту необходимы использование удобного представления структуры курса и связности тем, поддержки компетентностного подхода, а также хранение подробной информации об оценках и студенте в целом. В результате анализа различных моделей с учетом обозначенных требований были объединены три подхода: оверлейный, моделирования ошибочных знаний и на основе байесовских сетей. Далее рассмотрим каждый из подходов.

ОВЕРЛЕЙНЫЙ ПОДХОД

Основная идея данного подхода к моделированию студента – это то, что модель обучаемого представляет собой некое подмножество основной модели знаний (часть полных знаний) [3]. Так как модель знаний студента имитирует доменную модель знаний (модель предметной области), получаем не только

связанные единицы знаний (темы), как в модели предметной области, но и значение, которое показывает уровень знаний по каждой теме. Это может быть полезно, например, для рекомендательной подсистемы, чтобы предлагать студенту задачи по смежным темам с целью устранения пробелов в знаниях по текущей изучаемой теме.

Данный подход хорошо зарекомендовал себя в интеллектуальных обучающих системах при обучении понятиям, но оказался недостаточно эффективным при обучении умениям и выработке компетенций [4]. Вся информация при таком подходе представляется в виде графа, что очень удобно передает структуру всего курса, отображение всех связанных тем, подтем, тестов, заданий курса для студента. Благодаря такому представлению легко создавать образовательные траектории студента.

БАЙЕСОВСКИЕ СЕТИ

Байесовская сеть, как и оверлейная модель, реализуется в виде графа, в котором каждая вершина представляет *n*-значную переменную, дуги обозначают существование зависимостей между переменными, а сила этих зависимостей количественно выражается в виде условных вероятностей, сопоставленных каждой из переменных [5].

Байесовская сеть дает возможность внедрить компетентностный подход. Компетенция здесь рассматривается как узел, который не имеет родителей, но имеет множество потомков, с помощью которых вычисляются показатели знаний (уровень освоения данной компетенции) [6]. Еще одно несомненное преимущество применения данного подхода – это возможность вывода информации об успеваемости студента внутри курсов. Преподаватель может поставить некую цель своим студентам, например, чтобы результат за некоторый промежуток времени соответствовал определенной части всех компетенций. Тогда, выполняя различные задания и проходя тесты, студент и преподаватель в реальном времени видят, какие результаты смог достичь студент.

МОДЕЛЬ ОШИБОК

Подход моделирования ошибочных знаний относится к направлению имитационных моделей, в которых модель студента может быть использована как хранилище ошибок, так как возможно определение того, что студент не понял в

необходимой мере [3]. В нашем случае данный подход предполагает хранение информации о полученных знаниях (корректных), о неполученных знаниях и об ошибочных знаниях (когда студент совершает ошибку).

Такой подход может решить проблему позднего вмешательства системы или преподавателя в учебный процесс студента, так как происходит корректировка траектории обучения студента на ранних стадиях.

ОБЪЕДИНЕНИЕ ПОДХОДОВ

После объединения трех подходов получившаяся модель представляет собой обработку данных из модели предметной области и получение структуры компетенций и результатов студента, основываясь на имеющихся у него оценках. Данное поведение является оверлейным.

В основе структуры модели лежит ориентированный граф. Все связи (отношения) между вершинами однонаправленные и идут сверху-вниз, что дает возможность удобного построения причинно-следственных связей для элементов (узлов). При этом различные узлы могут иметь общих родителей, а различные родители – общих потомков.

В Байесовской сети используется условная вероятность для каждого узла графа. Это означает, что каждый потомок узла непосредственно влияет на вероятность родительского узла, вероятность которого в свою очередь влияет далее на его родителя, и так до самого первого предка. В модели студента данные вероятности очень удобно использовать для вычисления оценки освоения студентом некоторого материала (в виде процентного соотношения).

В каждый момент времени каждый узел сети имеет всю информацию, необходимую для вычисления условной вероятности соответствующей переменной, а при получении оценки семантических элементов заданий какого-либо узла этот узел отправляет сообщения об изменении соседним узлам. Далее эта информация передается по цепочке, и в каждом узле пересчитываются условные вероятности. Таким образом, поступающая информация о значениях наблюдаемых узлов распространяется по байесовской сети, вызывая обновление вероятностей для каждого из оставшихся узлов. Подсчет вероятности производится по формуле [7]:

$$P\left(X_{1,\dots,}X_n\right) = \prod_{i=1}^{N} P(X_i|parents(X_i)), \quad (1)$$

где parents(X_i) – родители узла X_i.

Процесс оценивания ответов студентов происходит по следующим параметрам: «known», «not known», которые принимают значения от 0 до 100%. Таким образом, модель хранит более точный результат для студента.

Благодаря применению подхода выявления ошибочных знаний мы добавили еще одну переменную «not learnt». С ее помощью, используя данные из модели студента, можно определить, какое количество знаний студент недополучил, то есть насколько он ошибся. Применение данного подхода позволяет системе, студенту и преподавателю отслеживать, где студент совершил ошибки и какую учебную информацию ему следует предоставить.

РЕАЛИЗАЦИЯ

Рассмотренная модель студента была реализована в виде прототипа со следующими компонентами: графовая база данных, реляционная база данных Postgresql, программный сервис, приложение для визуализации Angular. В качестве графовой базы данных использовалась Neo4j [8], что подходит для работы с графовой структурой данных Байесовской сети. Каждая вершина в ней представляет собой компетенцию, тему, подтему, тест или вопрос. Связь между вершинами осуществляется с помощью отношения Relation, которая содержит в себе свойство «вес» для каждого отношения. Это позволяет хранить сложность и важность каждой единицы знаний для каждой вышестоящей вершины отдельно. Все вершины были созданы с помощью запросов на языке Cypher [9]. К каждой из них были добавлены свойства и настроены отношения (связи) между ними, а также прописана важность для каждого отношения. Весь процесс работы системы показан на Рис. 1. Программный сервис на Java Spring осуществляет связь с Neo4j и Postgresql базами данных, для работы с которыми созданы специальные сущности: Student, Node, Course. Чтобы предоставить данные внешним приложениям, также создан API интерфейс, который передает данные о студенте в виде JSON с помощью библиотеки Jackson. Приложение Angular осуществляет связь с программным сервисом через API интерфейс. После получения данных в виде JSON они передаются специализированным библиотекам: Sigma JS [10], Chart JS [11], D3JS [12]. Данные библиотеки предоставляют студенту или преподавателю информацию в виде графов, деревьев, таблиц и различных

графиков, например, результаты студента, структура курса, освоенные компетенции, отношение компетенций и тем (Рис. 2).

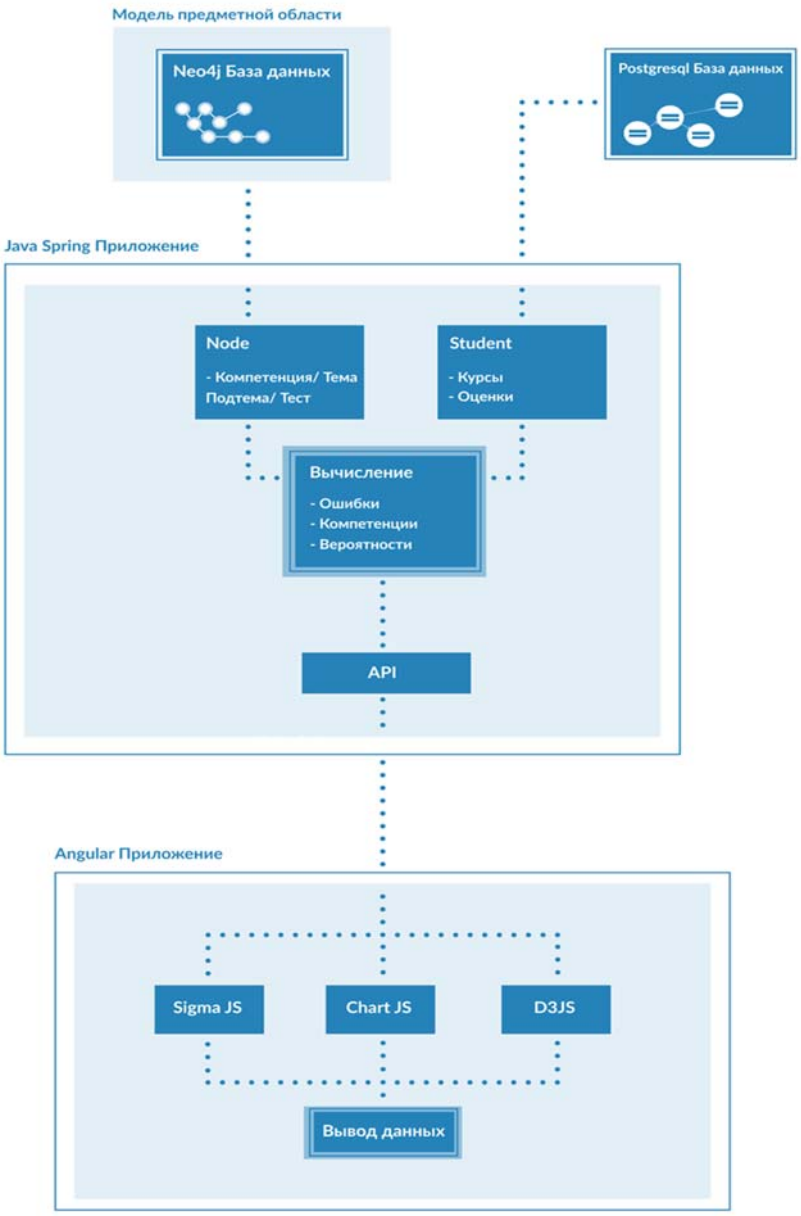


Рис. 1. Процесс работы системы

Рис. 2. Отношения компетенций и тем

ЗАКЛЮЧЕНИЕ

Результат проведенного анализа существующих моделей показал, что объединение нескольких подходов моделирования студента – наиболее подходящее решение для реализации персонализированного подхода, поддержки адаптивного обучения и образовательной аналитики в интеллектуальной обучающей системе.

Объединение Байесовской сети с моделью ошибок позволяет отображать студентов, которые имеют различные проблемы, например, не успевающих по учебному плану или совершающих слишком много ошибок по определенной теме. Оверлейная модель с сетями Байеса дает возможность легко представить в системе карту тем и знаний в виде графа, что является довольно удобным и понятным представлением для студента и преподавателя, а также внедрять оценивание сложности задания.

Рассмотренная модель студента, которая объединяет в себе несколько подходов, учитывает различные характеристики студента для реализации персонализированного обучения и является информативной для преподавателя с точки зрения отображаемых результатов студента, чего нельзя добиться при применении только одного подхода моделирования студента. Данная модель будет использоваться в разработке интеллектуальной обучающей системы, а также может быть применена в других адаптивных системах, где требуется отслеживать успеваемость студента, его знания, навыки и компетенции, а также в системах, направленных на персонализированный подход в обучении.

СПИСОК ЛИТЕРАТУРЫ

1. *Konstantina Chrysafiadi, Maria Virvou*. Student modeling approaches: A literature review for the last decade // Expert Systems with Applications. 2013. No 40. P. 4715–4729.

2. *Буль Е.Е.* Обзор моделей студента для компьютерных систем обучения // Educational Technology & Society. 2003. № 6 (4). С. 245–250.

3. *Атанов Г.А., Пустынникова И.Н.* Обучение и искусственный интеллект, или основы современной дидактики высшей школы. Донецк: Изд-во ДОУ, 2002. 504 с.

4. *Клюкин В.Э.* Web-ориентированные интеллектуальные обучающие системы на основе нечеткого деятельностного подхода в обучении // Наука и образование. 2012. № 11. С. 450–461.

5. *Cristina Conati.* Bayesian Student Modeling // Advances in Intelligent Tutoring Systems. 2010. No 308. P. 281–289.

6. *Хлопотов М.В.* Применение байесовской сети при построении моделей для оценки уровня сформированности компетенций // Интернет-журнал «Науковедение». 2014. № 5 (24). С. 1–28.

7. *Дзюбан Ю., Болдак Л.* Анализ алгоритмов вывода в Байесовских сетях доверия. В кн.: Високопродуктивні обчислення. Междунар. конф., Киев, 8–10 октября 2012. С. 167–169.

8. The Neo4j Graph Platform. https://neo4j.com/product/

9. Cypher. URL: http://neo4j.com/docs/developer-manual/current/cypher/

10. Sigma.js. URL: http://sigmajs.org

11. Chart.js. URL: https://www.chartjs.org
12. D3.js – Data-Driven Documents. URL: https://d3js.org

EDUCATIONAL ANALYTICS AND ADAPTIVE TRAINING USING STUDENT MODEL IN THE INTELLECTUAL LEARNING SYSTEMS

M. V. Kayashev[1], D. Yu. Makarov[2], A. A. Marchenko[3]

[1-3] *Higher School of Information Technologies and Intelligent Systems at Kazan (Volga region) Federal University*

[1] mishauni13@gmail.com, [2] den.mkr@gmail.com, [3] anton.a.marchenko@gmail.com

Abstract

For support of adaptive training and educational analytics in the intellectual learning systems, it is necessary to collect, process data on progress of the student and his various individual characteristics. It can be realized by means of the student model. The analysis of approaches to modeling of the student has shown that application of several types of models is an optimal solution, considering requirements to the learning system. Three approaches were chosen and united into one model: overlay, Bayesian network, error model. Use of overlay model allows to build individual trajectories of student training. Bayesian networks realize competence-based approach in training. The model of mistakes keeps track of wrong knowledge of the student and helps the student to correct them at early stages. The student model uniting in itself these approaches is more suitable for realization of the personalized training, allows to keep track of progress of the student according to various characteristics and also gives the chance to easily submit the card of subjects, knowledge, competence of the student of various areas in the form of the count that is quite convenient and clear representation.

Keywords: *intellectual learning system, student model, competence, adaptive training, educational analytics, overlay model, Bayesian network, domain model*

REFERENCES

1. *Konstantina Chrysafiadi, Maria Virvou*. Student modeling approaches: A literature review for the last decade // Expert Systems with Applications. 2013. No 40. P. 4715–4729.

2. *Bul' E.E.* Obzor modelej studenta dlja komp'juternyh sistem obuchenija // Educational Technology & Society. 2003. № 6 (4). S. 245–250.

3. *Atanov G.A., Pustynnikova I.N.* Obuchenie i iskusstvennyj intellekt, ili osnovy sovremennoj didaktiki vysshej shkoly. Doneck: Izd-vo DOU, 2002. 504 s.

4. *Kljukin V.Je.* Web-orientirovannye intellektual'nye obuchajushhie sistemy na osnove nechetkogo dejatel'nostnogo podhoda v obuchenii // Nauka i obrazovanie. 2012. № 11, S. 450–461.

5. *Cristina Conati.* Bayesian Student Modeling // Advances in Intelligent Tutoring Systems. 2010. No 308. P. 281–289.

6. *Hlopotov M.V.* Primenenie bajesovskoj seti pri postroenii modelej dlja ocenki urovnja sformirovannosti kompetencij // Internet-zhurnal «NAUKOVEDENIE». 2014. № 5 (24). S. 1–28.

7. *Dzjuban Ju., Boldak L.* Analiz algoritmov vyvoda v Bajesovskih setjah doverija. V kn.: "Visokoproduktivni obchislennja" mezhdunar. konf., Kiev, 8–10 oktjabrja 2012. S. 167–169.

8. The Neo4j Graph Platform. https://neo4j.com/product/

9. Cypher. URL: http://neo4j.com/docs/developer-manual/current/cypher/

10. Sigma.js. URL: http://sigmajs.org

11. Chart.js. URL: https://www.chartjs.org

12. D3.js – Data-Driven Documents. URL: https://d3js.org

СВЕДЕНИЯ ОБ АВТОРАХ

КАЯШЕВ Михаил Владиславович – магистр Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета.

Mihail Vladislavovich KAYASHEV – Master of the Higher School of Information Technologies and Intelligent Systems at Kazan (Volga region) Federal University.

e-mail: mishauni13@gmail.com

МАКАРОВ Денис Юрьевич – бакалавр Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета.

Denis Yur'evich MAKAROV – Bachelor of the Higher School of Information Technologies and Intelligent Systems at Kazan (Volga region) Federal University.

e-mail: den.mkr@gmail.com

МАРЧЕНКО Антон Александрович – преподаватель Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета, ассистент кафедры программной инженерии.

Anton Aleksandrovich MARCHENKO – lecturer of the Higher School of Information Technologies and Intelligent Systems at Kazan (Volga region) Federal University, department assistant of software engineering.

e-mail: anton.a.marchenko@gmail.com

Материал поступил в редакцию 1 июня 2018 года

УДК 004.414.3

ФОРМИРОВАНИЕ АКАДЕМИЧЕСКИХ ГРУПП И ПРОЕКТНЫХ КОМАНД НА ОСНОВЕ СБОРА ДАННЫХ ОБ ОБУЧАЮЩИХСЯ

Н. А. Коргутлова[1], С. Ю. Басаргина[2], М. М. Абрамский[3], М. А. Солнцев[4], Т. С. Бузукина[5]

[1-5]*Высшая школа информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета*

[1]korgutlova97@gmail.com, [2]basargina.svetlana@gmail.com, [3]ma@it.kfu.ru, [4]mrt.solncev@gmail.com, [5]taya.buzukina@gmail.com

Аннотация

Обсуждены вопросы использования данных об обучающихся, представленных в электронном виде, в задачах генерации распределений обучающихся по академическим группам, элективам и проектным командам. Проиллюстрировано применение алгоритмов машинного обучения для этих задач. Показана возможность использования данных, собранных из социальных сетей.

***Ключевые слова**: личностный портрет студента, кластеризация, распределение по компетенциям, анализ социальных сетей*

ВВЕДЕНИЕ

Согласно Указу Президента РФ от 07.05.2018 г. №204 [1], основными задачами в сфере образования в ближайшие годы являются: «внедрение на уровнях основного общего и среднего общего образования [...] образовательных технологий, обеспечивающих освоение обучающимися базовых навыков и умений, повышение их [...] вовлеченности в образовательный процесс, а также модернизация профессионального образования, в том числе посредством внедрения адаптивных, практико-ориентированных и гибких образовательных программ».

Одним из приоритетных направлений модернизации образования является формирование новых моделей организации учебной деятельности, учитывающей вариативность и индивидуальность высшего образования. Главным достоинством индивидуального обучения [2] является то, что оно позволяет пол-

ностью адаптировать содержание, методы и темпы учебной деятельности студента к его особенностям, следить за каждым его действием при решении конкретных задач.

Одним из видов реализации индивидуального обучения является обучение в сотрудничестве или малых группах, однако сразу же встает вопрос о методе распределения по группам, которое должно быть не случайным, а основано на личностном портрете студента – его характеристиках, компетенциях, интересах и др. Выполнить такое распределения вручную достаточно сложно, при этом все равно могут быть не учтены способности студентов, так как этот процесс является достаточно трудоемким и длительным. Ручное распределение большого объема данных может привести к случайным ошибкам из-за человеческого фактора. Вследствие этого возникает необходимость автоматизации данного процесса. Таким образом, в рамках настоящей работы поставлен следующий вопрос: как быстро и функционально можно распределить студентов по группам, имея их личностные портреты?

Статья построена следующим образом. В разделе 1 приведены ключевые характеристики личностного портрета студента. В разделе 2 раскрыта возможность уточнения цифрового портрета с помощью анализа данных социальных сетей. В разделах 3 и 4 представлены разработанные методы и инструменты распределения студентов по академическим группам и курсам по выбору. В разделе 5 рассмотрена возможность построения проектных команд.

1. ЛИЧНОСТНЫЙ ПОРТРЕТ ОБУЧАЮЩЕГОСЯ

В течение всего времени обучения в образовательном учреждении происходит сбор большого объема информации о каждом обучающемся. К этой информации относятся:

- данные анкеты – ФИО, дата и место рождения, семейное положение, статус семьи и др.;
- физические особенности, группа здоровья;
- перечень и оценки по предметам за предыдущие семестры, а также (если необходимо) за предметы старших классов средней школы;
- информация о профильной лаборатории/кафедре, куда обучающийся распределен;
- выполненные курсовые работы и проекты за время учебы;

- иные полученные результаты, достижения, освоенные компетенции (сюда могут относиться сертификаты онлайн-курсов, выигранные конкурсы и т. п.).

Собираться эти данные могут из анкеты и личного кабинета обучающегося во внутреннем портале образовательного учреждения, часть данных может собираться вручную администрацией учебного заведения, некоторую часть данных обучающийся может представлять о себе сам. В рамках данной работы нет необходимости четко описывать весь объем извлекаемых данных, поскольку решается только определенная задача распределения.

2. СБОР ДАННЫХ ИЗ СОЦИАЛЬНОЙ СЕТИ

Анализ, проведенный в рамках исследования, показал отсутствие открытых инструментов, которые дают возможность высшему учебному заведению получать информацию об абитуриентах и студентах, размещенную на их страницах в социальных сетях, которые могут быть источниками дополнительной информации для формирования личностного портрета студента [3]. Сотрудники высших учебных заведений имеют возможность собирать эту информацию вручную, но для этого требуется большое количество времени, напрямую зависящее от количества абитуриентов и студентов. Для автоматизации этого процесса нами было принято решение разработать инструмент сбора информации из социальных сетей.

Такой инструмент был разработан для социальной сети ВКонтакте; была использована библиотека VK API с открытым исходным кодом [4], представляемая разработчиками ВКонтакте. VK API – это программный интерфейс, который позволяет получать информацию из базы данных vk.com с помощью HTTP-запросов к специальному серверу. Для использования VK API не нужно подробно знать, как строится база данных, из каких типов таблиц и полей она состоит, – достаточно, чтобы API-запрос «знал» об этом. Синтаксис запросов и тип возвращаемых данных строго определены на стороне службы.

На данный момент инструмент позволяет найти следующую информацию: «имя» и «фамилия», информацию из разделов «о себе», «интересы», «образование», «карьера», «любимые книги». Также есть возможность узнать информацию о записях, размещенных студентами и абитуриентами на их страницах, и о группах и сообществах, членами которых они являются.

Пример запроса с использованием VK API на языке Java:

```
List<UserXtrCounters> students = vk.users().get(actor).userIds(studentIds).fields(UserField.ABOUT,
        UserField.BDATE,
        UserField.INTERESTS,
        UserField.SEX,
        UserField.HOME_TOWN,
        UserField.SCHOOLS,
        UserField.CAREER,
        UserField.BOOKS)
        .execute();
```

Пример результата запроса VK API, приходящий в формате JSON[1] (звездочками скрыты персональные данные):

```
[{      "id": ********,
"name": "***** ********",
"birthDate": "**.**.****",
"interests": "",
"sex": "MALE",
"hometown": "Казань",
"school": "Лицей №***;",
"books": "Э. Хемингуэй, Э. М. Ремарк",
"groups": [
        {"name": "че",
        "activity": "Юмор"},
        {"name": "Новые Альбомы",
        "activity": "СМИ"}
        ]
}]
```

Для работы с библиотекой VK API необходимо зарегистрировать разработанное приложение в социальной сети «ВКонтакте». Инструмент возвращает данные в формате JSON, поэтому их можно использовать в других инструментах.

3. РАСПРЕДЕЛЕНИЕ СТУДЕНТОВ ПО АКАДЕМИЧЕСКИМ ГРУППАМ

Вопросы эффективного распределение студентов по академическим группам в рамках высшего образования ставятся довольно часто. Необходимо учитывать целую серию факторов, при этом разные образовательные организации могут по-разному формировать критерии хорошего распределения. В данной работе рас-

[1] JavaScript Object Notation

сматриваются требования Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета (ИТИС КФУ), где:

- группы должны делиться по двум потокам для двух языков программирования;
- в каждой группе должен быть примерно одинаковый уровень по ЕГЭ и английскому языку;
- должны быть по возможности учтены все пожелания студентов к совместной учебе, если это не приведет к серьезному нарушению ограничений, названных выше.

Рис. 1. Распределение студентов на академические группы

Для решения данной задачи был создан инструмент, где в качестве входных параметров о студенте указываются: ФИО, баллы по ЕГЭ, уровень английского языка и выбранный язык программирования (рис. 1). Алгоритм распределения использует кластеризацию [5] методом *K-means* [6] (k-средних), основная идея которого заключается в разбиении множества элементов на несколько кластеров и минимизации среднеквадратического отклонения элементов кластера от его «центра масс».

4. РАСПРЕДЕЛЕНИЕ СТУДЕНТОВ ПО ЭЛЕКТИВАМ

Еще одним видом распределения является разбиение студентов на курсы по выбору или элективы – предметы, которые студенты выбирают самостоятельно для собственного развития. Выбор должен быть последовательным, т. е. выбранный предмет должен сочетаться с текущими компетенциями студента, иначе освоение предмета может быть неэффективным.

В качестве входных данных для данного модуля используются данные как о студентах, так и о курсах по выбору. Основой созданного алгоритма распределения является кластеризация C-means [6], которая позволяет определить вероятность отношения к каждому классу вместо однозначного позиционирования объекта в один кластер. Также преимуществами данного алгоритма являются распределение объекта, основанное на нескольких его признаках, и возможность определения количества и параметров центров кластеров [7] (в качестве центров используются курсы по выбору и информация о них: название, преподаватель, описание курса, данные о студентах, которые окончили данный курс). После окончания работы алгоритма пользователю возвращается матрица принадлежности [8] объектов к каждому кластеру. Схема работы данного алгоритма представлена на рис. 2.

На основе полученной матрицы принадлежности для каждого студента выбираются наиболее подходящий курс, а также список «возможных» курсов, т. е. тех курсов, которые также с достаточно большой вероятностью должны быть полезны данному студенту.

Следующим этапом является формирование групп студентов на курсы по выбору. Существует несколько вариантов разбиения:

- для каждого студента выбирается наиболее подходящий курс;
- помимо наиболее подходящих курсов используется список «возможных» курсов.

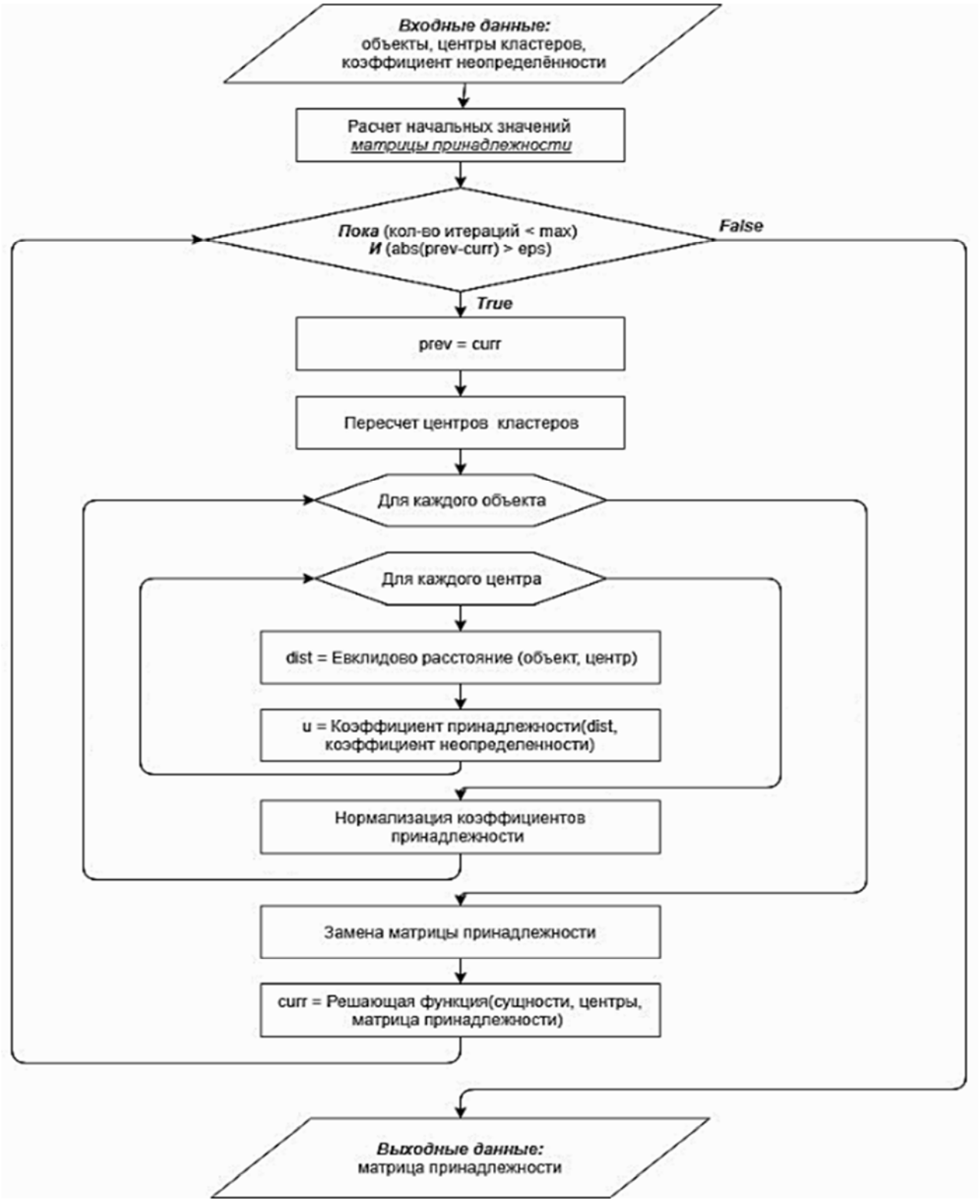


Рис. 2. Кластеризация данных для распределения студентов по элективам

При первом варианте разбивки группы получаются неравномерными, на некоторые курсы попадает большое количество студентов, а на других курсах

студентов достаточно мало. Второй вариант предоставляет возможность сделать группы более равными по количеству, при этом личностные параметры студентов также учитываются. Данный алгоритм распределения предполагает перемещение студентов, для которых существует несколько подходящих курсов, из групп с наибольшим количеством студентов в другие группы из списка «возможных», пока такое перераспределение выполнимо.

5. ПОСТРОЕНИЕ ПРОЕКТНОЙ РАБОТЫ

Частью личностного портрета студента являются набор компетенций, которыми он владеет, и их уровень. На основе этого набора компетенций может быть поставлена задача распределения студентов по проектным командам для выполнения учебного или научного проекта. При этом распределении нужно учитывать как компетенции, которыми участник проекта уже должен владеть, так и компетенции, которые ему необходимо развивать.

На рис. 3 представлена схема проектной работы, где работа выстраивается вокруг ролей, в которых нужно обладать определенным уровнем компетенций и развивать его.

Для решения задачи распределения по проектным командам был реализован специальный инструмент. Входными данными для него являются параметры личностных портретов студентов. Нужно подать на вход *csv* файл, содержащий информацию о студентах в формате: группа, ФИО, список компетенций. Преподаватель может распределить группу студентов на набор проектных команд. При создании одной проектной команды преподаватель имеет возможность задать ему параметры:

- название проекта;
- курс;
- список ролей в проекте;
- список компетенций, требуемых для выполнения этих ролей;
- требуемые и выходные уровни владения компетенциями.

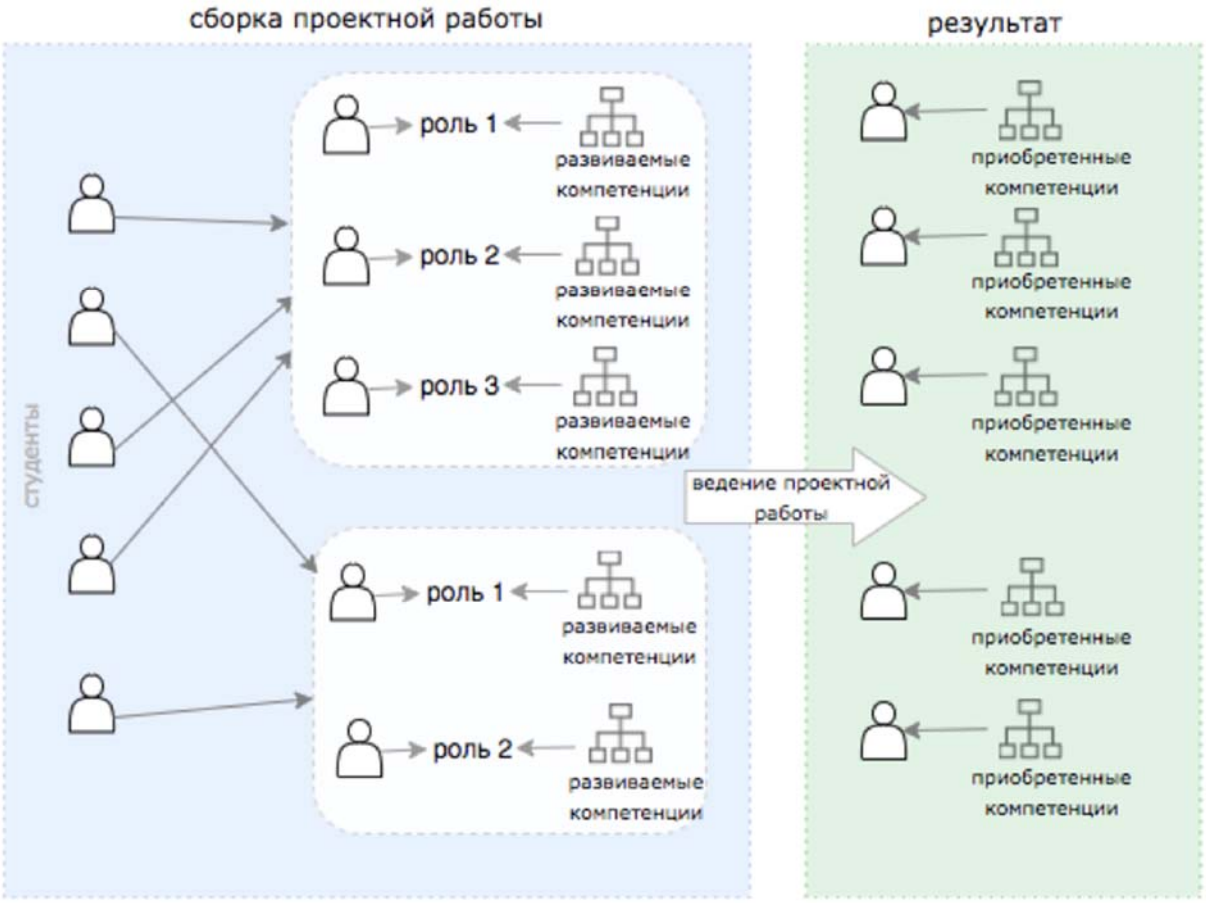


Рис. 3. Схема построения проектной работы

В силу разных целей и критериев распределения инструмент был спроектирован с некоторой долей адаптивности. Было реализованы несколько типов распределения:

- *Распределение студентов по схожим компетенциям.* Это распределение предполагает создание команд, в каждой из которых все студенты должны обладать как минимум одной схожей компетенцией. Для этого необходимо подсчитать, какое количество команд будет выделяться под ту или иную компетенцию путем вычисления пропорции. Затем происходит добавление студентов в команды, компетенциями которых они обладают. Схема работы данного алгоритма представлена на рис. 4.

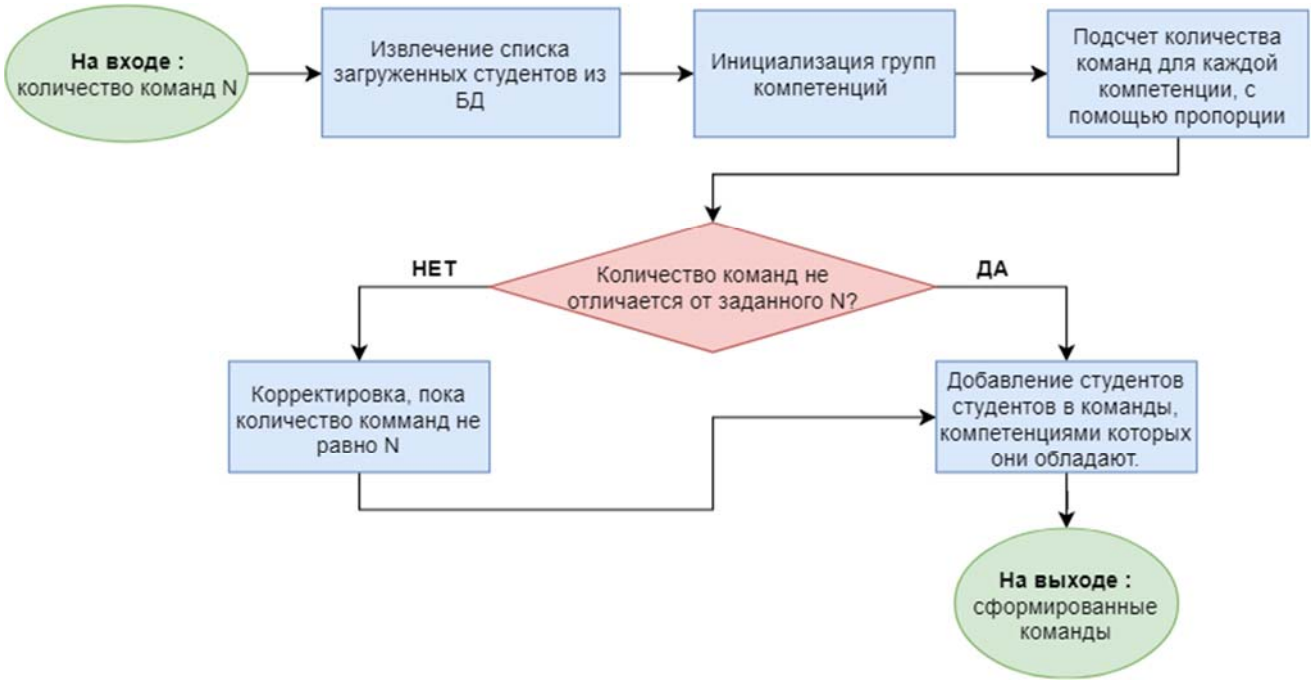


Рис. 4. Схема алгоритма распределения студентов по схожим компетенциям

- *Распределение студентов по различным компетенциям.* В данном распределении предоставляется возможность задать параметры для каждой компетенции или выбрать автоматическую настройку. Параметр для одной компетенции принимает на вход число, которое обозначает количество людей, обладающей данной компетенцией, как минимум входящих в состав каждой команды.

Начальный этап включает инициализацию групп компетенций и добавление в каждую из них студентов, обладающих единственной компетенцией.

Второй этап подразумевает добавление студентов в группы компетенций, пока не будет достигнута *квота* данной компетенции. Квота вычисляется умножением количества команд на параметр пользователя для данной компетенции. Если все квоты, то есть минимальные требования пользователя, достигнуты, то идет переход к последнему этапу, в противном случае необходимо выполнить обмен студентов между группами компетенций. На рис. 5. представлен пример обмена, таким образом можно выполнить обмен не только между двумя студентами, но и составить цепочки из некоторого числа студентов.

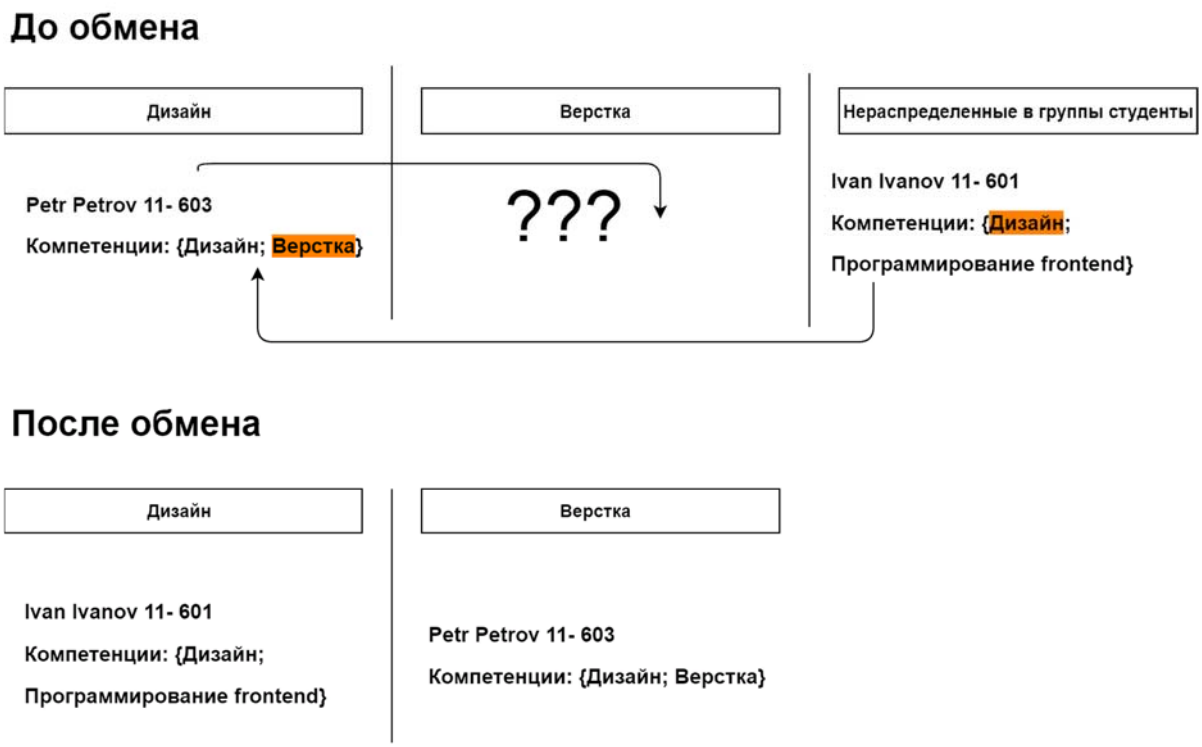


Рис. 5. Пример обмена между студентами

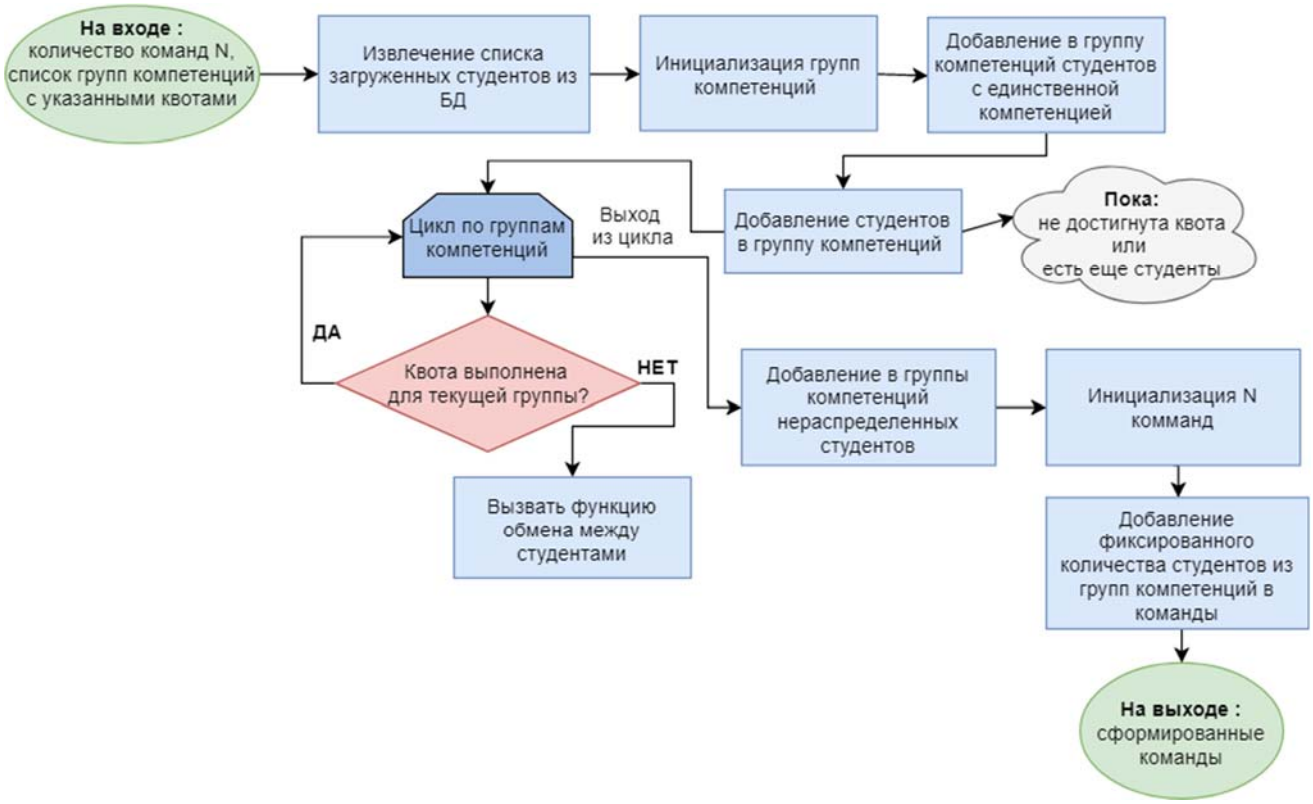


Рис. 6. Блок-схема алгоритма распределения студентов по различным компетенциям

Последний этап включает добавление оставшихся студентов в группы компетенций и далее генерацию команд на основе данных групп. Вся вышеописанного схема алгоритма представлена на рис. 6.

- *Распределение случайным образом.* В зависимости от того, какой набор требуемых компетенций и уровней указал преподаватель, генерируется команда из студентов, наиболее подходящих по заданным параметрам. Для каждой роли указываются компетенции, которые будут развиваться в рамках проектной работы. Также для каждой компетенции вносится значение минимального уровня владения, без которого участие в проекте бессмысленно.

При формировании сущности «проекта» может учитываться также и иерархичность используемых в нем компетенций: преподаватель может указать требуемые входные уровни для всех компетенций, являющихся родительскими по отношению к выбранной. По итогам проектной работы, в случае успешной сдачи проекта, студенты расширяют перечень освоенных компетенций, что может быть отмечено в их личностном портрете.

ЗАКЛЮЧЕНИЕ

Выполнен анализ данных, необходимых для различных видов распределения по группам. Исследованы методы сбора данных и их разбивки, разработаны модели и инструменты для распределения студентов по академическим группам и проектным командам. Созданный инструмент апробирован при распределении студентов по академическим группам, поступивших в 2017 году в Высшую школу ИТИС КФУ.

В дальнейшем планируется расширение возможностей данного инструмента под конкретные академические задачи. Планируется апробировать полученный инструмент в рамках проектной работы студентов.

СПИСОК ЛИТЕРАТУРЫ

1. Указ Президента Российской Федерации № 204 от 07.05.2018 г. «О национальных целях и стратегических задачах развития Российской Федерации на период до 2024 года». URL: http://kremlin.ru/acts/news/57425

2. *Селевко Г.К.* Энциклопедия образовательных технологий. Том 1. М.: Народное образование, 2005. С. 224–225.

3. *Семенов А.В., Веретенник Е.В., Пронин А.С.* Формирование учебных групп в университете с помощью анализа социальных сетей // Вопросы образования. 2014. № 3. С. 18–21.

4. Официальный сайт проекта VK API. URL: https://vk.com/dev

5. *Королев М.А.* Статистический словарь. М.: Финансы и статистика, 1989. 623 с.

6. *Котов А., Красильников Н.* Кластеризация данных, 2006. URL: https://logic.pdmi.ras.ru/~yura/internet/02ia-seminar-note.pdf

7. *Нейский И.М.* Классификация и сравнение методов кластеризации // Интеллектуальные технологии и системы. Сб. учебно-методических работ и статей аспирантов и студентов. М.: НОК «CLAIM», 2006. Вып. 8. С. 130–142.

8. *Тараскина А.С.* Нечеткая кластеризация по модифицированному методу С-средних и ее применение для обработки микрочиповых данных // Проблемы интеллектуализации и качества систем автоматизации. 2013. № 13. С. 217–228.

GENERATION OF ACADEMIC GROUPS AND PROJECT TEAMS BASED ON LEARNERS DATA ACQUISITION

N. A. Korgutlova[1], S. Y. Basargina[2], M. M. Abramskiy[3], M. A. Solncev[4], T. S. Buzukina[5]

[1-5] *Higher School for Information Technologies and Intelligent Systems of Kazan (Volga Region) Federal University*

[1]korgutlova97@gmail.com, [2]basargina.svetlana@gmail.com, [3]ma@it.kfu.ru, [4]mrt.solncev@gmail.com, [5]taya.buzukina@gmail.com

Abstract

The questions of usage of the learners' data in the solutions for generating student academic groups, electives and project teams are considered. The applications of Machine Learning clustering algorithms for these tasks are illustrated. The opportunity of usage of social network data is shown.

Keywords*: personal portrait of student, clustering, competence distribution, social networking analysis*

REFERENCES

1. Decree of President of Russian Federation # 204 from 07.05.2018. "O natsional'nykh tselyakh i strategicheskikh zadachakh razvitiya Rossijskoj Federatsii na period do 2024 goda". URL: http://kremlin.ru/acts/news/57425

2. *Selevko G.K.* Entsiklopediya obrazovatel'nykh tekhnologij. Tom 1. M.: Narodnoe obrazovanie, 2005. S. 224–225.

3. *Semenov A.V., Veretennik E.V., Pronin A.S.* Formirovanie uchebnykh grupp v universitete s pomoshh'yu analiza sotsial'nykh setej // Voprosy obrazovaniya. 2014. № 3. S. 54–57.

4. Ofitsial'nyj sajt proekta VK API. URL: https://vk.com/dev

5. *Korolev M.A*. Statisticheskij slovar'. M.: Finansy i statistika, 1989. 623 s.

6. *Kotov A., Krasil'nikov N.* Klasterizatsiya dannykh, 2006. URL: https://logic.pdmi.ras.ru/~yura/internet/02ia-seminar-note.pdf

7. *Nejskij I.M.* Klassifikatsiya i sravnenie metodov klasterizatsii // Intellektual'nye tekhnologii i sistemy. Sbornik uchebno-metodicheskikh rabot i statej aspirantov i studentov. M.: NOK «CLAIM», 2006. Vypusk 8. S. 130–142.

8. *Taraskina A.S.* Nechetkaya klasterizatsiya po modifitsirovannomu metodu S-srednikh i ee primenenie dlya obrabotki mikrochipovykh dannykh // Problemy intellektualizatsii i kachestva sistem avtomatizatsii. 2013. № 13. S. 217–228.

СВЕДЕНИЯ ОБ АВТОРАХ

КОРГУТЛОВА Наталья Александровна – студентка Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета.

Natalya Aleksandrovna KORGUTLOVA – student of the Higher School of ITIS KFU.

email: korgutlova97@gmail.com

БАСАРГИНА Светлана Юрьевна – студентка Высшей Школы Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета.

Svetlana Yurievna BASARGINA – student of the Higher School of ITIS KFU.

email: basargina.svetlana@gmail.com

АБРАМСКИЙ Михаил Михайлович – старший преподаватель кафедры программной инженерии Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета.

Mikhail Mikhailovich ABRAMSKIY – senior lecturer at Department of Software Engineering of Higher School of ITIS KFU

email: ma@it.kfu.ru

СОЛНЦЕВ Марат Альбертович – студент Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета.

Marat Albertovich SOLNCEV – student of the Higher School of ITIS KFU.

email: mrt.solncev@gmail.com

БУЗУКИНА Таисия Сергеевна – студентка Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета.

Taisiya Sergeevna BUZUKINA – student of the Higher School of ITIS KFU.

email: taya.buzukina@gmail.com

Материал поступил в редакцию 31 июля 2018 года

УДК 004.774.6+37.016

ИНСТРУМЕНТЫ ПОДДЕРЖКИ РОЛЕВЫХ ЗАДАНИЙ ПО СТРАТЕГИИ STAD В ОБУЧАЮЩЕЙ СИСТЕМЕ

В. В. Матюнин[1], А. А. Марченко [2]

[1,2] Высшая школа информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета

[1]vlad.matyunin96@gmail.com, [2]anton.a.marchenko@gmail.com

Аннотация

Представлена одна из возможных реализаций модели совместного обучения по ролям, основанной на стратегии STAD (Student Teams-achievement Divisions) кооперативного обучения в LMS (Learning Management System, Система управления обучением). Подходы, описанные в данной образовательной методике, развивают у обучающихся навыки командной работы, необходимые в профессиональной деятельности, а их внедрение в систему обучения позволит автоматизировать и оптимизировать некоторые процессы и открыть новые возможности для реализации новых инструментов.

***Ключевые слова**: кооперативное обучение, STAD, LMS, обучающие системы*

ВВЕДЕНИЕ

Большинство систем управления обучением (далее LMS – Learning Management System) акцентирует свое внимание на развитие прикладных навыков. Для специалистов программной инженерии примерами подобных навыков являются программирование на определенном языке и использование различных инструментов разработки. Также современные LMS содержат инструменты для формирования прикладных навыков: тестирование, онлайн-уроки с возможностью выполнять и проверять задания. С их помощью обучаемые могут закрепить пройденный материал, изучить пропущенные темы.

Безусловно, прикладные навыки необходимы для будущих специалистов. Однако от них также требуются гибкие навыки (soft skills): работа в команде, адаптация к новым требованиям и условиям. Одним из основных сценариев их

приобретения является коллективная работа. В работах [1, 2] отмечается, что самые распространенные LMS, такие, как Moodle, Blackboard, хотя и обладают достаточным функционалом (хранение и распределение материалов, тестирование, построение курсов и уроков), но не соответствуют современным стандартам обучения с совместным выполнением заданий, автоматизацией.

Несмотря на наличие некоторых коммерческих e-learning платформ с возможностью поддержки командной работы (Teamie, Projectfoundry, Mindscroll), пока не существует платформ, сфокусированных на ролевых заданиях. Их основной задачей является создание окружения, в котором студенты могут работать над проектом или задачей совместно. Также роли каждого из учащихся в таких системах не декларированы. Из этого можно сделать вывод, что на данный момент не существует LMS с поддержкой ролевых заданий.

МОДЕЛИ СОВМЕСТНОГО РОЛЕВОГО ОБУЧЕНИЯ

Среди большого множества моделей обучения с поддержкой совместной ролевой деятельностью наиболее популярными являются модели кооперативного, совместного и проектного обучения. Каждая из них по-своему представляет то, как учащиеся обучаются, решая определенные задачи в группах.

В работе [3] *проектное обучение* определяется как модель, которая организует образовательный процесс вокруг проектов, при этом под проектом подразумевается множество сложных задач, побуждающих учащихся планировать, исследовать и принимать решения. Проектное обучение также дает возможность студентам работать относительно независимо друг от друга в течение продолжительного времени. Работа над проектом мотивирует студентов, формирует у них чувство удовлетворенности, повышает ответственность.

В *кооперативном обучении* основной упор делается на взаимодействие студентов, которое не обязательно можно достичь с помощью групповых заданий. В 1994 году Роджер Джонсон и Давид Джонсон сформировали 5 основных терминов данной модели [4]:

1. Положительная взаимозависимость – принцип, при котором успех каждого из членов команды напрямую зависит от остальных, при этом учащийся не может преуспеть без вклада его коллег;
2. Индивидуальная ответственность – правило кооперативного обучения, когда каждый член команды несет ответственность за групповой результат.

Она проявляется тогда, когда происходит оценка работы учащегося и результаты возвращаются к нему и его команде;

3. Прямая поддержка необходима для реализации положительной взаимозависимости, то есть нужно мотивировать студентов для помощи своей команде, тем самым повышая общий успех команды;

4. Чтобы пункт, описанный выше, действительно работал, необходимо развивать в учащихся навыки общения;

5. Групповая работа проявляется, когда команда обсуждает текущий общий результат и результат каждого в команде индивидуально.

Совместное обучение – это модель, при которой двое или более студентов совместно изучают что-либо, при этом количество обучающихся может быть совершенно разным (группа 2–4 людей, большая группа или даже сообщество). При этом чаще всего основная учебная деятельность в данной модели сконцентрирована вокруг решения какой-либо задачи, а само обучение является лишь результатом данного подхода. При совместном обучении большую роль играют также сами задания: они должны быть построены на тесном взаимодействии внутри группы учащихся. Такой подход имеет как ряд плюсов, так и ряд минусов: из-за тесной связи внутри коллектива каждый из студентов демонстрирует и укрепляет свои способности работать сообща, разделять или нести ответственность, но при этом та же связь создает множество трудностей при оценке результатов (при совместном обучении оценивается результат всей команды в целом) и зависимость внутри команды (один из учащихся не может завершить свою задачу, так как она связана с задачами его коллег).

СТРАТЕГИЯ STAD

Стратегия STAD является одной из разновидностей кооперативного обучения и была сформирована американским психологом Робертом Славиным [5]. Основной ее идеей является работа в небольших командах из 4–5 участников с разными способностями, успеваемостью. Образовательный процесс при использовании данной стратегии можно описать следующим образом: в начале преподаватель формирует группы, после чего формирует задания для каждой из них. Далее происходит преподавание материала всем учащимся, которое закрепляется их командным заданием. После его выполнения происходит тести-

рование каждого из студентов лично, и на основе данных оценок формируется общий результат команды. Важно отметить, что в данном случае главным отличием такой методики от традиционного обучения является выполнение задания, а преподавание материала остается прежним (преподаватель рассказывает материал всем учащимся).

В работе [6] отмечается, что при STAD-стратегии студенты демонстрировали гораздо более высокие результаты на финальном тестировании, чем их коллеги, обучавшиеся методами традиционного обучения.

При формировании групп преподаватель может также обозначить роли внутри каждой из команд. Это не только позволит повысить общую эффективность команды, но и даст возможность студентам обучаться навыкам командной работы в фиксированной роли.

ПРЕДПОЛАГАЕМЫЙ ОБРАЗОВАТЕЛЬНЫЙ ПРОЦЕСС

Из-за простоты стратегии STAD ее легко реализовать в формате обучающей системы. Если первую часть обучения (преподавание теоретического материала) можно перенести на формат LMS, то для поддержки совместной работы над задачами необходимы специальные инструменты.

При формировании образовательного процесса внутри будущей платформы были сформированы два главных типа ее участников: преподаватель и студент. Задание для групп можно представить в виде проекта, который строго делится на временные периоды. В течение каждого из таких периодов преподаватель вносит общее большое задание, которое студенты преобразуют в небольшие задачи. По истечении данного периода преподаватель выставляет оценку за выполненное задание всей группе.

Задача относится строго к роли ответственного в проекте. Соответственно, обязанности каждого из студентов зависят от его роли в проекте. Так как реализуемая платформа адаптирована для будущих ИТ-специалистов, основные роли в команде могут быть следующими: руководитель проекта, frontend-разработчик, backend-разработчик и тестировщик. При этом возможны добавление новых ролей или адаптация платформы для специалистов других отраслей. Основными задачами руководителя являются формирование задач из общего задания преподавателя и контроль их выполнения другими участниками. Для других участников главной обязанностью является выполнение строго своих

задач (относящиеся к их роли). Также допускается, что один и тот же студент может быть вовлечен в две разные команды в разных ролях, поэтому необходимо контролировать общую загруженность учащегося.

Деятельность преподавателя на этой платформе можно описать следующим образом: после формирования команд преподаватель назначает задание каждой из групп. Во время его выполнения он может вмешиваться в процесс, помогать, давать советы. Также возможны изменение состава команд и ролей или добавление новых участников. После окончания этого периода команда сдает задание преподавателю и получает общую оценку, которую члены команды делят между собой. Преподаватель может закончить проект, изменить состав команды или оставить его неизмененным и добавить новое задание. Далее начинается новый цикл, описанный выше.

Несмотря на полную картину описанного образовательного процесса, он все еще не реализует полностью стратегию STAD. Чтобы достичь этого, необходимо также добавить ту часть стратегии, когда преподаватель обучает студентов и проводит индивидуальное тестирование. Данные требования полностью покрывают большинство современных LMS, в которых имеется функционал для создания и проведения тестов, распространения учебных материалов разного формата. Необходимо лишь провести интеграцию между LMS и платформой.

РЕАЛИЗАЦИЯ

Для реализации необходимых инструментов для поддержки заданий по ролям необходимо определить приоритеты каждого из них. В исследовании, описанном в работе [7], отмечается, что, хотя такие интерактивные инструменты, как чат или вики, высоко оценивались пользователями, согласно логированию использовалось меньше других инструментов. Из этого можно заключить, что приоритетными будут инструменты, связанные с управлением проектом.

Как уже отмечалось ранее, ядром платформы является Gitlab, вокруг которого строится веб-приложение. Оно предоставляет дополнительный функционал для управления проектами и работы по ролям: создание задач, трекинг ошибок, ограничение действий пользователя на Gitlab в зависимости от роли в проекте.

Также имеются инструменты для общения, такие, как чат, заметки внутри проекта, оповещения. Это необходимо для установления контакта, когда члены команды не знакомы друг с другом.

Ключевым функционалом является страница проекта с доской, на которой размещаются задачи. Для каждого периода и каждого члена команды предусмотрен свой раздел. На отдельной странице имеется возможность детального просмотра задачи: прикрепленных документов, полного описания и комментариев.

Основные модули

Серверная часть приложения состоит из следующих блоков:

- Model – базовые сущности, которые отображены в базе данных, скрипты миграции;
- Provider-сервисы, работающие с базой данных. В данном проекте также реализованы провайдеры, контролирующие доступ к данным;
- BackendPortal – веб-приложение, обрабатывающее запросы и хранящее конфигурации;
- PortalService – сервисы и расширения, необходимые для веб-приложения: расширение для случайного выбора из коллекции, сервисы оповещений, работы с файлами, сервис для чата и формирования команд;
- GitlabClient – клиент для работы с Gitlab. Сервисы, формирующие запросы к Gitlab API и десериализирующие присланные данные;
- SimulationService – сервисы, отвечающие за симуляции проектов и ролей.

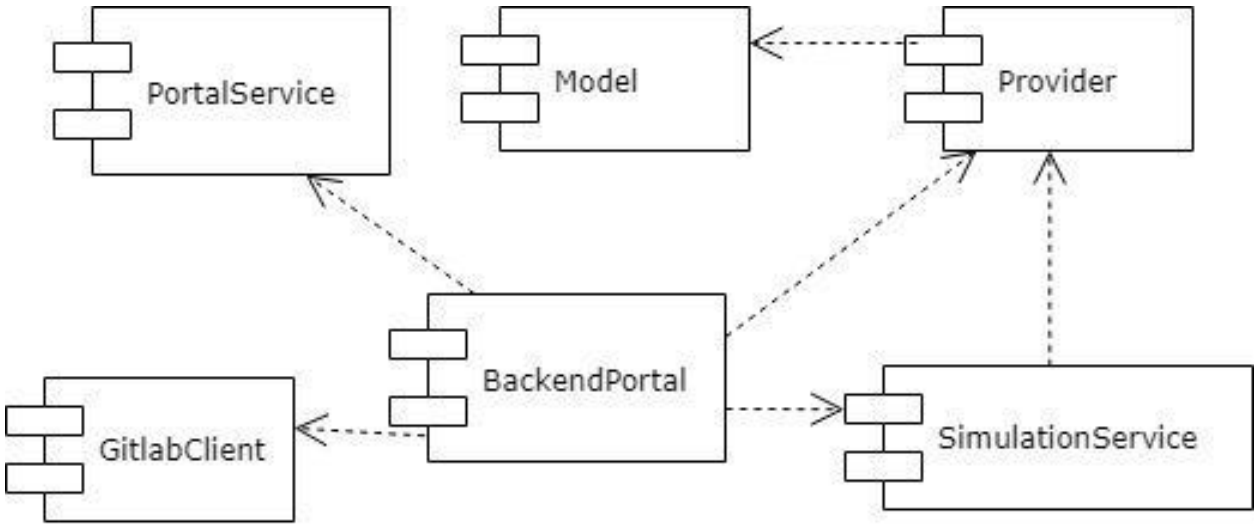


Рис. 1. Диаграмма компонент серверной части приложения

Описание имеющихся сущностей

User – Пользователь системы. Это может быть студент или преподаватель. Каждая из этих ролей имеет свои права на платформе.

Task – Задача на проекте в текущем периоде. Каждая задача прикрепляется только к одному пользователю. Имеет атрибуты приоритета (низкий, средний и высокий), тип (баг или новый функционал), статус (в ожидании, сделана, подтверждена). Задачи могут быть созданы исключительно только преподавателем или руководителем проекта. Каждая задача прикреплена к проекту, пользователю и периоду (Milestone).

Milestone – период, в течение которого группа работает над проектом. Имеет дату начала и конца периода, заголовок и описание. Также у каждого периода есть прикрепленные заметки и чат.

Project – сущность, содержащая данные о проекте: ссылку на gitlab-репозиторий, текущий состав команды с ролями, название, описание и прикрепленные документы.

ProjectRole – роль студента в проекте, например, это может быть разработчик серверной или клиентской части, руководитель проекта или тестировщик.

Безопасность

В качестве основного инструмента для аутентификации и авторизации было решено использовать JWT. JSON Web Token (JWT) – это открытый стандарт

(RFC 7519) для создания токенов доступа, основанный на JSON формате. Как правило, используется для передачи данных авторизации в клиент-серверных приложениях. Токены создаются сервером, подписываются секретным ключом и передаются клиенту, который в дальнейшем использует данный токен для подтверждения своей личности [8].

Провайдеры данных

В качестве базового интерфейса для работы с базой данных был использован CRUD-провайдер с базовыми операциями, реализованными в абстрактном классе:

- найти сущность по уникальному идентификатору;
- найти все сущности из таблицы;
- удалить сущность по уникальному идентификатору;
- создать сущность;
- обновить сущность.

Все остальные провайдеры, наследующие этот класс, дополняли его необходимыми функциями.

Для обеспечения безопасности данных для каждого провайдера был реализован декоратор, реализующий тот же интерфейс и переопределяющий каждый метод. Сначала происходит проверка наличия у пользователя прав на данную операцию. Если прав достаточно, происходит выполнение операции, иначе высылается ошибка клиенту.

Для работы с Gitlab были реализованы сервисы для работы с проектами и пользователями. Например, при создании пользователя в платформе формируется профиль для него на Gitlab, аналогично с проектами. Для доступа был сгенерирован ключ авторизации на Gitlab, который добавляется в заголовки каждого запроса. API данной гит-платформы дает возможность полностью контролировать всеми имеющимися сущностями.

Также разработан сервис для переопределения состава команд: преподаватель может изменить составы команд разных проектов, причем роли и новый состав определит сама платформа. В дальнейшем данный функционал можно будет сделать более адаптивным и формировать новые команды на основе определенных параметров.

Алгоритм формирования команд:

- изначально имеются проекты и позиции, на которые нужно добавить новых студентов;
- также имеются студенты, которых можно использовать в формировании новых команд;
- далее идет итерация по требуемым ролям для каждого проекта; из выбранных студентов берутся те, кто не был еще на этом проекте и не имеет этой роли в другом проекте; из сформированного множества студентов случайным образом выбирается один студент и добавляется к этому проекту.

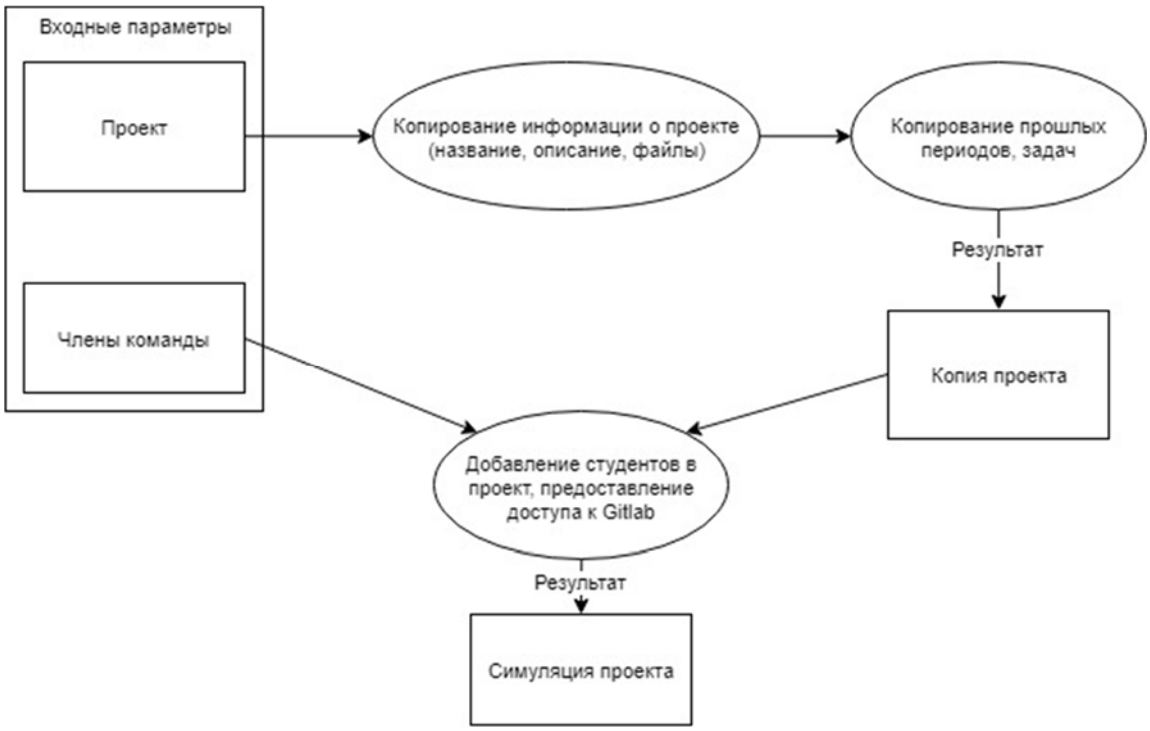


Рис. 2. Диаграмма симуляции проекта

Сервис оповещений построен следующим образом: все оповещения имеют два статуса (новое оповещение и просмотренное), клиент каждые десять секунд посылает запрос к данному сервису, который высылает новые оповещения. Пользователь может просмотреть оповещение, нажав на специальную кнопку. Тогда в меню на сайте это оповещение будет помечено как прочитанное и далее не будет отображаться в специальном индикаторе новых оповещений.

Рис. 3. Сервис для работы с файлами

Для управления файлами разработан сервис, реализующий необходимый функционал, работающий с файловой системой и базой данных. При добавлении файла делается запись в таблице БД и на диске. Все остальные операции происходят тем же образом.

Клиентская часть

Базовой частью всей клиентской части является фреймворк Angular 4 [9]. Была также использована библиотека Angular Material [10], которая позволяет использовать готовые элементы графического интерфейса.

При логине сгенерированный JWT-токен сохраняется в браузере и далее при каждом запросе добавляется в заголовок запроса. При выходе с платформы этот токен удаляется.

ЗАКЛЮЧЕНИЕ

Несмотря на большое количество обучающих платформ разных типов, ни в одной из них не реализована поддержка совместного ролевого выполнения заданий. В данной работе рассмотрен один из возможных вариантов реализации подобной платформы. В качестве основной модели обучения выбрана стратегия STAD кооперативного обучения, а в качестве главного инструмента для поддержки групповой работы над проектами – Gitlab. Реализованная система предоставляет инструменты для управления проектами, поддержки выполнения заданий по ролям, а также имеет дополнительный функционал для обеспечения коммуникаций внутри команд.

В дальнейшем на основе имеющихся данных о проектах и задачах возможно создание симуляций проектов: преподаватель сможет выбрать определенный период реализации проекта и использовать его для обучения новых студентов. Это позволит формировать специальные ситуации для развития определенных компетенций у учащихся.

СПИСОК ЛИТЕРАТУРЫ

1. *Стивен Л., Стефани Д.* Экономия времени или инновационная практика: исследование восприятия и использования систем управления обучением // Эльзевир. 2008. №53 (3). С. 5.

2. *Жао Д., Ксайлонг Ф., Кан Ф., Кифенг Л.* Интерактивная и совместная платформа электронного обучения с интегрированным социальным программным обеспечением и системой управления обучением. Изд-во ИТСЕ, 2012. С. 11–13.

3. *Джон В.* Обзор исследований по обучению на основе проектов. Изд-во The Autodesk Foundation, 2000. С. 1–3.

4. *Джонсон Д., Джонсон Р.* Что заставляет кооперативное обучение работать// Лоуренс Эрлбаум Ассошиэйтс, 1999. №38 (2). С. 2.

5. *Роберт Славин.* URL: http://education.jhu.edu/directory/robert-e-slavin-phd/

6. *Диленбург П.* Что вы подразумеваете под совместным обучением? Изд-во Оксфорд Эльзевир, 2007. С. 4.

7. *Ван Д.Т.* Эффекты стратегии СТАД по академическим достижениям и отношения учащихся средней школы в математике // Международный научный журнал. 2013. С. 1–3.

8. JSON Web Token. URL: https://ru.wikipedia.org/wiki/JSON_Web_Token

9. Angular Framework. URL: https://angular.io/

10. Angular Material. URL: https://material.angular.io/

INSTRUMENTS SUPPORTING ROLE-BASED EXCERCISES USING STAD STRATEGY IN E-LEARNING SYSTEMS

V. V. Matyunin[1], A. A. Marchenko [2]

[1,2] *Higher School of Information Technologies and Intelligent Systems at Kazan (Volga region) Federal University*

[1] vlad.matyunin96@gmail.com, [2]anton.a.marchenko@gmail.com

Abstract

In this paper one of the possible implementations of cooperative education models based on STAD (Student Teams-achievement Divisions) strategy of cooperative learning in LMS (Learning Management System) is described. The approaches of this learning methodology increase the level of teamwork skills, need in further professional activity, and their injection into LMS can help to automate and optimize some processes and open new opportunities for implementation of new instruments.

Keywords*: cooperative learning, STAD, LMS, e-learning*

REFERENCES

1. *Steven L., Stephanie D.* Saving time or innovating practice: Investigating perceptions and uses of Learning Management Systems // Elsevier. 2008. No 53 (3). S. 5.

2. *Zhao D., Xiaolong F., Can F., Qifeng L.* Interactive and Collaborative E-Learning Platform with Integrated Social Software and Learning Management System. Izd-vo ITSE, 2012. S. 11–13.

3. *John W.* A review of research on project-based learning. Izd-vo The Autodesk Foundation, 2000. S. 1–3.

4. *Johnson D., Johnson R.* What Makes Cooperative Learning Work// Lawrence Erlbaum Associates, 1999. No 38 (2). S. 2.

5. *Robert Slavin.* URL: http://education.jhu.edu/directory/robert-e-slavin-phd/

6. *Dillenbourg P.* What do you mean by collaborative learning? Izd-vo Oxford Elsevier, 2007. S. 4.

7. *Van D.T.* Effects of Student Teams Achievement Division (STAD) on Academic Achievement, and Attitudes of Grade 9th Secondary School Students towards Mathematics // International Journal of Sciences. 2013. P. 1–3.

8. JSON Web Token. URL: https://ru.wikipedia.org/wiki/JSON_Web_Token

9. Angular Framework. URL: https://angular.io/

10. Angular Material. URL: https://material.angular.io/

СВЕДЕНИЯ ОБ АВТОРАХ

МАТЮНИН Владислав Владимирович **–** бакалавр Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета.

Vladislav Vladimirovich MATYUNIN **–** Bachelor of the Higher School of Information Technologies and Intelligent Systems at Kazan (Volga region) Federal University.

e-mail: vlad.matyunin96@gmail.com

МАРЧЕНКО Антон Александрович **–** преподаватель Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета, ассистент кафедры программной инженерии.

Anton Aleksandrovich MARCHENKO **–** lecturer of the Higher School of Information Technologies and Intelligent Systems at Kazan (Volga region) Federal University, department assistant of software engineering.

e-mail: anton.a.marchenko@gmail.com

Материал поступил в редакцию 2 июня 2018 года

УДК 004.414.3

РАЗРАБОТКА ИГРОВОГО ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ ОБУЧЕНИЯ ЯЗЫКУ ПРОГРАММИРОВАНИЯ JAVA С ИСПОЛНЕНИЕМ КОДА В РЕАЛЬНОМ ВРЕМЕНИ

Л. Р. Нуруллина[1], Д. Д. Ильясов[2], А. И. Хайруллин[3], Р. Р. Мирхусаинов[4], М. Р. Сидиков[5], М. М. Абрамский[6], А. Р. Ахметшин[7]

[1-7] *Высшая школа информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета*

[1]sadenliia@gmail.com, [2]damirilyasov1997@gmail.com, [3]az.khayrull@gmail.com, [4]mirhusainow@gmail.com, [5]sidikov.marsel@gmail.com, [6]ma@it.kfu.ru, [7]raakhme@gmail.com

Аннотация

Разработан прототип приложения, обучающего в игровой форме базовому синтаксису языка Java. Рассмотрены вопросы связи между реализацией игрового процесса и обучающими упражнениями, в которых должен исполняться программный код. Приведены геймплей и архитектура клиентской и серверной частей.

***Ключевые слова**: язык программирования java, обучение программированию, игровые обучающие приложения, клиент-серверные приложения, фреймворк Spring*

ВВЕДЕНИЕ

Большинство современных профессиональных разработчиков в свое время обучалось программированию со значительной долей самостоятельной работы [1]. Игровой подход к обучению может существенно облегчить обучение синтаксису языков программирования [2]. Существует несколько успешных проектов для разных языков программирования, которые используют его. Один из ярких примеров – RubyWarrior [3] – онлайн-игра, позволяющая управлять героем с помощью написания программного кода. С ее помощью можно сравнительно быстро обучиться базовым конструкциям и синтаксису языка программирования Ruby.

По данным ресурса StackOverflow.com на 2018 год, Java входит в топ-5 популярных языков программирования [1]. Существует несколько игровых онлайн-приложений для обучения синтаксису Java, самые популярные среди них – JavaRush [4] и CodinGame [5]. На JavaRush материал дается в рамках некоторой истории из игрового сюжета, с которой связывается задание. Но выполнение задания не вызывает никакой интерактивной реакции в пользовательском интерфейсе, а также пользователю дается для прочтения большой объем материала. У CodinGame есть определенный сюжет, связанные с ним задания и их графическая интерпретация, но она не связана в реальном времени с логикой кода, который пишет обучающийся. В связи с этим появилась идея создать игровое онлайн-приложение, где пользователь мог бы управлять героем с помощью программного кода, тем самым изучая синтаксис и базовые конструкции языка программирования.

В настоящей работе рассмотрены основные вопросы реализации подобного приложения. В разделе 1 описана концепция игры. В разделе 2 представлены описание разработки серверной части приложения, ее архитектура. Раздел 3 представляет создание браузерного приложения. В разделе 4 обсуждены вопросы проверки правильности кода и его исполнения.

1. ГЕЙМПЛЕЙ

В основу концепции по аналогии с приложением [3] было решено положить историю о герое, который преодолевает препятствия и сражается с врагами на пути к основной цели (прохождению уровня). Управление героем осуществляется посредством написания программного кода, описывающего движения игрока. Такая модель дает возможность неограниченно создавать новые уровни, повышая их уровень сложности. Разработка игры строилась на основе следующего сюжета.

Главный персонаж работает в подразделении службы специального назначения (S.W.A.T.), его отправляют на задание, финальной целью которого является поимка опасного преступника. На начальном этапе в качестве игровых препятствий было решено создать «пики» и «врагов», которые могут наносить урон.

Пользователь может совершать четыре основных действия: передвижение, передвижение в прыжке, атака врага, отдых – для восстановления очков здоровья (*health points, HP*).

Существует возможность проверки ситуаций: наличие врага, наличие препятствия, количество HP.

Таким образом, пользователь может проверять возможные опасности на поле и в зависимости от них определять, какие действия должен сделать герой. Схема взаимодействий изображена на gameplay-диаграмме (рис. 1).

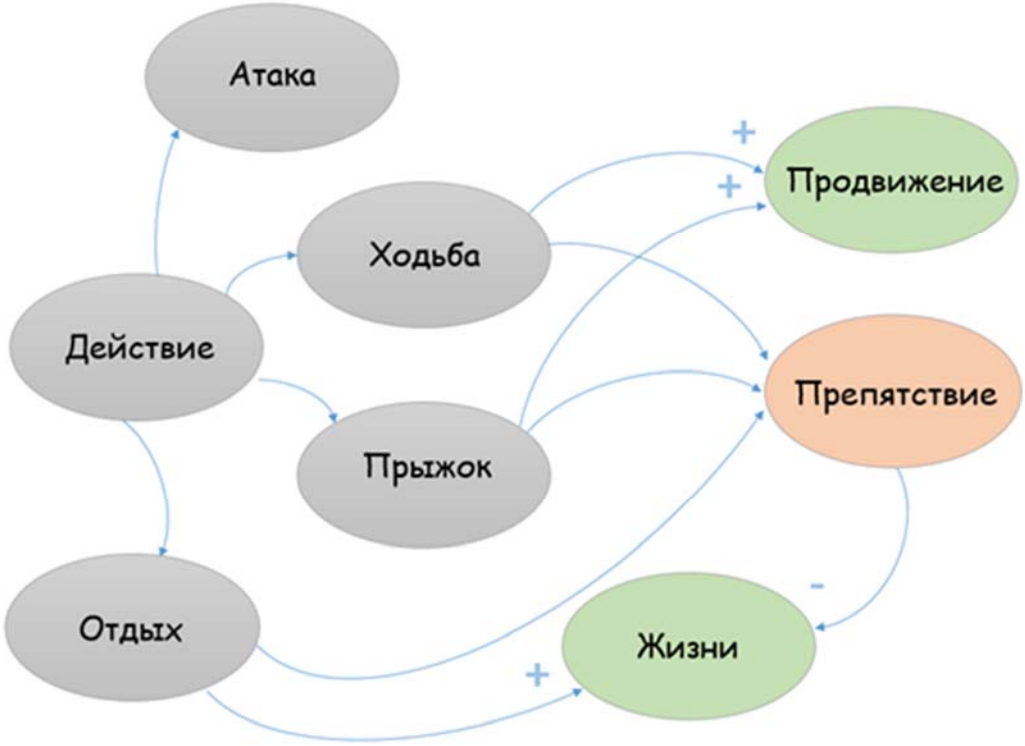


Рис. 1. Gameplay-диаграмма

Если герой в движении попадает на препятствие, то теряет очки HP, иначе – продвигается вперед. Если герой отдыхает – HP увеличиваются. Но если при этом он был рядом с врагом, то одновременно его HP уменьшатся.

Игра поделена на уровни. С каждым уровнем увеличиваются сложность, а также количество навыков, которые нужно применить. На первом уровне доступна всего одна возможность, то есть один метод – метод ходьбы. Таким образом, пользователь должен понять, как вызываются методы. На втором уровне появляется метод для прыжка. Далее добавляются препятствия. С их появлением реализуется метод проверки их наличия. При появлении методов для определения ситуации программисту необходимо также научиться писать условия.

Поле с расставленными на нем препятствиями, а также его размерность будут генерироваться на каждом уровне случайным образом, но с учетом сложности

уровня. Такой подход дает пользователю возможность проходить игру заново, даже если она уже была пройдена им.

В игре предусмотрена возможность создавать аккаунт для сохранения результата и создания рейтинговых таблиц, поскольку соревновательный аспект может добавить игроку мотивации для улучшения собственных результатов.

2. АРХИТЕКТУРА СЕРВЕРНОЙ ЧАСТИ

За основу при разработке серверной части был взят фреймворк Spring Boot [6]. Сервер имеет контроллер с одним главным методом, который принимает на вход код, написанный пользователем (игроком) в виде строки (тип String). Данный код обрабатывается слоем сервисов, затем возвращается ответ. С помощью перечисленной в ответе информации клиентское приложение отображает визуализацию действий персонажа и результаты прохождения уровня.

Так может выглядеть модель ответа сервера:

```
{
    “actions” : [
        { “action” : “MOVE_FORWARD” }
        { “action” : “MOVE_FORWARD” }
    ],
    stageCompleted: true
}
```

Из ответа сервера клиентское приложение должно сделать вывод, что персонаж два раза сделал шаг вперёд и прошёл уровень.

Список всех классов-сервисов серверного приложения, через которые проходит пользовательский код, в том числе те, в которых он проверяется на корректность и исполняемость:

- CompileService – сервис, отвечающий за исполнение клиентского кода;
- ActionService – сервис, в котором содержатся все методы, доступные пользователю:
    - walk() – идти вперёд;
    - attack() – атаковать клетку перед собой;
    - jump() – перепрыгнуть одну клетку;
    - rest() – отдыхать (восполняется здоровье);
    - health() – метод возвращает текущее состояние здоровья;

- isEnemyAhead() – возвращает true, если перед персонажем враг, иначе false;
- isSpikesAhead() – возвращает true, если перед персонажем пики, иначе false;

- BanValidateService – сервис, проверяющий пришедший от пользователя код на предмет возможности его компиляции и наличие вредоносных участков.

Для обеспечения безопасности данных пользователей используется Spring Security [7] в сочетании с JWT[2]. Обработка кода игроков, авторизация, регистрация осуществляются на одном сервере. Подробнее об обработке пользовательского кода рассказано в разделе 4.

3. РАЗРАБОТКА БРАУЗЕРНОЙ ВЕРСИИ

Для разработки браузерной версии игры было решено использовать подход одностраничного веб-приложения. Одностраничные приложения (Single-Page Applications, сокр. SPA) – это веб-приложения, состоящие из одного единственного html-документа, содержимое которого динамически изменяется при взаимодействии с сервером под влиянием действий пользователя.

Задачи обновления данных на странице, предоставления переходов между представлениями, называемых роутингом, а также управления состоянием приложения полностью выполняет браузерная часть приложения. Работа пользователя с такими приложениями напоминает работу в настольных программах – действия пользователя не перезагружают интерфейс окна, а содержимое поступает к пользователю посредством его взаимодействия с приложением.

Для разработки одностраничного приложения выбран следующий список технологий:

- **ReactJS** – JavaScript библиотека для создания пользовательских интерфейсов, разработанная компанией Facebook. Расширяя ее возможности с помощью других JavaScript библиотек, таких, как React-Router для организации навигации внутри одностраничного приложения, а также Redux для хранения глобального состояния приложения, ReactJS становится полноценным инструментом для разработки одностраничных приложений [8];

[2] JWT – JSON Web Token

- **TypeScript** – скриптовый язык программирования, компилируемый в JavaScript. Язык расширяет синтаксис JavaScript статической типизацией, поддержкой полноценных классов и подключения модулей [9];
- **Webpack** – сборщик и оптимизатор модулей JavaScript. Он отслеживает зависимости каждого JavaScript скрипта, строя граф зависимостей, и собирает их в один выходной файл, одновременно решая задачу оптимизации и минимизации результатов сборки [10];
- **SASS** – язык на основе CSS, расширяющий возможности работы с файлами стилей. Добавляет в CSS поддержку переменных, функций, циклов, а также наследования и вложенных конструкций [11].

Создание одностраничного приложения в виде игры потребовало применения подходов, обеспечивающих высокий уровень масштабируемости и изменчивости:

- **Методология БЭМ (Блок, Элемент, Модель)** – методология, которая задает правила по именованию CSS-классов. Основана на трех определениях: блок, элемент, модификатор. **Блоком** называется самостоятельный элемент, являющийся независимым от других, который встраивается в страницы. **Элементом** называется составная неделимая часть блока, существование которой в отрыве от него невозможно. **Модификатором** называется сущность, которая определяет внешний вид, состояние и поведение блока или элемента. Примером CSS-класса, определенного с использованием БЭМ, является .header__logo-red, который указывает на логотип шапки сайта красного цвета. Применение данной методологии гарантирует разработчику изолированность стилей элементов на странице;
- **Компонентный подход** – подход, при котором приложение делится на самостоятельные модули, которые можно переиспользовать в различных частях приложения. Примером компонента в игре является компонент игровой карты (рис. 2), который также состоит из компонентов объектов карты, таких, как Главный герой, Шипы и Враг.

Рис. 2. Компонент «Игровая карта»

4. ВАЛИДАЦИЯ И ИСПОЛНЕНИЕ КОДА НА СЕРВЕРЕ

Перед отправкой кода на компиляцию необходимо провести процесс валидации, в котором можно выделить две основные задачи: выявление синтаксических ошибок и проверка пользовательского кода на безопасность.

Для выявления синтаксических ошибок в программном коде пользователя выбор был сделан в пользу готового решения – javax.tools.Diagnostic [12].

Метод для проверки синтаксиса находится в сервисе BanValidateServiceImpl и называется checkSyntax, параметр у которого один – String code (пользовательский код). Так как мы используем класс DiagnosticCollector[13], необходимо записать пользовательский код в специально подготовленный пустой файл и позже скомпилировать его с помощью JavaCompiler. После отработки метода getTask() в экземпляре класса DiagnosticCollector будет определен список Diagnostic – объектов, содержащих в себе информацию о синтаксических ошибках и предупреждениях компилятора. Из каждого экземпляра Diagnostic мы достаем:

- Kind (тип):
  - ERROR (ошибка);
  - WARNING (предупреждение);
  - MANDATORY_WARNING (обязательное предупреждение);
  - NOTE (примечание);
  - OTHER (другое);
- Местонахождение ошибки (включая номер строки и столбца);
- Сообщение.

Если ошибки будут обнаружены, информация о них передается в объект класса Validation с флагом isValid = false. Если ошибок нет, можно перейти к следующему этапу – проверке безопасности.

Возможность пользователя запускать практически любой Java-код на сервере влечет за собой большие риски. Чтобы эффективно противодействовать вредоносному коду, необходимо проанализировать направления возможной атаки:

- **Spring** – данный вектор атаки можно не учитывать, так как код компилируется в процессе выполнения программы, Spring не имеет доступа к скомпилированному классу;
- **throw** – с помощью этого ключевого слова пользователь может сгенерировать RuntimeException, которое сервер не обработает;
- **Рефлексия** – с помощью данного средства можно, например, подобрав имена классов, получить доступ к методам и полям;
- **File** и другие классы для работы с файлами – могут быть использованы для доступа к файловой системе машины, на которой запущен сервис;
- **Runtime** – может дать злоумышленнику доступ к командной строке (если это Windows) одной строчкой кода: Runtime.getRuntime().exec(“cmd”).

Оценив риски и проанализировав возможные векторы атаки, мы создали словарь запрещенных конструкций, которые пользователь не сможет использовать в своем программном коде. Сам процесс проверки кода на содержание запрещенных конструкций (далее – ЗП) реализует метод checkSecurity. В нем с помощью рефлексии и регулярных выражений подготавливается список ЗП и определяется их наличие. Если таковые были обнаружены, создается объект класса Validation с флагом isValid = false и сообщением об использованной ЗП, чтобы пользователь понимал, почему сервис отказался компилировать его код.

При отсутствии ошибок введенный код передается на выполнение. Так как он должен состоять только из операторов и определенных методов, заранее был подготовлен класс, из которого к ним есть доступ.

Данный класс содержит информацию о количестве доступных итераций и указатель на то, было ли совершено какое-либо действие ранее. На выходе мы получаем объект ***GameResult***, который содержит в себе результат исполнения поль-

зовательского кода или же сообщение об ошибке. Принцип работы основной процедуры класса заключается в циклическом выполнении введенного пользователем кода до тех пор, пока не закончатся доступные итерации или уровень не будет пройден.

ЗАКЛЮЧЕНИЕ

Разработан прототип веб-приложения, обучающего базовому синтаксису языка программирования Java. Несмотря на то, что изначально работа носила характер адаптации идеи приложения [3] на язык Java, в рамках проделанной работы была разработана архитектура клиент-серверного обучающего приложения с исполнением кода на Java-технологиях. Данные архитектуру и прототип можно использовать для разработки других подобных приложений, в том числе для обучения языкам программирования, исполняемым на Java Virtual Machine.

СПИСОК ЛИТЕРАТУРЫ

1. Developer Survey Results 2018. URL: https://insights.stackoverflow.com/survey/2018/
2. *Chang C.C., Liang C., Chou P. N., and Lin G. Y.* Is game-based learning better in flow experience and various types of cognitive load than non-game-based learning? Perspective from multimedia and media richness // Computers in Human Behavior. 2017.
3. Официальный сайт RubyWarrior. URL: https://www.bloc.io/ruby-warrior
4. Официальный сайт JavaRush. URL: https://javarush.ru/
5. Официальный сайт CodinGame. URL: https://www.codingame.com/
6. Официальный сайт Spring Boot. URL: https://spring.io/projects/spring-boot
7. Официальный сайт Spring Security. URL: https://spring.io/projects/spring-security
8. Официальный сайт ReactJS. URL: https://reactjs.org/
9. Официальный сайт TypeScript. URL: https://www.typescriptlang.org/
10. Официальный сайт WebPack. URL: https://webpack.js.org/
11. Официальный сайт SASS. URL: https://sass-lang.com/

12. Официальная документация java.tools.Diagnostic. URL: https://docs.oracle.com/javase/6/docs/api/index.html?javax/tools/Diagnostic.html

13. Официальная документация java.tools.DiagnosticCollector. URL: https:// docs.oracle.com/javase/7/docs/api/javax/tools/DiagnosticCollector.html

DEVELOPMENT OF APPLICATION FOR GAME-BASED LEARNING OF JAVA LANGUAGE WITH REALTIME CODE RUNNING

L. R. Nurullina[1], D. D. Iliyasov[2], A. I. Khairullin[3], R. R. Mirkhusainov[4], M. R. Sidikov[5], M. M. Abramskiy[6], A. R. Akhmetshin[7]

[1-7] *Higher School of Information Technologies and Intelligent Systems of Kazan (Volga Region) Federal University*

[1]sadenliia@gmail.com, [2]damirilyasov1997@gmail.com, [3]az.khayrull@gmail.com, [4]mirhusainow@gmail.com, [5]sidikov.marsel@gmail.com, [6]ma@it.kfu.ru, [7]raakhme@gmail.com

Abstract

This paper describes the development of the application prototype for learning Java language syntax. There are raised questions about the connection between gaming process implementation and learning exercises with the code running parts, and given the gameplay, architecture of client and server parts of the application.

Keywords*: java programming language, programming learning, game-based learning applications, client-server applications, Spring Framework*

REFERENCES

1. Developer Survey Results 2018. URL: https://insights.stackoverflow.com/survey/2018/

2. *Chang C.C., Liang C., Chou P. N., and Lin G. Y.* Is game-based learning better in flow experience and various types of cognitive load than non-game-based learning? Perspective from multimedia and media richness // Computers in Human Behavior. 2017.

3. RubyWarrior official page. URL: https://www.bloc.io/ruby-warrior
4. JavaRush official page. URL: https://javarush.ru/
5. CodinGame official page. URL: https://www.codingame.com/
6. Spring Boot official page. URL: https://spring.io/projects/spring-boot
7. Spring Security official page. URL: https://spring.io/projects/spring-security
8. ReactJS official page. URL: https://reactjs.org/
9. TypeScript official page. URL: https://www.typescriptlang.org/
10. WebPack official page. URL: https://webpack.js.org/
11. SASS official page. URL: https://sass-lang.com/
12. Official documentation of java.tools.Diagnostic. URL: https://docs. oracle.com/javase/6/docs/api/index.html?javax/tools/Diagnostic.html
13. Official documentation of java.tools.DiagnosticCollector. URL: https:// docs.oracle.com/javase/7/docs/api/javax/tools/DiagnosticCollector.html

СВЕДЕНИЯ ОБ АВТОРАХ

НУРУЛЛИНА Лия Радиковна – студентка Высшей школы информационных технологи и интеллектуальных систем Казанского (Приволжского) федерального университета (ИТИС КФУ).

Liia Radikovna NURULLINA – student of Higher School of ITIS KFU.

email: sadenliia@gmail.com

ИЛЬЯСОВ Дамир Дмитриевич – студент Высшей школы информационных технологи и интеллектуальных систем Казанского (Приволжского) федерального университета (ИТИС КФУ).

Damir Dmitrievich IlYASOV – student of Higher School of ITIS KFU.

email: damirilyasov1997@gmail.com

ХАЙРУЛЛИН Азат Ильдарович – студент Высшей школы информационных технологи и интеллектуальных систем Казанского (Приволжского) федерального университета (ИТИС КФУ).

Azat Ildarovich KHAYRULLIN – student of Higher School of ITIS KFU.

email: az.khayrull@gmail.com

МИРХУСАИНОВ Руслан Радикович – студент Высшей школы информационных технологи и интеллектуальных систем Казанского (Приволжского) федерального университета (ИТИС КФУ).

Ruslan Radikovich MIRKHUSAINOV – student of Higher School of ITIS KFU.

email: mirhusainow@gmail.com

СИДИКОВ Марсель Рафаэлевич – руководитель лаборатории Java Высшей школы информационных технологи и интеллектуальных систем Казанского (Приволжского) федерального университета (ИТИС КФУ).

Marsel Rafaelevich SIDIKOV – head of Java Lab of ITIS KFU.

email: sidikov.marsel@gmail.com

АБРАМСКИЙ Михаил Михайлович – старший преподаватель кафедры программной инженерии Высшей школы информационных технологи и интеллектуальных систем Казанского (Приволжского) федерального университета (ИТИС КФУ).

Mikhail Mikhailovich ABRAMSKIY – senior lecturer at Department of Software Engineering of ITIS KFU

email: ma@it.kfu.ru

АХМЕТШИН Азат Ринатович – студент Высшей школы информационных технологи и интеллектуальных систем Казанского (Приволжского) федерального университета (ИТИС КФУ).

Azat Rinatovich AKHMETSHIN – student of Higher School of ITIS KFU.

email: raakhme@gmail.com

Материал поступил в редакцию 28 июля 2018 года

УДК 004.414.3

КОНЦЕПТ ИНСТРУМЕНТА АВТОМАТИЧЕСКОГО СОЗДАНИЯ СЦЕНАРНОГО ПРОТОТИПА КОМПЬЮТЕРНОЙ ИГРЫ

Г. Ф. Сахибгареева[1], В. В. Кугуракова [2]

[1,2]*Высшая школа информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета;*

[1]gulnara.sahibgareeva42@gmail.com, [2]vlada.kugurakova@gmail.com

Аннотация

На основе существующих решений описано представление об архитектуре инструмента генерации сценарного прототипа из текста. Сформированы требования к разрабатываемому инструменту. Разработан прототип инструмента, иллюстрирующий основной принцип работы пользователя с приложением.

***Ключевые слова**: игровая сценаристика, нарративный дизайн, сценарный прототип, прототипирование, разработка игр, погружение, повествование*

ВВЕДЕНИЕ

Повествование – это изображение действия во времени. Обычно оно ведется от имени автора или рассказчика. Другими словами, это упорядоченная последовательность связанных событий, в которых участвуют одна или несколько сущностей, и которая излагается в виде истории.

В кинематографе повествование наделено дополнительными свойствами: наглядность и звучание. На глазах у зрителя история оживает в виде конкретных образов.

В играх повествование – это не только то, какая история складывается в голове у игрока и то, какие эмоции вызывает увиденное и услышанное, но и то, что называется уникальным игровым опытом. Игрок «проживает» события игры, становится главным героем истории. Грамотно построенное повествование способно погрузить игрока в вымышленный мир, показать ему, какие механики ему

доступны, научить его ими пользоваться и достигать с их помощью внутриигровых целей. Однако реализация такого повествования – не такая простая задача, какой кажется на первый взгляд. В игровой индустрии решаются задачи, с которыми не сталкиваются сценаристы кино и театра.

Игровые сценаристы и нарративные дизайнеры в своей работе опираются на интерактивную природу игр, т. к. они знают, что хорошее повествование лежит в самой структуре игрового дизайна [1]. Поэтому игровой сценарий отличается тем, что нацелен на то, чтобы рассказать историю игрой.

На начальных этапах работы сценариста сложно оценить, будет ли игра захватывающей, выполнит ли повествование свою функцию, не будет ли в сюжете несостыковок. Только качественное повествование способствует погружению игрока в придуманный сценаристом мир, обеспечивает для него уникальный нарративный опыт [2].

Залог качественного повествования – раннее прототипирование, что также является особенностью разработки в игровой разработке [3]. Прототипирование – основа хорошего игрового дизайна. Прототип – это идея в том виде, в котором ее можно протестировать. Другими словами, это упрощенная модель будущей игры. Хорошая аналогия – это эскизы, которые художники набрасывают перед тем, как писать картину. Прототипирование помогает найти ошибки, устранить несостыковки и снизить риски. Нет нужды в том, чтобы беспокоиться о том, как будет выглядеть прототип, главное, как он будет работать [4].

Предлагаемый нами способ прототипирования повествования – создание и тестирование сценарного прототипа. Из-за отсутствия данного термина в научной литературе, дадим его определение.

Сценарный прототип – это модель игрового повествования, представленная в виде, доступном для демонстрации, анализа и тестирования. На основе сценарного прототипа можно оценить качество создаваемой истории, проверить, есть ли интерес к ней со стороны целевой аудитории. Создание сценарного прототипа минимизирует затраты на начальных этапах разработки, т. к. появляется возможность для тестирования.

ПОСТАНОВКА ПРОБЛЕМЫ

Для разработки сценария существуют инструменты, призванные облегчить структурирование сюжета. Но средств, способных автоматизировать процесс со-

здания структуры сценария, еще нет. Раннее прототипирование возможно только после создания структуры вручную в программах, которые представляют сюжет в виде связных карточек. Прототипирование предполагает быстрый результат без лишних затрат.

Если некоторые игровые механики можно проверить, создав, при возможности, настольную версию игры или упрощенную компьютерную симуляцию, то в случае истории ее нужно рассказать, а лучше, показать команде разработчиков и целевой аудитории игры.

Итак, сформированная проблема заключается в том, что не существует инструмента, способного автоматизировать процесс прототипирования игровой истории.

Развивая идею о том, как может работать инструмент, призванный автоматизировать структурирование сценария, можно предположить, что хорошим выходом станет использование технологий автоматического распознания текста.

На данный момент сильно прогрессируют в развитии сфера машинного обучения, а также сфера работы с нейросетями и алгоритмы анализа больших массивов текста. В сфере лингвистики такие технологии решают вопросы анализа стилистики текста, его смысловой нагрузки. Имеющиеся технологии позволили реализовать генерацию текста на заданную тему. С каждым разом улучшаются механизмы перевода текста с одного языка на другой.

Вопрос извлечения из текста необходимых сущностей стал обыденностью для анализа соцсетей и продвижения брендов [5]. Поэтому извлечение из текста сущностей и последующая их обработка считаются возможными с точностью, которую предлагают существующие инструменты.

Другой вопрос, связанный с автоматической сборкой сценарного прототипа, решается написанием собственных алгоритмов обработки полученных данных.

АНАЛИЗ СУЩЕСТВУЮЩИХ РЕШЕНИЙ

Классический пайплайн

Классический план разработки игры включает в себя несколько последовательных этапов: формирование идеи, предварительная работа, разработка, запуск игры, поддержка игры после запуска [6].

На интересующем нас этапе предварительных работ основное время посвящается решению следующих задач: создание истории, определение временных рамок, создание раскадровки, создание бумажного прототипа, дизайн уровней, реализация игровых механик, расчет бюджета разработки и создание технической документации.

Создание на данном этапе сценарного прототипа означает объединение нескольких видов задач (создание истории, определение временных рамок, создание раскадровки) и создание основы для тестирования. По окончанию этого этапа должно стать ясно, приносит ли игра «фан», захватывает ли она игроков, интересная ли у игры история [7].

Выбор технологии анализа текста

Среди существующих инструментов анализа текста были выделены следующие разработки: Texterra, OpenNLP, Watson Natural Language Understanding, Google Cloud Natural Language. Тестирование данных инструментов (см. Табл. 1) показало, что инструмент от Google эффективнее всех.

Таблица 1. Результаты тестирования инструментов

№	Название инструмента	Типы сущностей	Результат
1	Texterra	ORGANIZATION	1
		TIME	1
2	OpenNLP	ORGANIZATION	1
		TIME	1
3	IBM Watson Natural Language Understanding	Facility	2
4	Google Cloud Natural Language	Person	7
		Location	4
		Organization	1
		Other	14
		CustomerGood	1

В рамках исследования мы проанализировали существующие решения, которые реализуют функционал в следующих направлениях: визуализация структуры, визуализация контента, автоматическая сборка контента, комплексные решения.

Когда мы говорим о визуализации структуры повествования, в голову приходит идея о представлении информации в виде графа. Но создатели инструмента StoryFlow пошли дальше и представили структуру в виде особой структуры Yarn [8]. В ней структуре события располагаются в хронологическом порядке: слева направо, причем каждая следующая сущность связана с предыдущей.

Существующие инструменты

У данной разработки есть версия, расширяющая функционал – Extended StoryFlow [9]. В данной версии учтены случаи, когда персонаж проходит определенное событие самостоятельно, и случаи, в которых возможен выбор. Этот инструмент полезен для красивой и понятной визуализации игровых событий, взаимодействия персонажей и определения временных промежутков.

Инструмент, решающий в некотором плане проблему прототипирования повествования, ScriptViz, позволяет визуализировать сценарий в виде сцены, которая генерируется в реальном времени [10]. Инструмент состоит из следующих модулей: модуля понимания текста, модуля планирования верхнего уровня и модуля генерирования сцены.

Пользователь вводит текст своего сценария. Условия для такого текста следующие: текст должен быть написан на английском, иметь понятные формулировки и хорошо структурирован.

Система интерпретирует текст в модуле понимания текста и отправляет результат в модуль планировщика верхнего уровня. Планировщик строит план действий для определенных скриптов. Скрипты выполняют план, а генератор сцен отображает полученный результат.

Данный инструмент рассматривается в исследовании, проведенном для реализации программного решения SceneMaker [11].

SceneMaker планируется разработать для решения следующей задачи – автоматизированное создание анимационного прототипа фильма. Проведенное для этого исследование имеет аналогичную с нашей цель: на основе прототипа иссле-

довать перспективы разрабатываемого продукта, отработать на нем различные подходы и методы.

Пайплайн работы SceneMaker следующий: из готового сценария извлекаются эмоции, настроение и жанр. Данные визуализируются в виде 3D-сцены, в которой эмоции отражаются через мимику, жесты и позу персонажей, через освещение, тайминг, постановку камеры и настройку аудиосопровождения в автоматическом режиме в зависимости от жанра. На основе визуализации можно сделать выводы о проделанной работе.

Для получения информации создатели SceneMaker берут за основу эмоции, извлекаемые из текста. На их основе в сцене формируется действие, определенное вплоть до мимики, жестов, позы и голоса персонажей, а также вплоть до постановки кадра, освещения и звука.

В качестве инструмента, способного на основе описательного текста подобрать подходящий анимационный материал, интересна нейросеть CRAFT. Принцип её работы состоит в следующем: на основе 25 184 фрагментов из мультсериала Flintstones нейросеть по текстовому описанию собирает мультфильм [12]. В собранных сценах, конечно, ещё встречаются ошибки, такие, как неправильное наложение объектов или неправильный выбор анимаций.

Интерес представляет инструмент Machination Tool [13], способный проанализировать интерактивную составляющую игры.

Machinations – это визуальный язык представления игровых механик в виде диаграмм, создающий игровую имитацию без написания кода. Так как игровая механика скрыта в системе и не доступна для анализа, кроме как из процесса игры, использование Machinations считается эффективным для анализа геймплея. Схемы обработки в диаграммах Machinations представляют собой потоки и структуры обратной связи, которые могут существовать в игровой системе. Именно эти структуры обратной связи в значительной степени дают представление о динамике игры.

Рассмотренные инструменты реализуют функционал, который представляет интерес в рамках нашего исследования, но в целом их недостаточно. Решением видится комплексный инструмент, который смог бы использовать существующие наработки.

НАШЕ ПРЕДСТАВЛЕНИЕ

На основе проведенного исследования был создан прототип инструмента, который иллюстрирует первый взгляд на то, какую архитектуру нужно реализовывать для достижения поставленной цели.

Были сформированы следующие требования: инструмент должен принимать любой, не обязательно формализованный, текст; извлекать максимальное количество полезных сущностей; иметь возможность хранить сущность с привязкой к предложениям; иметь возможность редактировать сущности, относящиеся к предложению; иметь возможность сборки сценарного прототипа с участием пользователя.

Инструмент работает по следующему алгоритму: пользователь вводит текст; затем он просматривает полученные сущности и редактирует их; пользователь выбирает режим сборки прототипа; затем он собирает сценарный прототип, который состоит из сцен, создающихся из сущностей, указанных пользователем предложений; пользователь просматривает полученный сценарный прототип.

Интерфейс инструмента состоит из следующих частей: обработка текстовых данных и сборка сценарного прототипа (Рис. 1).

Обработка текстовых данных

Автоматическое понимание текста включает в себя множество этапов анализа и сопоставления полученных результатов. Следовательно, необходимо использовать сторонние сервисы и разрабатывать собственные алгоритмы обработки.

В нашей реализации предполагается ручное редактирование, поэтому лучше, если сторонний сервис сможет распознать максимальное число полезных сущностей. Под полезными подразумеваются сущности, которые потенциально рассматриваются для сборки сценарного прототипа.

Работа инструмента предполагает следование следующему алгоритму: принять введенный текст, отправить его на обработку на сторонний сервис или в используемую библиотеку, принять и распределить сущности по категориям, сохранить введенные изменения.

Под категориями подразумеваются следующие типы сущностей: действующие персонажи, место действия и артефакты (сущности, присутствие в сцене которых обязательно).

Рис. 1. Архитектура разработанного инструмента

Модуль сборки сценария

Получив на выходе необходимые сущности, можно собрать сценарный прототип по подготовленному шаблону. В качестве шаблона выберем следующую структуру: персонажи, место действия, артефакты. В такой структуре хранятся сущности выбранных для сцены предложений.

Представление сценарного прототипа

Реализовать сценарный прототип можно с разной степенью абстракции. Предлагаемый вариант прост в реализации и скорее является достаточным для прототипа, нежели реальным решением поставленной задачи. Это представление сценарного прототипа в виде сцен, каждая из которых перечисляет необходимые для нее сущности: локацию, персонажей и артефакты.

РЕАЛИЗАЦИЯ ПРОТОТИПА

Рассмотрим принцип работы полученного прототипа. Прототип реализован на языке C#, поэтому используются соответствующие термины.

Обработка текстовых данных

Обработка текста осуществляется по принципу, который был сформирован в ходе разработки. Информация по каждому предложению хранится в виде отдельного объекта, который содержит в себе следующие данные: текст предложения и структура, хранящая в себе сущности, которые найдены в этом предложении. В качестве структуры для хранения этих сущностей выбрана структура Dictionary<TKey, TValue>, т. к. данные, которые возвращаются с Google Cloud Language, включают в себя дубликаты.

Для работы с полученными сущностями выделяется несколько категорий: Person, Location и Atrefact. Категории Google Cloud Language отображаются в соответствии с перечисленными категориями: сущности типа Person отображаются в категории Person, сущности типа Location и Organization отображаются в категории Location и все остальные сущности хранятся в категории Atrefact, чтобы не упустить из виду проанализированные сущности.

Редактирование сущностей осуществляется относительно каждого предложения текста. Если Google Cloud Language не определил какие-то сущности или сделал это неправильно, пользователь может это исправить.

Модуль сборки сценария

Для создания сценарного прототипа пользователь должен выбрать несколько предложений, подтвердить свой выбор, и программа выведет значения сущностей, которые будут задействованы в данной сцене.

Теперь пользователю доступен сценарный прототип, который представляет собой определенное количество сцен, каждая из которых включает в себя следующие категории: персонажи, локации и артефакты. В дальнейшем данный прототип может использоваться для планирования этапов разработки, проверки плотности событий в сцене и для отслеживания развития событий.

Наша реализация – первый шаг на пути создания полноценного программного обеспечения, которое будет полезно при разработке крупных проектов.

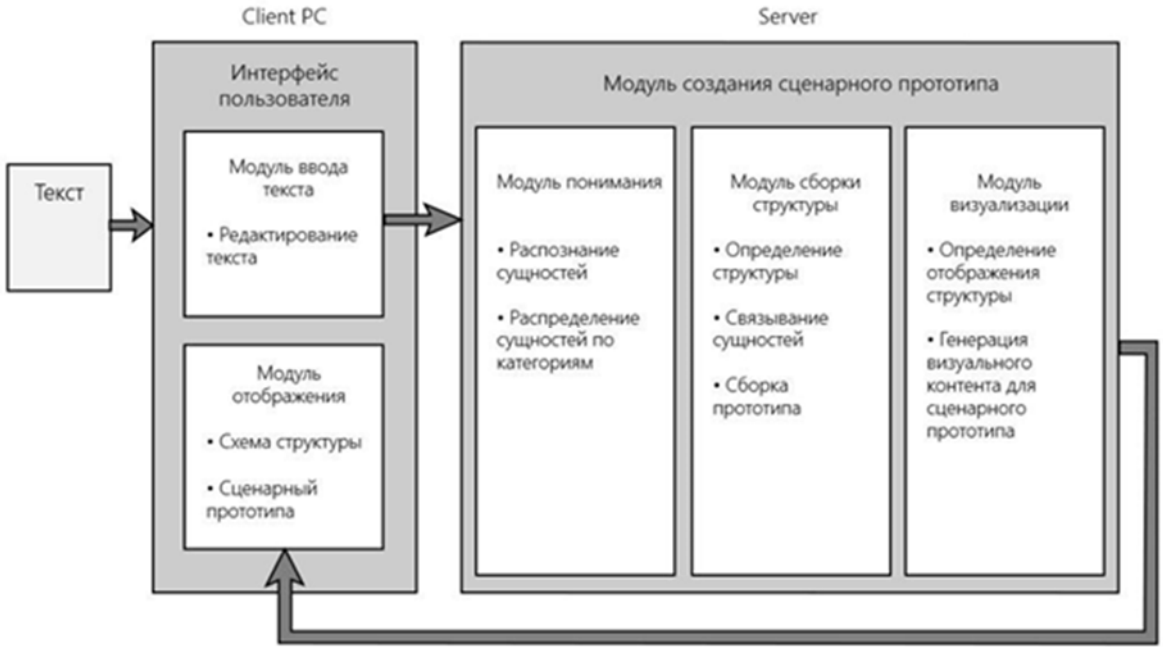


Рис. 2. Предлагаемая архитектура

ДАЛЬНЕЙШЕЕ РАЗВИТИЕ

На основе проведенного исследования была сформирована архитектура, которая обозначает векторы развития инструмента (Рис. 2). Инструмент должен будет иметь следующий функционал: принимать формализованный или неформализованный текст (из поля ввода или в виде файла); максимально точно определять и распределять сущности по категориям; определять структуру сценария (в том числе нелинейную) и отображать ее в визуальном виде (граф или Yarn); автоматически собирать сценарный прототип; добавлять в сценарный прототип визуальный контент.

Ветвление сюжета

Текущая версия прототипа поддерживает только линейный сюжет. В перспективе данный инструмент должен будет иметь возможность собирать разветвленную структуру. Данный подход может быть визуализирован в виде Yarn [9].

Генерация визуального контента

На данный момент предложено несколько вариантов подвязывания и генерации визуального контента. Инструмент SceneMaker [11] предлагает свое решение. Также полезен опыт разработки нейросети CRAFT [12].

Если внедрить данный функционал в создаваемый инструмент, то это позволит с наименьшими усилиями и временными затратами создавать визуализацию сценарного прототипа, что благоприятно скажется на тестировании повествования.

Автоматическая сборка сценарного прототипа

На данный момент работает полуавтоматическая генерация сценарного прототипа, т. е. генерация происходит с участием пользователя. В идеале сценарный прототип должен генерироваться автоматически, исходя из определенных шаблонов.

Данные для генерации уровня

Извлекаемые из текста сущности можно интерпретировать по-разному. В зависимости от контекста их можно разделить на разные категории. Одна из таких категорий – описание уровня. Пусть текст дает неполную информацию, но может задать основные правила, будь то относительное расположение улиц в городе или наличие особенного вида растений вдоль тропинок. Поэтому можно связать генерирование сценарного прототипа и создании на его основе правил для генерации уровня [14].

ЗАКЛЮЧЕНИЕ

Прототипирование и тестирование – важные этапы создания игр. Пренебрежение этими шагами может повысить риски и увеличить затраты, запланированные на проект.

Сама идея получения игрового прототипа на основе текстовых описаний, кажется трудной для реализации, но привлекательной в перспективе. Такой инструмент мог бы в комплексе решать вопрос прототипирования игр. Реализация такого инструмента также может означать повышение качества создаваемых игр, т. к. многие ошибки сценария можно будет заметить на ранних стадиях разработки.

СПИСОК ЛИТЕРАТУРЫ

1. *Lee T.* Designing game narrative. URL: http://hitboxteam.com/designing-game-narrative

2. *Ramadhan R. and Hendradjaya B.* Development of Game Testing Method for Measuring Game Quality // Proc. of Int. Conf. on Data and Software Engineering. 2014. № 7062694.

3. *Пшеничный И.* Чем отличается разработка игр от разработки «обычных» ИТ-проектов. URL: https://vc.ru/4897-gamedev-it

4. *Fullerton T., Swain C. and Hoffman S.S.* Game design workshop: a play-centric approach to creating innovative games. Burlington: Elsevier Inc., 2008. 470 p.

5. *Khatoon M., Aisha Banu W., Zohra A.A. and Chinthamani S.* Sentiment Analysis on Tweets // Advances in Intelligent Systems and Computing. 2019. V. 731. P. 717–724.

6. *Mignano M.* Game Development Pipeline: From Concept to Store. URL: http://gamedevelopertips.com/game-development-pipeline/

7. *Garneau P.-A.* Fourteen Forms of Fun. URL: https://www.gamasutra.com/view/feature/227531/fourteen_forms_of_fun/

8. *Liu S., Wu Y., Wei E., Liu M. and Liu Y.* Storyflow: Tracking the Evolution of Stories // IEEE Transactions on Visualization and Computer Graphics. 2013. No. 12. P. 2436–2445.

9. *Padia K., Bandara K.H. and Healey C.G.* Yarn: Generating Storyline Visualizations Using HTN Planning // Graphics Interface. 2018. P. 17–24.

10. *Liu Z.-Q. and Leung K.-M.* Script Visualization (ScriptViz): A Smart System that Makes Writing Fun // Soft Computing. 2006. V. 10. P. 34–40.

11. *Akser M., Bridges B., Campo G., Cheddad A., Curran K., Fitzpatrick L., Hamilton L., Harding J., Leath T., Lunney T., Lyons F., Ma M., Macrae J., Maguire T., McCaughey A., McClory E., McCollum V., Mc Kevitt P., Melvin A., Moore P., Mulholland E., Muñoz K., O'Hanlon G. and Roman L.* SceneMaker: Creative Technology for Digital Storytelling // Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering. 2016. P. 29–38.

12. *Gupta T., Schwenk D., Farhadi A., Hoiem D. and Kembhavi A.* Imagine This! Scripts to Compositions to Videos // Computer Vision and Pattern Recognition. 2018.

13. *Adams E.* The Designer's Notebook: Machinations, A New Way to Design Game Mechanics. URL: https://www.gamasutra.com/view/feature/176033/the_designers_notebook

14. *Баранов В.С., Сергеев А.С., Кугуракова В.В., Ситдиков А.Г., Хафизов М.Р.* Археологические объекты болгарского городища X–XV вв., как материал для создания виртуальной культурно-исторической реконструкции // Электронные библиотеки. 2015. Т. 18. №5. С. 269–282.

THE CONCEPT OF AUTOMATIC CREATION TOOL FOR COMPUTER GAME SCENARIO PROTOTYPE

G. F. Sahibgareeva[1], V. V. Kugurakova[2]

[1,2]*Higher School of Information Technologies and Intelligent Systems. Kazan Federal University*

[1]gulnara.sahibgareeva@gmail.com, [2]vlada.kugurakova@gmail.com

Abstract

The description of the architecture of the tool for generating a scenario prototype from the text outlined in this paper is described based on the existing solutions.

We formed software requirements and developed prototype of the tool that illustrating the basic principle of the user's work with the application.

Keywords*: game scripts, narrative design, scenario prototype, prototyping, game development, immersion, narrative*

REFERENCES

1. *Lee T.* Designing game narrative. URL: http://hitboxteam.com/designing-game-narrative

2. *Ramadhan R. and Hendradjaya B.* Development of Game Testing Method for Measuring Game Quality // Proc. of Int. Conf. on Data and Software Engineering. 2014. № 7062694.

3. *Pshenichnyj I.* CHem otlichaetsya razrabotka igr ot razrabotki «obychnyh» IT-proektov. URL: https://vc.ru/4897-gamedev-it

4. *Fullerton T., Swain C. and Hoffman S.S.* Game design workshop: a play-centric approach to creating innovative games. Burlington: Elsevier Inc., 2008. 470 p.

5. *Khatoon M., Aisha Banu W., Zohra A.A. and Chinthamani S.* Sentiment Analysis on Tweets // Advances in Intelligent Systems and Computing. 2019. V. 731. P. 717–724.

6. *Mignano M.* Game Development Pipeline: From Concept to Store. URL: http://gamedevelopertips.com/game-development-pipeline/

7. *Garneau P.-A.* Fourteen Forms of Fun. URL: https://www.gamasutra.com/view/feature/227531/fourteen_forms_of_fun/

8. *Liu S., Wu Y., Wei E., Liu M. and Liu Y.* Storyflow: Tracking the Evolution of Stories // IEEE Transactions on Visualization and Computer Graphics. 2013. No. 12. P. 2436–2445.

9. *Padia K., Bandara K.H. and Healey C.G.* Yarn: Generating Storyline Visualizations Using HTN Planning // Graphics Interface. 2018. P. 17–24.

10. *Liu Z.-Q. and Leung K.-M.* Script Visualization (ScriptViz): A Smart System that Makes Writing Fun // Soft Computing. 2006. V. 10. P. 34–40.

11. *Akser M., Bridges B., Campo G., Cheddad A., Curran K., Fitzpatrick L., Hamilton L., Harding J., Leath T., Lunney T., Lyons F., Ma M., Macrae J., Maguire T., McCaughey A., McClory E., McCollum V., Mc Kevitt P., Melvin A., Moore P., Mulholland E., Muñoz K., O'Hanlon G. and Roman L.* SceneMaker: Creative Technology for Digital Storytelling // Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering. 2016. P. 29–38.

12. *Gupta T., Schwenk D., Farhadi A., Hoiem D. and Kembhavi A.* Imagine This! Scripts to Compositions to Videos // Computer Vision and Pattern Recognition. 2018.

13. *Adams E.* The Designer's Notebook: Machinations, A New Way to Design Game Mechanics. URL: https://www.gamasutra.com/view/feature/176033/the_designers_notebook

14. *Baranov V.S., Sergeev A.S., Kugurakova V.V., Sitdikov A.G., Khafizov M.R.* Arheologicheskie ob"ekty bolgarskogo gorodishcha X–XV vv., kak material dlya sozdaniya virtual'noj kul'turno-istoricheskoj rekonstrukcii // Elektronnye biblioteki. 2015. T. 18. №5. S. 269–282.

СВЕДЕНИЯ ОБ АВТОРАХ

САХИБГАРЕЕВА Гульнара Фаритовна – бакалавр Высшей школы информационных технологий и интеллектуальных систем (ВШ ИТИС) Казанского (Приволжского) федерального университета (КФУ). Сфера интересов: игровая сценаристика, нарративный дизайн, изучение вопроса эффективности создания сценарного прототипа и возможности автоматизировать данный процесс.

Gulnara Faritovna SAHIBGAREEVA – Bachelor of Higher School ITIS. Sphere of interests: game scripts, narrative design, studying the effectiveness of creating a scenario prototype and the ability to automate this process.

email: gulnara.sahibgareeva42@gmail.com

КУГУРАКОВА Влада Владимировна – старший преподаватель ВШ ИТИС КФУ, руководитель лаборатории «SIM». Сфера интересов: разработка игр, иммерсивность виртуальных сред, интерпретация биосигналов человека, погруженного в виртуальную среду, методы трансформации нарративной идентичности.

Vlada Vladimirovna KUGURAKOVA – Senior Lecturer of Higher School of Information Technology and Intelligent Systems, Head of Laboratory «SIM». Sphere of interests: game development, immersivity of virtual environment, narrative identity.

email: vlada.kugurakova@gmail.com

Материал поступил в редакцию 1 июля 2018 года.

УДК 004.415.25+004.42+004.514

АЛГОРИТМ ГЕНЕРАЦИИ КОДА ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА МОБИЛЬНЫХ ПРИЛОЖЕНИЙ НА ОСНОВЕ ДАННЫХ ГРАФИЧЕСКОГО РЕДАКТОРА

А. Ю. Усачёв

Высшая школа информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета

usacheow.ar@gmail.com

Аннотация

Работа посвящена разработке алгоритма генерации кода пользовательских интерфейсов нативных Android-приложений на основе данных графического редактора. Рассмотрена проблема негативного воздействия на время разработки продукта выполнения шаблонных действий и предложен программный инструмент для решения данной проблемы.

Ключевые слова: *пользовательский интерфейс, графический редактор, алгоритм генерации, мобильные приложения*

ВВЕДЕНИЕ

Длительность процесса разработки мобильных приложений напрямую влияет на сроки разработки и итоговую стоимость конечного продукта. Ниже приведены работы, выполнение которых занимает наибольшее время [1]:

- анализ требований и формулирование технического задания;
- разработка дизайна и пользовательского интерфейса;
- вёрстка макетов пользовательского интерфейса;
- написание кода и тестирование.

Автоматизация этапов жизненного цикла разработки мобильных приложений, требующих выполнения шаблонных действий, позволит значительно сократить затраты на создание программного продукта.

На текущий момент процесс создания пользовательского интерфейса требует написания программного кода для каждого элемента интерфейса по предоставленному дизайну экранов разрабатываемого мобильного приложе-

ния. Согласно официальному сайту программы Sketch [2], большинство мобильных дизайнеров выполняет свою работу в данном графическом редакторе. Файл программного инструмента Skecth может быть представлен в виде набора JSON-файлов, в которых содержится вся информация о дизайне экранов приложения, что делает возможной автоматизацию данного процесса путем анализа и обработки полученных JSON-файлов.

Разработка программного инструмента, автоматизирующего процесс разработки пользовательского интерфейса, сопровождается решением следующих задач:

- определение перечня элементов, которые поддаются генерации программой, и разработка требований к структуре sketch-проекта;
- разработка алгоритма для генерации кода пользовательского интерфейса на основе данных графического редактора;
- разработка графического интерфейса для взаимодействия пользователя с программой.

АНАЛИЗ ИЗВЕСТНЫХ РЕШЕНИЙ ПРОБЛЕМЫ

На данный момент есть два программных инструмента, способных на генерацию элементов интерфейса.

Первый инструмент носит название Zeplin, он является связующим звеном между дизайнером и программистом, позволяя импортировать sketch-проект для представления его в таком виде, который позволит сократить время разработчика на взаимодействие с данным проектом [3]. Кроме того, Zeplin поддерживает автоматическую генерацию одного элемента интерфейса – текстового поля, в то время как пользовательский интерфейс может состоять из множества различных элементов.

Вторым инструментом является продукт под названием Supernova [4], к плюсам которого можно отнести интуитивно понятный пользовательский интерфейс. Однако все элементы, генерируемые данной программой, состоят исключительно из текстового поля и картинки. Отсюда вытекает тот факт, что данное решение подходит только для генерации макетов в тех случаях, когда требуется представить макеты пользовательских интерфейсов на мобильном устройстве, а для дальнейшей работы они не подходят.

Следовательно, можно сделать вывод, что программного инструмента, который обладал бы возможностью генерации верстки макета пользовательского интерфейса в полном объёме, не существует.

ФОРМУЛИРОВАНИЕ ТРЕБОВАНИЙ К РЕШЕНИЮ

При составлении технических требований было решено определить начальную группу элементов интерфейса android-приложений, которые будут определяться на экране интерфейса и на основе которых будет составляться файл макета в формате xml, требуемом для использования в android-проектах. Таким образом, было решено взять основные и наиболее часто используемые элементы [5]:

- *Toolbar* – это панель, расположенная в верхней части экрана, на которой отображается заголовок данного экрана либо логотип приложения и кнопки действий, например, открытие меню, переход назад или какие-либо специфичные для данного экрана действия (Рис. 1);

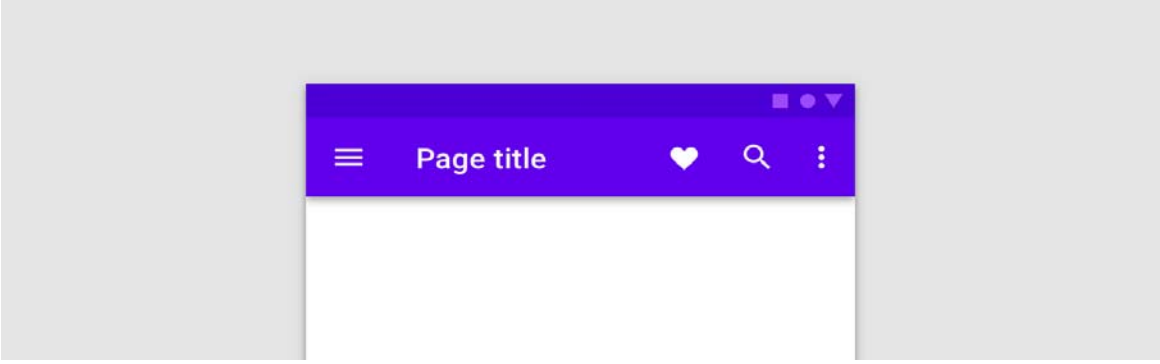


Рис. 1. Toolbar

- *TabLayout* – это элемент, на котором расположены вкладки данного экрана, которые могут быть представлены в виде текста и/или иконки (Рис. 2);
- *AppBarLayout* – элемент интерфейса, схожий с LinearLayout, использующийся для задания поведения дочерних элементов при прокручивании экрана указанием дополнительных флагов;
- *ScrollView* – элемент, использующийся для прокрутки содержимого. Может иметь только один дочерний элемент;
- *LinearLayout* – это диспетчер компоновки, который позволяет располагать внутри себя элементы последовательно друг за другом (вертикально либо горизонтально), указав соответствующий флаг в атрибутах (Рис. 3);

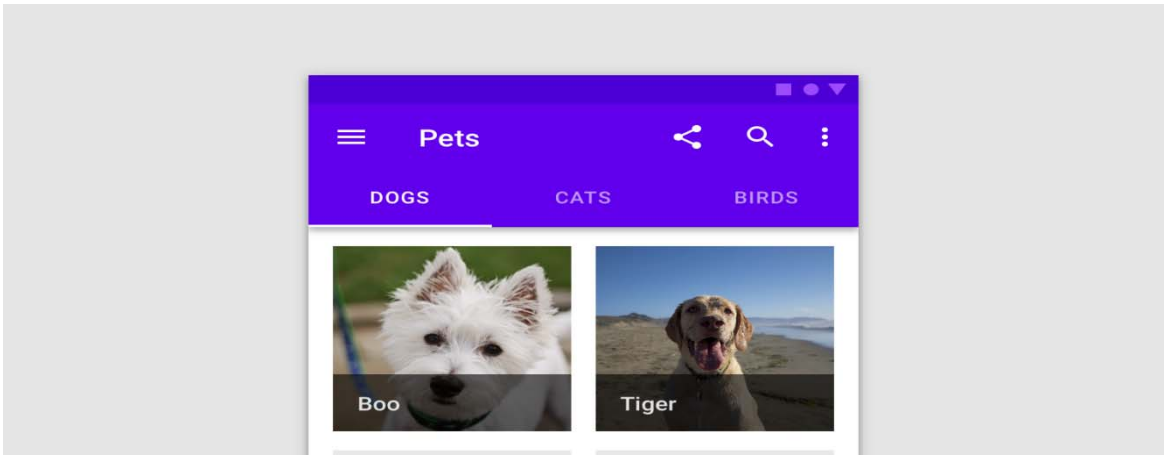


Рис. 2. TabLayout

Рис. 3. LinearLayout: горизонтальный и вертикальный

- *CardView* – элемент, предназначенный для отображения содержимого в виде карточки с заданной высотой и радиусом закругления углов (Рис. 4);

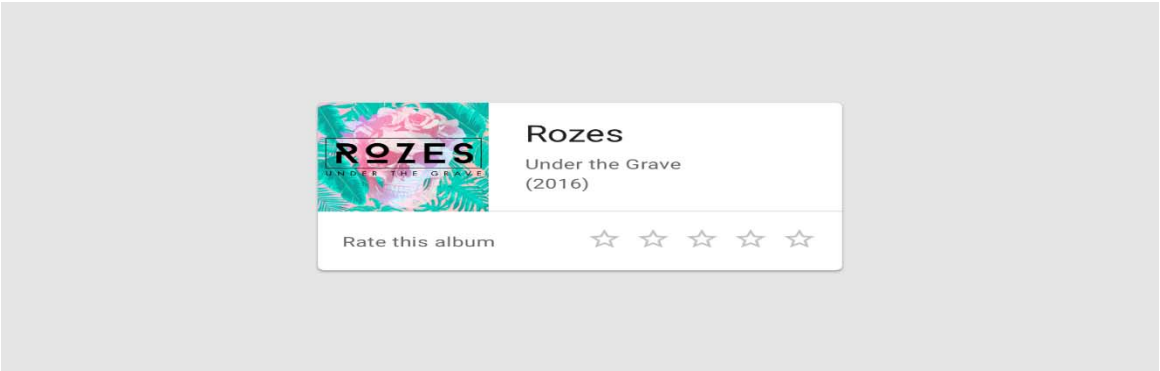


Рис. 4. CardView

- RecyclerView – это элемент, который служит для отображения списка данных (Рис. 5). Может быть вертикальным или горизонтальным, что определяется соответствующим значением в атрибутах. Элементы могут быть различны-

ми, благодаря чему можно разделить список на группы, отображая заголовок для каждой из них;

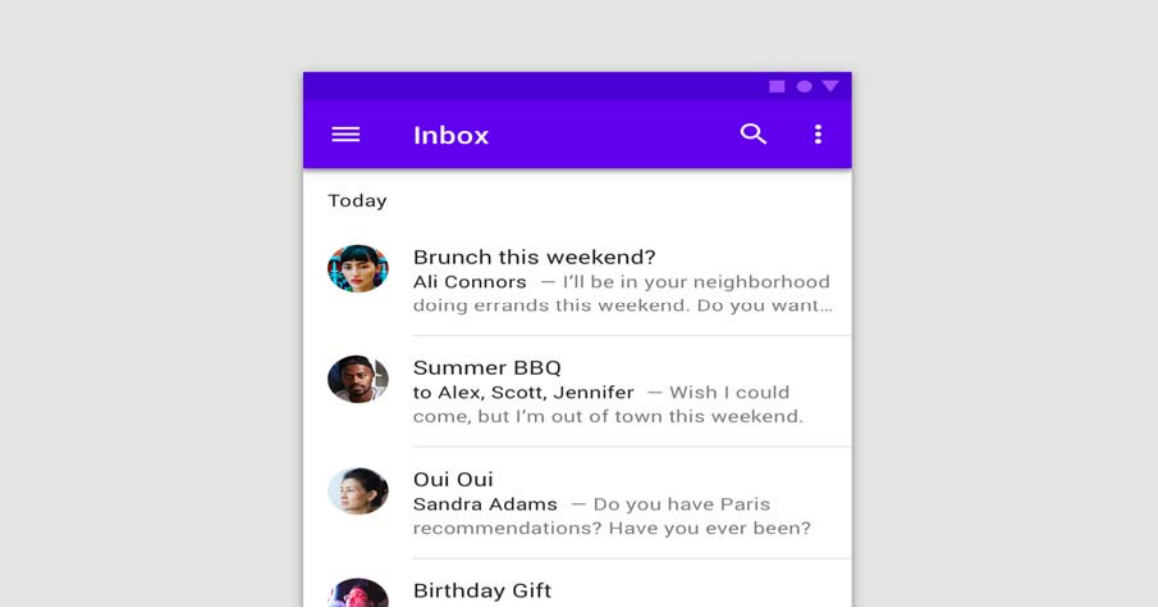


Рис. 5. RecyclerView

- SwipeRefreshLayout – это контейнер, который может содержать только один дочерний элемент (Рис. 6). Используется для отображения обновляемого списка элементов;

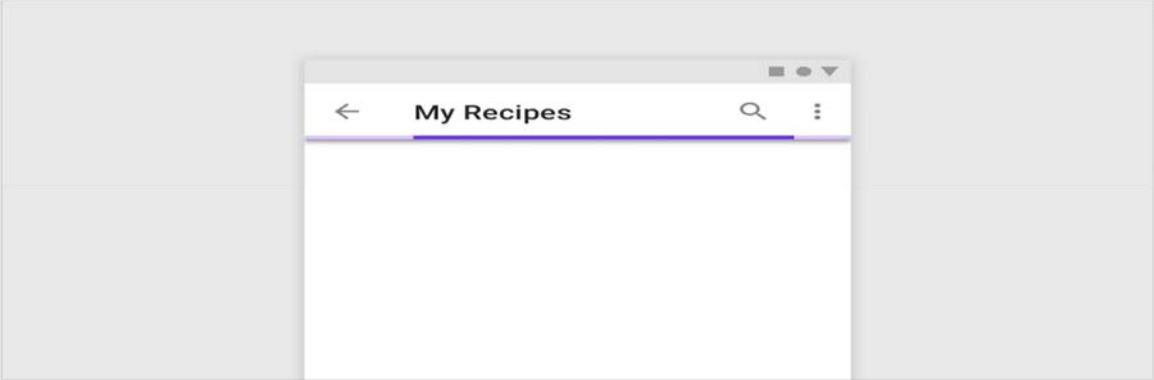


Рис. 6. SwipeRefreshLayout

- ViewPager – это контейнер, который содержит внутри себя несколько страниц. Данный компонент используется в связке с компонентом TabLayout: вместе они дают возможность пролистывать экраны с вкладками;
- Button – это кнопка, которая может содержать текст и реагировать на нажатия (Рис. 7);

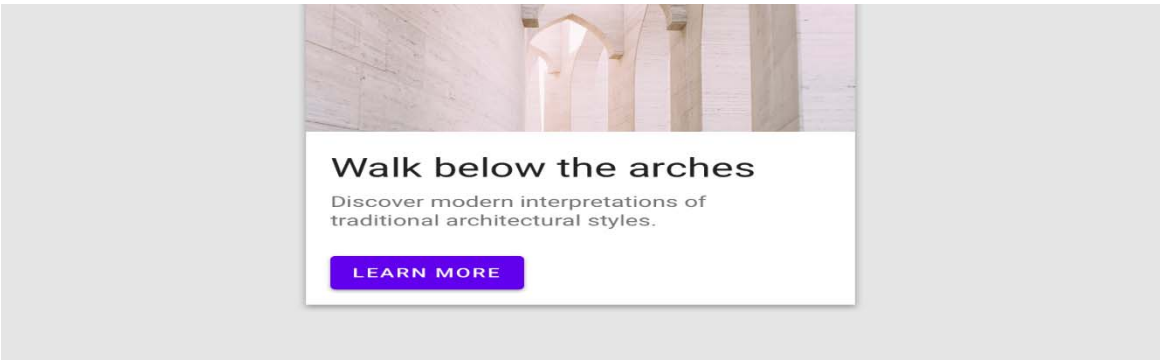


Рис. 7. Кнопка

- EditText – это поле для ввода текста, которое может быть в двух состояниях: активном и неактивном (Рис. 8). Может содержать подсказку о требуемой информации. В зависимости от указанного параметра в атрибутах может позволять ввод ограниченных групп символов;

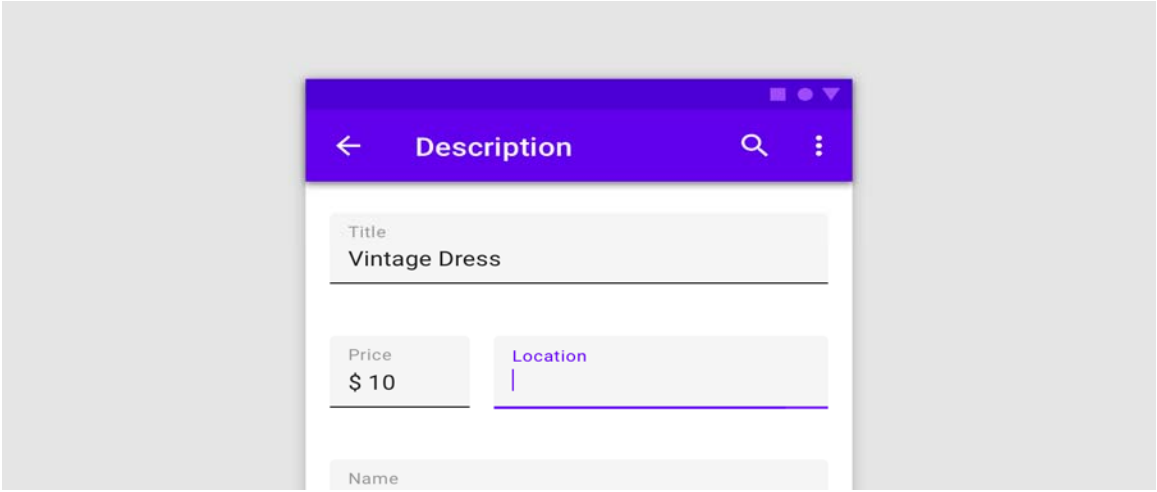


Рис. 8. EditText: активный и неактивный

- TextView – это текстовое поле, которое может содержать текст и иконку. В зависимости от параметра в атрибутах содержимое поля может находиться в разных положениях, например, в центре, справа, сверху;
- ImageView – контейнер для изображений.

Далее были сформулированы требования к наименованию элементов в sketch-проекте. Для этого было решено использовать обобщенные названия, так как зачастую дизайн пользовательских интерфейсов для приложений различных операционных систем делается вместе, а названия для одинаковых элементов могут сильно различаться в зависимости от требуемой платформы [6].

При составлении требований к структуре учитывался тот факт, что дизайнеры зачастую не знают всех тонкостей в вёрстке макетов экранов под разные платформы, поэтому в первую очередь преследовались следующие цели:

- избежать особенностей расположения элементов для разных платформ;
- не перегрузить проект большим количеством различных папок, но сделать структуру интуитивно понятной для программиста.

Таким образом, были приняты следующие варианты:

- Toolbar – папка с названием header. В ней может располагаться текст с заголовком экрана;
- TabLayout – папка с названием tabs;
- AppBarLayout – папка с названием sheader. В ней должны располагаться две папки: header и tabs, то есть Toolbar и TabLayout соответственно;
- ScrollView – папка с названием slayout. Может содержать любые элементы из списка перечисленных в технических требованиях;
- LinearLayout – папка с названием vlayout либо hlayout, что соответствует вертикальной либо горизонтальной компоновке дочерних элементов соответственно. Может содержать любые элементы из списка перечисленных в технических требованиях;
- CardView – папка с названием card. Может содержать один дочерний элемент;
- RecyclerView – папка с названием list. Может содержать один любой элемент из списка, указанного в технических требованиях. Если у папки указать имя rlist, то сгенерируется RecyclerVIew, обернутый в SwipeRefreshLayout. Такая папка аналогично может содержать любой элемент из списка, указанного в технических требованиях;
- ViewPager – папка с названием pager. Может содержать любой элемент из списка, указанного в технических требованиях.
- Button – папка с названием button. Может содержать текст с названием кнопки и стиль;
- EditText – папка с названием input. Может содержать текст с подсказкой, иконку и стиль;

- TextView – текст, в качестве имени которого указано содержимое текстового поля;
- ImageView – изображение с произвольным именем;
- Background - стиль для элементов EditText, Button и LinearLayout. Должен располагаться в корне элемента, к которому относится.

ПРИНЦИП РАБОТЫ АЛГОРИТМА

За основу алгоритма был выбран обход в глубину: из каждого json-файла, полученного из исходного sketch-проекта, строится дерево элементов, а затем обходом по данному дереву генерируются итоговые макеты экранов интерфейса. В итоге для каждого экрана на входе имеется одно дерево элементов, а на выходе – два: для экранов и для стилей, которые задают цвет, фон и форму элементов на экране (Рис. 9).

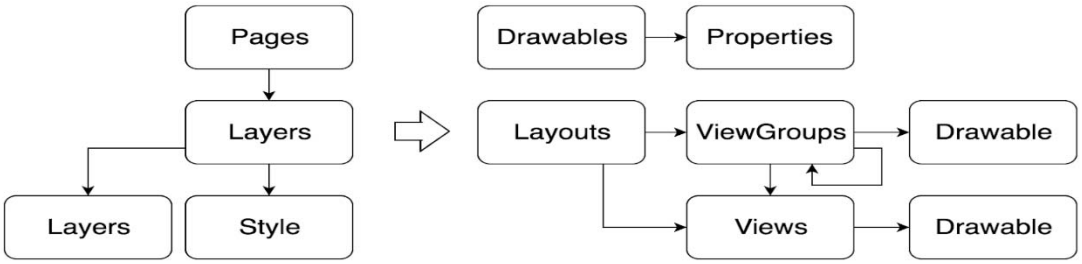


Рис. 9. Входные и выходные данные

Алгоритм генерации макетов интерфейса состоит из следующих шагов:

- распаковка файла проекта и сохранение полученных файлов в памяти устройства;
- обработка полученных json-файлов и генерация дерева всех элементов;
- генерация стилей и общих элементов для всех экранов;
- генерация макетов всех экранов;
- генерация всех итоговых файлов в памяти устройства;
- удаление вспомогательных файлов, созданных при работе программы, и перевод алгоритма в начальное состояние.

Упрощённая блок-схема данного алгоритма представлена на рис. 10 и 11.

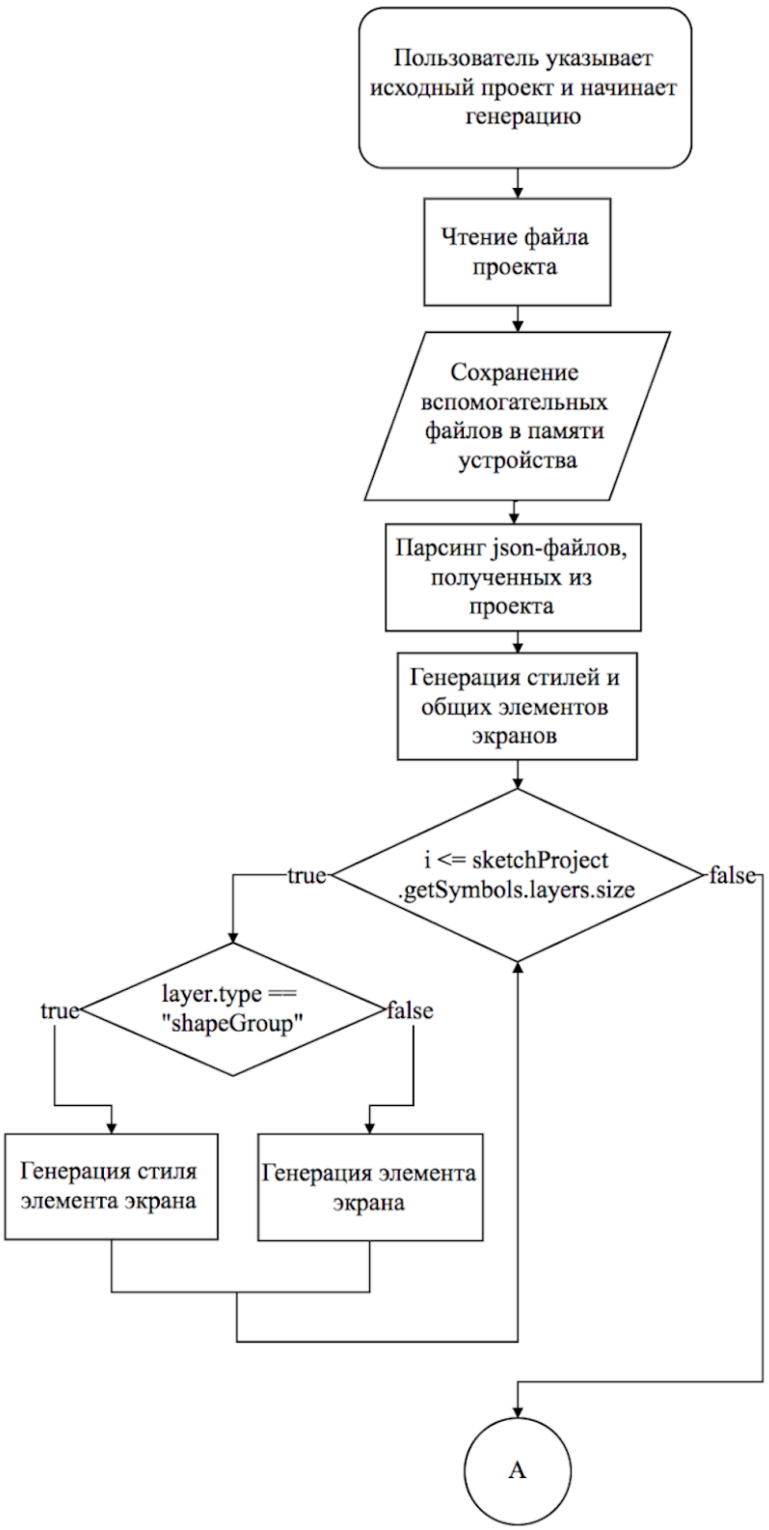


Рис. 10. Упрощённая блок-схема алгоритма (часть 1)

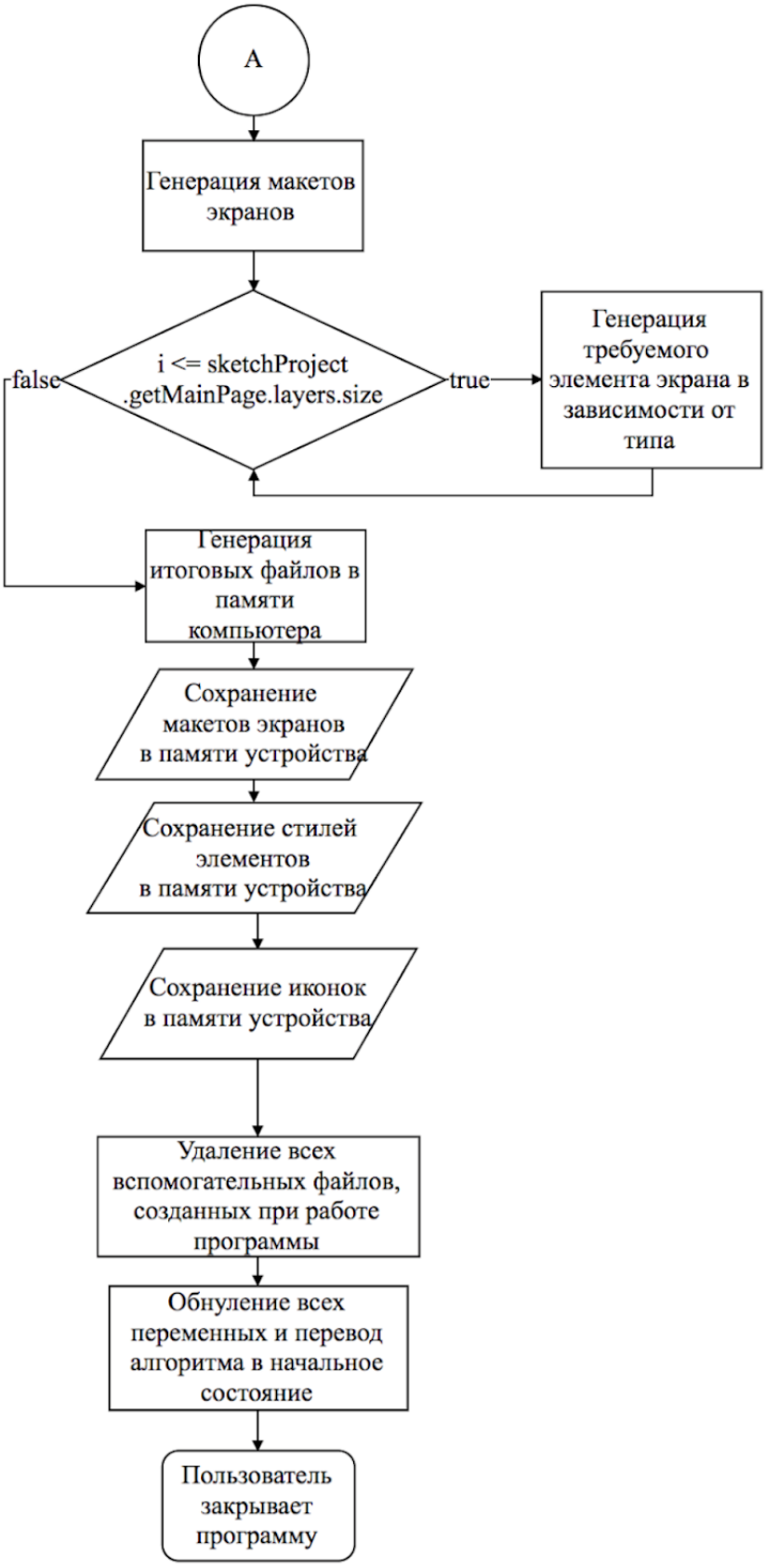


Рис. 11. Упрощённая блок-схема алгоритма (часть 2)

ПРИНЦИП РАБОТЫ ПРОГРАММНОГО РЕШЕНИЯ

Пользователь взаимодействует с программой посредством графического интерфейса (Рис. 12).

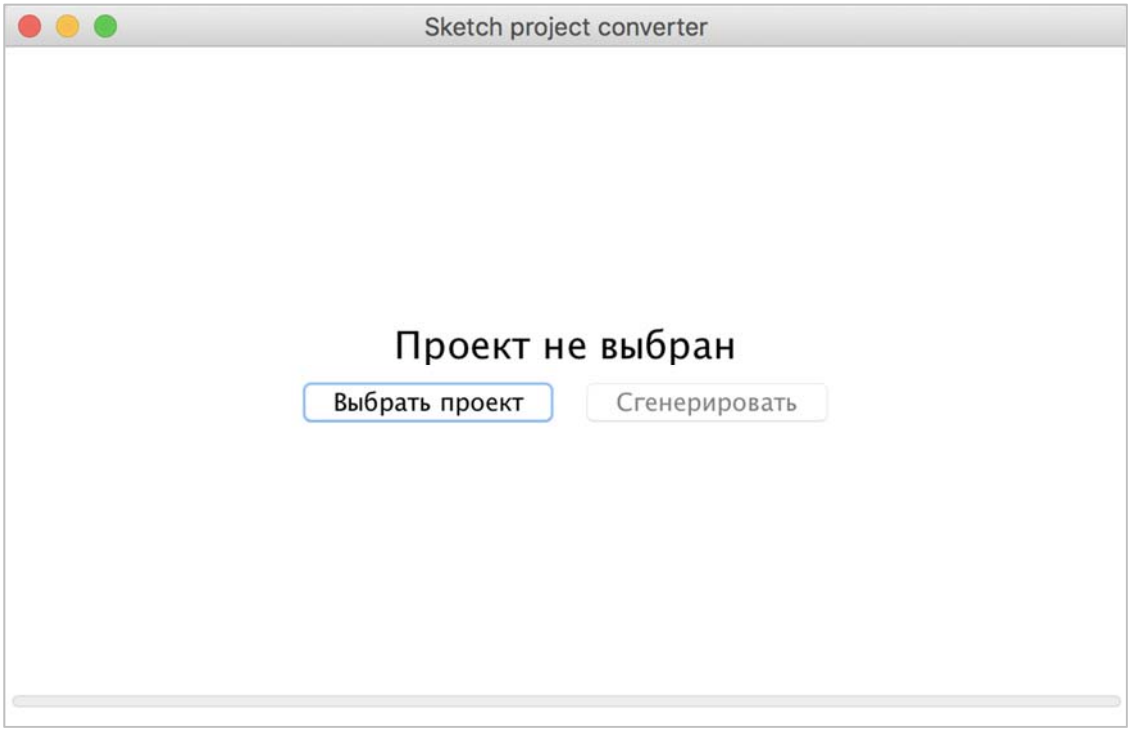


Рис. 12. Пользовательский интерфейс программы

После выбора проекта сверху над кнопкой отобразится название выбранного пользователем проекта. Если нажать на кнопку «Сгенерировать», то начнётся работа алгоритма. Когда все требуемые макеты будут сгенерированы, полоса прогресса заполнится на сто процентов, а под названием проекта отобразится количество сгенерированных макетов экранов и количество сгенерированных дополнительных макетов (Рис. 13), например, дочерние элементы для RecyclerView, страницы для ViewPager. Сгенерированные макеты экранов сохранятся в папке рядом с выбранным sketch-проектом.

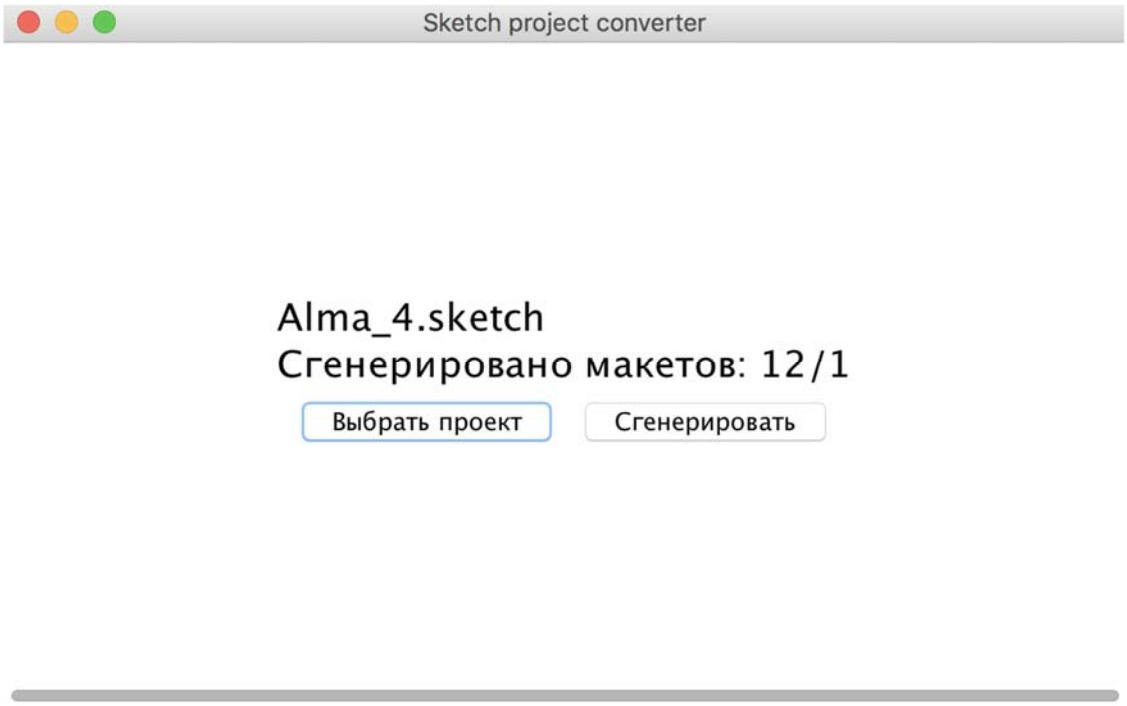


Рис. 13. Иерархия классов моделей

По исходному JSON-файлу будет сгенерирован следующий код:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  android:background="@drawable/bg"
  android:gravity="center_horizontal"
  android:layout_height="match_parent"
  android:layout_width="match_parent"
  android:orientation="vertical">
  <LinearLayout
    android:gravity="center_horizontal"
    android:layout_height="0dp"
    android:layout_weight="538"
    android:layout_width="321dp"
    android:orientation="vertical">
    <ImageView
```

```
            android:layout_height="0dp"
            android:layout_weight="254"
            android:layout_width="299dp"
            android:scaleType="centerCrop"
            app:srcCompat="@drawable/_logo" />
        <TextView
            android:gravity="center_horizontal"
            android:layout_height="0dp"
            android:layout_marginBottom="28dp"
            android:layout_weight="30"
            android:layout_width="match_parent"
            android:text="ALMA"
            android:textColor="#ff5c5c5c" />
        <LinearLayout
            android:gravity="center"
            android:layout_height="0dp"
            android:layout_marginBottom="56dp"
            android:layout_weight="89"
            android:layout_width="281dp"
            android:orientation="vertical">
    <EditText                                                    an-
droid:background="@drawable/inputc2593e28_62f5_4107_bb62"
            android:drawablePadding="4dp"
            android:drawableStart="@drawable/ic_phone"
            android:hint="Номер телефона"
            android:layout_height="0dp"
            android:layout_weight="45"
            android:layout_width="281dp"
            android:padding="8dp"
            android:textColor="#ff6bcbd9"
            android:textColorHint="#ff6bcbd9" />
    <EditText  android:background="@drawable/input1b1d12af_b927_49f3_8bc1"
            android:drawablePadding="4dp"
```

```
            android:drawableStart="@drawable/ic_password"
            android:hint="Пароль"
            android:layout_height="0dp"
            android:layout_weight="45"
            android:layout_width="281dp"
            android:padding="8dp"
            android:textColor="#ff6bcbd9"
            android:textColorHint="#ff6bcbd9" />
        </LinearLayout>
    <Button                                                    an-
droid:background="@drawable/buttonb04c99c4_f455_4cb5_9da9"
            android:layout_height="0dp"
            android:layout_marginBottom="19dp"
            android:layout_weight="48"
            android:layout_width="281dp"
            android:text="ВОЙТИ"
            android:textColor="#ffffffff" />
        <TextView
            android:gravity="center_horizontal"
            android:layout_height="wrap_content"
            android:layout_marginBottom="15dp"
            android:layout_width="match_parent"
            android:text="Создать аккаунт"
            android:textColor="#ff6e6e6e" />
      </LinearLayout>
    </LinearLayout>
```

Пример сгенерированного макета представлен на Рис. 14. Можно заметить, что макет экрана генерируется достаточно точно, но есть небольшие отличия в тексте, которые связаны с техническими особенностями хранения информации о нём в исходных данных sketch-проекта. Также вместо картинки отображается закрашенный квадрат, что связано с тем, что в исходном проекте хранится много изображений, среди которых присутствуют изображения по умолча-

нию, отделение которых от основных графических файлов накладывает дополнительный ряд ограничений к структуре sketch-проекта. Так как основная масса изображений в мобильных приложениях загружается из интернета, было решено отказаться от данной функции.

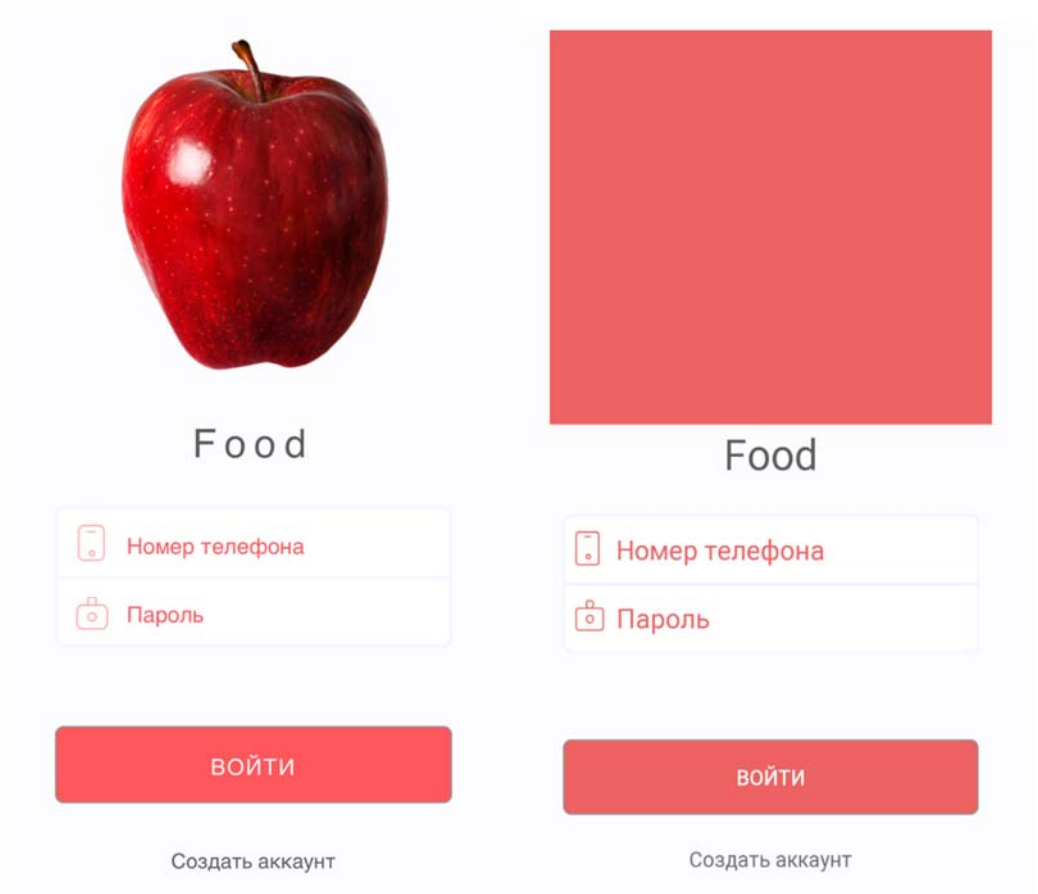


Рис. 14. Макеты: исходный и сгенерированный программой

ЗАКЛЮЧЕНИЕ

Разработано программное решение для генерации кода пользовательского интерфейса нативных мобильных приложений для операционной системы Android. Разработанный программный продукт обладает возможностью генерации пользовательских интерфейсов мобильных приложений по данным графического редактора.

Задачи, выполненные в рамках работы:

- разработаны технические требования и требования к структуре sketch-проекта;
- разработан алгоритм генерации кода макетов UI по данным графического редактора;

- создан программный инструмент, реализующий генерацию файлов пользовательских интерфейсов для дальнейшего использования в Android-проектах.

Разработанный программный инструмент позволит Android-разработчикам сократить время на создание пользовательских интерфейсов по макетам графического редактора, что, в свою очередь, сократит общее время разработки программного продукта для операционной системы Android.

В дальнейшем планируется расширить список генерируемых элементов, добавить генерацию паттернов MVP/MVVM и опубликовать проект в открытом доступе, что даст возможность сторонним разработчикам кастомизировать алгоритм согласно своим требованиям.

СПИСОК ЛИТЕРАТУРЫ

1. Введение в жизненный цикл разработки мобильных приложений [Электронный ресурс]. URL: https://codedocs.ru/xamarin/vvedenie-v-zhiznennyj-tsikl-razrabotki-mobilnyh-prilozhenij.html

2. Sketch [Электронный ресурс]. URL: https://sketchapp.com

3. Zeplin [Электронный ресурс]. URL: https://zeplin.io/

4. Supernova [Электронный ресурс]. URL: https://supernova.studio/

5. Material Design [Электронный ресурс]/ URL: https://material.io/ guidelines/

6. Human Interface Guidelines [Электронный ресурс]. URL: https://developer. apple.com/ios/human-interface-guidelines/overview/themes/

7. Android Studio [Электронный ресурс]. URL: https://developer.android.com/ studio/projects/index.html

ALGORITHM FOR GENERATION OF MOBILE APPLICATIONS' UI CODE BASED ON DATA OF GRAPHIC EDITOR

A. Usachev

Higher School of Information Technologies and Intelligent Systems at Kazan Federal University

usacheow.ar@gmail.com

Abstract

The paper is devoted to the development of the algorithm for generating the code of user interfaces for native Android applications based on the data of the graphical editor. The problem of negative impact on the development time of the product for performing routine actions is considered, and a software tool for solving this problem is proposed.

Keywords: *user interface, graphic editor, generation algorithm*

REFERENCES

1. CodeDocs site. URL: https://codedocs.ru/xamarin /vvedenie-v-zhiznennyj-tsikl-razrabotki-mobilnyh-prilozhenij.html
2. Official site of the program Sketch. URL: https://sketchapp.com
3. Official site of the program Zeplin. URL: https://zeplin.io/
4. Official site of the program Supernova. URL: https://supernova.studio/
5. Official site with guidelines for Material Design. URL: https://material. io/guidelines/
6. Official site with guidelines for Human Interface. URL: https://developer.apple.com/ios/human-interface-guidelines/overview/themes/
7. Official site for Android developer. URL: https://developer. android.com/studio/projects/index.html

СВЕДЕНИЯ ОБ АВТОРЕ

УСАЧЁВ Артемий Юрьевич – студент Высшей школы информационных технологий и интеллектуальных систем Казанского федерального университета, android разработчик.

Artemy Yurievich USACHEV – student of the Higher School of Information Technologies and Intelligent Systems of Kazan Federal University, middle developer of software for Android.

email: usacheow.ar@gmail.com

Материал поступил в редакцию 10 июня 2018 года

УДК 004.42+004.75

ЦИФРОВОЙ ПАСПОРТ КАРЬЕРНОЙ ТРАЕКТОРИИ, ОСНОВАННЫЙ НА ТЕХНОЛОГИИ РАСПРЕДЕЛЕННЫХ РЕЕСТРОВ

А. И. Шайфутдинов[1], А. Ф. Хасьянов[2]

Высшая школа информационных технологий и интеллектуальных систем, Казанский (Приволжский) федеральный университет

[1]aidarshaifutdinov@gmail.com, [2]ak@it.kfu.ru

Аннотация

Рассмотрены проблемы, связанные с документальным сопровождением процесса трудоустройства и фиксации трудового стажа. Сегодня эти задачи решаются через бумажные контракты и, в Российской Федерации, посредством «трудовых книжек». Предлагается заменить существующий бумажный документооборот программным решением, основанным на технологии распределенных реестров (блокчейн) и смарт-контрактах.

***Ключевые слова**: трудовые отношения, трудовой стаж, бумажный документооборот, блокчейн, смарт-контракты, цифровизация, децентрализованные приложения, Ethereum, Solidity, IPFS*

ВВЕДЕНИЕ

На сегодняшний день в процессах, связанных с трудоустройством, существуют сложности, с которыми приходится сталкиваться как работодателям, так и соискателям. Основные проблемы таковы:

1. Существующие технологии бумажного документооборота не эффективны, не эргономичны, не прозрачны и плохо масштабируемы [1], вследствие чего компании несут материальные издержки, а работники несут на себе риски, связанные с утерей документов, подтверждающих их трудовой стаж [2, 3].

2. По данным CareerBuilder, на сентябрь 2017 года более 58% соискателей размещают в резюме недостоверные данные о себе [4]. При этом не существует технологии, которая бы позволяла работодателям эффективно получать достоверные данные о карьерной траектории и достижениях соискателей. Как

следствие, HR-специалистам приходится проверять прошлые места работы, рекомендации и другие данные, тратя при этом время и деньги компании.

Для решения проблем, описанных выше, была поставлена цель оцифровать трудовые отношения, разработав децентрализованный программный продукт, основанный на технологии распределенных реестров (блокчейн), который предоставляет работодателям и соискателям возможность взаимодействовать и вести трудовые отношения друг с другом напрямую в цифровом формате. Помимо очевидных преимуществ цифровизации, среди которых: простота использования и поиска, скорость и эффективность работы, доступность, эргономичность, масштабируемость и безопасность, использование блокчейна позволяет добиться качественного прогресса по сравнению с традиционными централизованными электронными системам, а именно:

1. Обеспечивается прозрачность процесса трудоустройства и карьерной траектории каждого человека благодаря тому, что любая запись (например, о том, что человек устроился на работу в компанию) подтверждается обеими сторонами при помощи криптографической системы с открытым ключом (англ. public-key cryptography).
2. Обеспечиваются неизменяемость и надежность хранения данных, так как они распределены между всеми участниками сети и, следовательно, в системе отсутствует единая точка отказа (англ. single point of failure).

Для достижения поставленной цели было необходимо решить следующие задачи:

- идентификация пользователей;
- фиксация трудовых отношений в блокчейне с последующей возможностью подтверждения трудового стажа;
- обеспечение конфиденциальности данных о трудовых отношениях компании и работника: размер заработной платы, различные детали и условия, описанные в договоре или контракте.

ОБЗОР СУЩЕСТВУЮЩИХ ИССЛЕДОВАНИЙ

Для рынка труда всегда были актуальны вопросы достоверности данных и портфолио людей. С появлением и началом повсеместных попыток применения

технологии блокчейн мы видим все больше различных исследований и решений в этой области, некоторые из которых рассмотрим подробнее.

Например, в [5] описана концепция единой децентрализованной платформы для профессионалов в различных областях, которая бы «верифицировала опыт и достижения соискателей, позволяя им монетизировать свои навыки, и помогала компаниям в вопросах подбора персонала, предоставляя достоверную информацию о соискателях». Этот проект использует платформу Ethereum. Для вознаграждения талантливых соискателей предлагается использовать специальный токен. Недостатком данного решения является то, что верификация опыта и достижений соискателей производится самими пользователями системы постфактум на основе репутационной модели, что не дает объективных гарантий достоверности информации.

В [6] описана концепция системы хранения и управления персональными портфолио субъектов, основанная на блокчейне. В портфолио могут входить любые личные данные – от удостоверения личности до диплома о высшем образовании; эти данные представляются в виде персональных цифровых артефактов (англ. personal digital artifact). Определяются следующие роли:

- субъект – владелец своего портфолио;
- сертифицирующие сущности – институты, подтверждающие определенные записи в портфолио субъекта (например, университет или работодатель);
- клиенты – компании, нуждающиеся в подтверждении данных о субъекте, которые пользуются услугами сертифицирующих сущностей.

Чтобы получить цифровой сертификат, например, об образовании, субъект должен направить запрос своему высшему учебному заведению для подтверждения факта обучения и выпуска сертификата. Преимуществом данной концепции является то, что для непосредственного внесения записей в блокчейн предлагается использовать алгоритм консенсуса Delegated Proof of Stake, который является более эффективным с точки зрения потребляемой энергии, чем Proof of Work.

В данной работе предложена практическая реализация описанной концепции со следующими модификациями:

- роль клиента совпадает с ролью сертифицирующей сущности;

- вместо Delegated Proof of Stake используется Proof of Work.

Попытка решить проблемы, связанные с временными рабочими контрактами, при помощи технологии блокчейн предпринята в [7]. Всю совокупность трудовых взаимоотношений работодателей и соискателей, начиная с публикации вакансии и заканчивая оплатой труда, предлагается инкапсулировать в смарт-контрактах (англ. smart contract). Такие контракты могут представлять собой любую совокупность логических сценариев, представленных жестко запрограммированным алгоритмом и не подверженных изменениям. Такой подход должен помочь, с одной стороны, защитить права временных работников, а, с другой стороны, позволить работодателям гибко подстраиваться под изменчивые требования бизнеса, упростив процесс найма временных сотрудников. В рамках данной концепции предполагается наличие некой центральной сущности (англ. central authority), которая занимается авторизацией сторон и контролирует соблюдение законодательства. С одной стороны, это противоречит принципам децентрализованности, но в то же время может являться неким компромиссом между традиционными централизованными решениями и новым децентрализованным подходом.

Сравнение традиционных централизованных информационных систем, основанных на клиент-серверной архитектуре, и децентрализованных систем, основанных на технологии блокчейн, проведено в [8], где рассмотрены проблемы, связанные с достоверностью и подлинностью информации о людских ресурсах. К преимуществам децентрализованных решений относят: распределенное хранение данных, использование криптосистемы с открытым ключом для шифрования данных, открытость и неизменяемость. В статье также описана концепция системы управления данными о людских ресурсах на блокчейне. К ее недостаткам можно отнести использование закрытого блокчейна (англ. permissioned blockchain), где валидаторами могут выступать только заранее выбранные организации; то есть при определенных обстоятельствах права работников могут быть ущемлены в интересах организаций-валидаторов.

Одним из многообещающих сценариев использования технологии блокчейн являются фиксация и подтверждение академических достижений людей

(дипломы, сертификаты, информация о публикациях и научном вкладе) [9]. Возможность такого применения названной технологии рассмотрена в [10].

Holberton School – первая в мире школа, которая выдает своим выпускникам академические сертификаты, подкрепленные записью в блокчейне Bitcoin [11]. Такие записи являются публичным доказательством существования и подлинности соответствующих сертификатов.

Массачусетский технологический институт использовал блокчейн Bitcoin для выдачи цифровых версий дипломов своим выпускникам [12].

Sony Global Education планирует использовать технологию блокчейн для открытого и защищенного обмена академическими достижениями людей [13]. Компания запатентовала прототип платформы для управления образовательными портфолио [14].

Стоит также упомянуть о возможности использования технологии блокчейн для идентификации пользователей. Одним из передовых проектов в этой области является цифровая платформа идентификации Civic, реализованная на блокчейне Bitcoin [15]. Команда проекта решает задачу перехода от централизованной модели хранения персональных данных (англ. personally identifying information) к децентрализованной. Пользователь вносит различную информацию о себе, которая в зашифрованном виде хранится на его телефоне в виде дерева Меркла (англ. Merkle tree), а корневой хеш этого дерева записывается в блокчейн. Для подтверждения информации о пользователях привлекаются специальные идентификационные центры. Такой подход позволяет пользователям раскрывать только часть информации о себе при необходимости и легко проверять подлинность персональных данных.

ПРОЕКТИРОВАНИЕ РЕШЕНИЯ

Для лучшего понимания архитектуры и особенностей предлагаемого решения необходимо подробнее остановиться на технологиях, лежащих в его основе.

Блокчейн лежит в основе первой известной пиринговой (англ. Peer-to-peer) платежной системы и одноименной цифровой валюты Bitcoin, которую некто под псевдонимом Satoshi Nakamoto представил в 2009 году [16]. Bitcoin объединил в себе уже известные на тот момент явления и понятия, такие, как:

- криптографическая система с открытым ключом;

- механизм консенсуса для отслеживания владельца монет, известный как Proof of Work.

По сути, блокчейн – это распределенный, открытый для расширения, но закрытый для изменений, прозрачный и надежный реестр для транзакций и данных любого рода. С технической точки зрения он представляет собой последовательность блоков транзакций, связанных между собой при помощи криптографической хеш-функции, где корректность каждого блока проверяется и подтверждается всеми участниками сети [17].

За время, прошедшее с момента появления Bitcoin, было представлено немало блокчейн-платформ с различными характеристиками и предоставляемыми возможностями, каждая из которых обладает своей спецификой и предполагает определенные сценарии использования. В [18] проведено сравнение трех наиболее проработанных и распространенных на данный момент фреймворков, поддерживающих смарт-контракты: Hyperledger Fabric, R3 Corda, Ethereum.

Hyperledger Fabric – это модульная блокчейн-платформа, предназначенная для проектирования и разработки крупных промышленных решений. R3 Corda – блокчейн-платформа, предназначенная исключительно для разработки решений в индустрии финансовых услуг. Обе платформы предоставляют ограниченный доступ к участию в консенсусе (англ. Permissioned blockchain). Только определенная группа участников сети имеет доступ к истории всех транзакций и возможность проверять и подтверждать корректность добавляемых данных.

Ethereum – универсальная блокчейн-платформа общего назначения, которая делает возможным разработку любых приложений в самых разных областях [19]. Платформа является полностью открытой (англ. Public blockchain) и предоставляет всем участникам единообразный доступ к участию в консенсусе. Все участники сети имеют доступ к истории всех транзакций и возможность проверять и подтверждать корректность добавляемых данных. Универсальная и открытая природа данной платформы лучше всего подходит для проектирования предлагаемого решения. Ethereum унаследовал многие принципы работы и свойства у Bitcoin, в том числе механизм консенсуса Proof of Work. Но есть и принципиальные отличия, главные из которых таковы:

- поддержка смарт-контрактов; смарт-контракты – как контракты из жизни; любая логика, представленная жестко запрограммированным алгоритмом; смарт-контракты разворачиваются непосредственно в блокчейне и исполняются на всех узлах сети;
- поддержка встроенного Тьюринг-полного (англ. Turing-complete) языка программирования Solidity, который используется для написания смарт-контрактов.

Так как Ethereum поддерживает Тьюринг-полный язык программирования и каждый может создать любое децентрализованное приложение с какими угодно правилами, определив их в смарт-контракте, разработчикам платформы необходимо было предотвратить возможность специфических атак, эксплуатирующих неразрешимость проблемы остановки (англ. Haunting problem). Для этого каждая транзакция разбивается на конечное число операций, необходимых для ее выполнения. Каждой операции соответствует определенное число вычислительных шагов (англ. Gas), общее число которых для всех операций не может превышать заранее заданное значение (англ. Gas limit), которое в обязательном порядке указывается для всех транзакций.

Хочется обратить особое внимание на те особенности децентрализованных приложений, которые отличают их от традиционных централизованных приложений.

Архитектура традиционных веб-приложений (Рис. 1) подразумевает наличие сервера или группы серверов, которые отвечают за аутентификацию пользователей, то есть хранят данные учетных записей пользователей, за всю логику работы приложения и взаимодействия с пользователями, хранение и предоставление статических файлов, а также обмен сообщениями с другими сервисами. Пользователю в свою очередь предоставляется интерфейс взаимодействия с сервером.

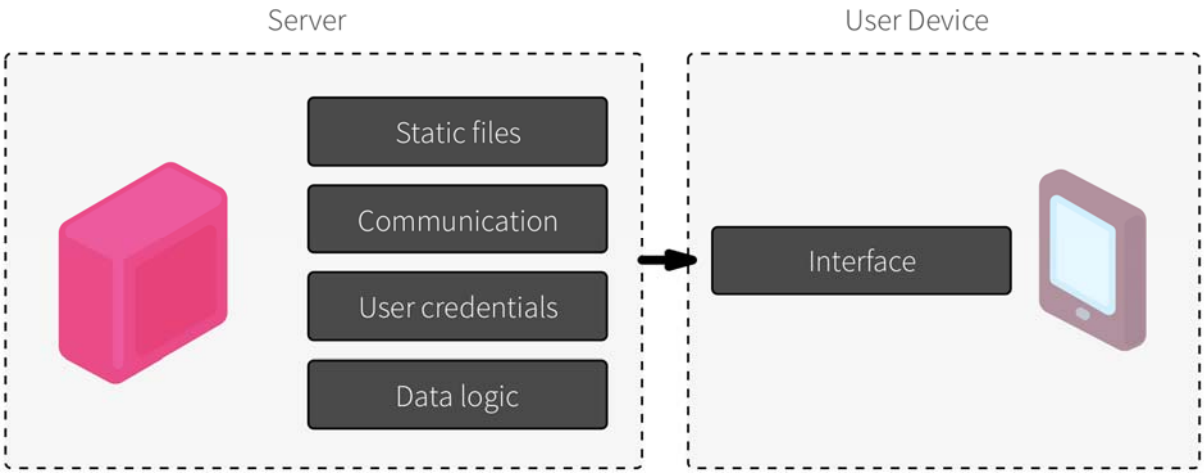


Рис. 1. Архитектура традиционных веб приложений [20]

Архитектура децентрализованных приложений (Рис. 2) также предполагает наличие пользовательского интерфейса, но, кроме этого, на пользователя ложится еще и задача по хранению данных своей учетной записи, конкретно – секретного ключа (англ. private key) в тайне, в то время как открытый ключ (англ. public key), составляющий пару секретному ключу, может свободно распространяться. Функции сервера по инкапсуляции логики работы приложения в данном случае выполняет блокчейн, в частности, смарт-контракты, к которым напрямую обращаются пользователи, используя свои ключи. Для хранения и предоставления по требованию статических файлов используются распределенные хранилища, такие, как Swarm и IPFS [21], а для обмена сообщениями между сервисами в настоящее время разрабатывается протокол Whisper.

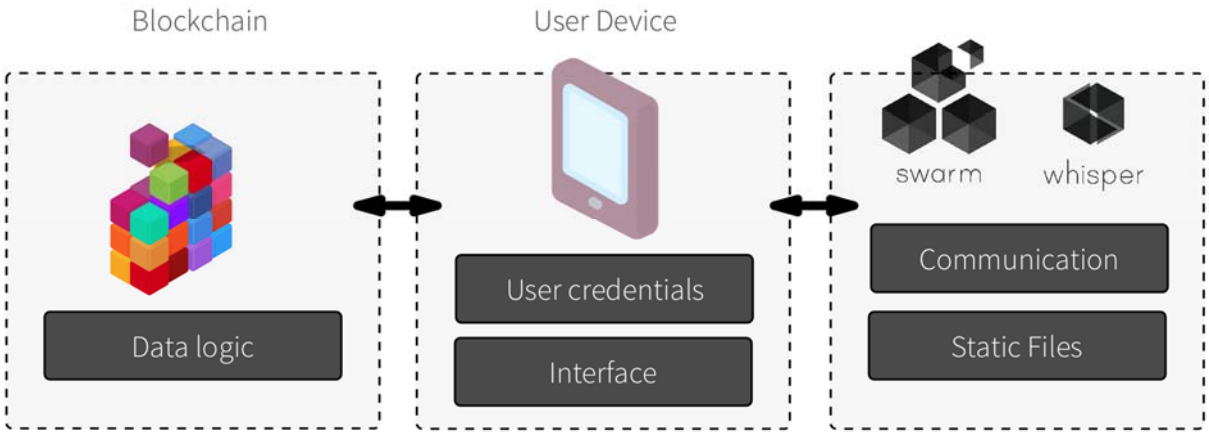


Рис. 2. Архитектура децентрализованных приложений [20]

РЕШЕНИЕ

В разработанном децентрализованном программном продукте можно выделить следующие основные компоненты (Рис. 3):

- пользовательский интерфейс;
- блокчейн Ethereum, а именно, смарт-контракты, описывающие участников системы и логику их взаимодействия, а также фиксирующие факты оформления трудовых отношений между ними;
- сервер, инкапсулирующий в себе ту часть логики работы приложения, которая не нуждается в свойствах блокчейна и может существовать за его рамками, тем самым не тратя лишние ресурсы; к такой функциональности можно отнести отправку предложения о работе и его согласование;
- распределенное хранилище IPFS, которое используется для хранения различных данных, таких, как данные профилей пользователей, открытые и секретные детали трудовых отношений, а также рекомендации, которые работники получают от своих работодателей. При этом в самом блокчейне хранятся только ссылки, соответствующие фрагментам этих данных.

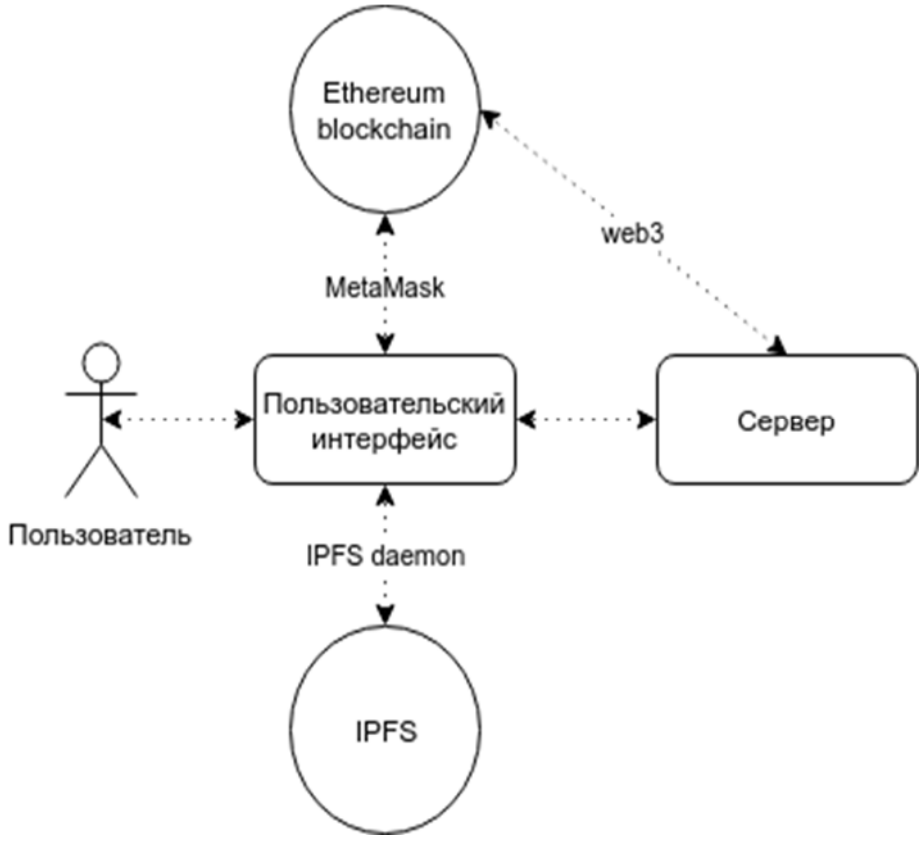


Рис. 3. Архитектура компонентов системы

Для взаимодействия пользовательского интерфейса с блокчейном Ethereum используется расширение для браузера MetaMask, которое предоставляет возможность хранения ключей и подписи транзакций на стороне пользователя. Важно заметить, что пользователь подписывает все транзакции локально и осуществляет полный контроль своих ключей. Для взаимодействия пользовательского интерфейса с распределенным хранилищем IPFS используется специальная программа, работающая в фоновом режиме, – демон IPFS. Для взаимодействия сервера с блокчейном Ethereum используется Javascript библиотека web3.js.

Рассмотрим задачу идентификации пользователей. Под идентификацией понимается процесс сопоставления цифровой и реальной сущностей (англ. identity), то есть проверка того, что пользователь действительно является тем, за кого себя выдает; будь то соискатель или представитель организации. В результате исследования различных решений в этой области был сделан вывод, что самый надежный и универсальный способ идентификации пользователей предоставляет Единая система идентификации и аутентификации (ЕСИА) [22]. Механизм аутентификации пользователей, использованный в программном интерфейсе ЕСИА, основан на спецификации OAuth 2.0, которая является своего рода стандартом и, следовательно, позволяет легко осуществлять интеграцию. Но, принимая во внимание тот факт, что процедура подключения к программному интерфейсу ЕСИА подразумевает наличие действующего юридического лица и удовлетворения других бюрократических требований, было принято решение отложить интеграцию предлагаемого решения с ЕСИА в качестве дальнейших планов. В рамках данной работы организации должны публично декларировать свой уникальный адрес, однозначно получаемый из открытого ключа, чтобы все пользователи могли убедиться в аутентичности организации, представленной на платформе, перейдя на ее информационный ресурс и сопоставив адреса. Идентификация соискателей в свою очередь доверена их первому работодателю, трудовые взаимоотношения с которым регистрируются при помощи данной системы.

На данный момент в системе есть два типа аккаунтов – соискатель и работодатель. При регистрации новые пользователи сами выбирают тип создаваемо-

го аккаунта и вводят соответствующие данные. Полученные данные сначала сохраняются в IPFS, после чего в памяти основного смарт-контракта сохраняется связка текущего адреса пользователя со ссылкой на его данные в IPFS. Такой замысловатый способ хранения данных мотивирован сугубо экономическими соображениями, о которых следует сказать подробнее.

Комиссия транзакции в сети Ethereum определяется количеством операций, необходимых для ее выполнения, и объемом оперируемых при этом данных. Теоретически чем меньше данных хранится в памяти смарт-контракта, тем меньше будет комиссия за его создание и эксплуатацию. Эта гипотеза была подтверждена экспериментально посредством сравнения двух разных смарт-контрактов:

1. В первом случае профили пользователей хранились непосредственно в памяти смарт-контракта в виде специально созданных структур данных. При создании нового аккаунта соответствующая структура инстанцировалась и сохранялась в памяти.
2. Во втором случае профили пользователей хранились в IPFS, а в памяти смарт-контракта сохранялась только соответствующая ссылка.

Полученные результаты говорят сами за себя: комиссия за создание второго смарт-контракта уменьшилась на 11% по сравнению с первым, а комиссия за создание нового аккаунта сократилась соответственно на 41%. Аналогичный подход к хранению данных применяется также и в других сценариях использования.

Работодатели имеют возможность искать подходящих кандидатов, просматривая портфолио соискателей. Процесс получения портфолио описан на диаграмме последовательности (англ. Sequence diagram) (Рис. 4). В случае взаимной заинтересованности работодатель может направить соискателю предложение о работе, включающее детали соглашения, подписанные цифровой подписью работодателя.

Для обеспечения конфиденциальности таких чувствительных деталей соглашения, как, например, размер заработной платы, детали соглашения делятся на две части: открытые и секретные (Рис. 5). К открытым данным среди прочих относятся: данные о приглашающей организации, предлагаемая должность и предварительная дата выхода на работу; к секретным данным: зарплата и дру-

гие условия, описанные в договоре или контракте. Открытые данные в исходном виде сохраняются в IPFS, тогда как секретные данные асимметрично шифруются при помощи секретного ключа работодателя-отправителя и открытого ключа соискателя-получателя, после чего в зашифрованном виде сохраняются в IPFS. Комбинация двух полученных в результате ссылок подписывается секретным ключом работодателя-отправителя. Все вышеперечисленные операции выполняются на стороне пользователя, после чего полученные предложения о работе отправляются на сервер.

Рис. 4. Диаграмма последовательности получения портфолио

Использование асимметричного шифрования гарантирует конфиденциальность чувствительных деталей соглашения – только соискатель-получатель может их расшифровать. Цифровая подпись работодателя-отправителя позволяет подтвердить аутентичность полученного предложения о работе и идентифицировать его отправителя.

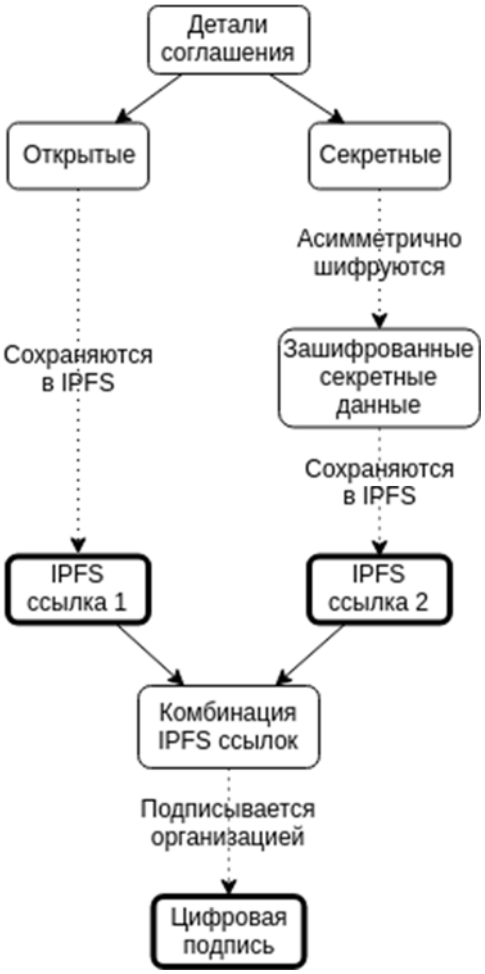


Рис. 5. Схема преобразования и подписи деталей соглашения

Соискатели всегда имеют доступ к своим портфолио и имеют возможность рассматривать полученные предложения о работе. В случае, если соискатель на данный момент нигде не работает, он может принять подходящее предложение, подписав детали соглашения своей цифровой подписью. Соответствующие данные отправляются на сервер, после чего в блокчейне будет создан новый смарт-контракт, инкапсулирующий в себе всю информацию о текущих трудовых отношениях, а в памяти основного смарт-контракта сохраняется связка адреса соискателя с адресом созданного смарт-контракта. Таким образом, факт оформления трудовых отношений надежно фиксируется в блокчейне вместе с актуальной временной меткой (англ. timestamp). Далее эту запись изменить будет нельзя, и она, фактически, является цифровым аналогом записи в трудовой книжке, включая подписи сторон, подлинность которых легко проверить (Рис. 6).

Рис. 6. Алгоритм проверки подлинности подписи

ЗАКЛЮЧЕНИЕ

Спроектирован и разработан децентрализованный программный продукт с открытым исходным кодом (англ. open-source) [23], основанный на технологии блокчейн, который предоставляет работодателям и соискателям возможность взаимодействовать и вести трудовые отношения друг с другом напрямую в цифровом формате.

К достоинствам данного решения можно отнести то, что в отличие от многих других работ и исследований приводятся не только концептуальное описание системы, но и практическая реализация с описанием деталей, важных для реального применения и внедрения.

Дальнейшие планы и направления исследования:

- совершенствование методов идентификации пользователей, интеграция с ЕСИА;
- добавление поддержки высших учебных заведений и возможности присвоения цифрового аналога академических сертификатов, подкрепленных записью в блокчейне;
- внедрение разработанного решения на базе Высшей школы информационных технологий и интеллектуальных систем Казанского федерального университета.

СПИСОК ЛИТЕРАТУРЫ

1. *Кадникова Н.* Электронный документооборот: преимущества и недостатки //Кадровик. 2014. №. 1. С. 179.

2. Сетдарова Л. Б. Электронная трудовая книжка: Реалии и Перспективы //Гуманитарные и юридические исследования. 2013. № 2.

3. Исследование ВЦИОМ. URL: https://wciom.ru/index.php?id=236&uid=116622

4. Исследование CareerBuilder. URL: http://www.careerbuilder.com/share/aboutus/pressreleasesdetail.aspx?sd=8/7/2014&id=pr837&ed=12/31/2014

5. *Cherkasov A.* Decentralized Professional Platform of the Ethereum blockchain. URL: https://aworker.io/docs/Aworker_Whitepaper.pdf

6. *Chen Z., Zhu Y*. Personal Archive Service System using Blockchain Technology: Case Study, Promising and Challenging //AI & Mobile Services (AIMS), 2017 IEEE International Conference on. IEEE, 2017. P. 93–99.

7. *Pinna A., Ibba S*. A blockchain-based Decentralized System for proper handling of temporary Employment contracts //arXiv preprint arXiv:1711.09758. 2017.

8. *Wang X*. et al. Human resource information management model based on blockchain technology //Service-Oriented System Engineering (SOSE), 2017 IEEE Symposium on. IEEE, 2017. P. 168–173.

9. *Grech A*. et al. Blockchain in Education. Joint Research Centre (Seville site), 2017. №. JRC108255.

10. *Sharples M., Domingue J*. The blockchain and kudos: A distributed system for educational record, reputation and reward //European Conference on Technology Enhanced Learning. Springer, Cham, 2016. P. 490–496.

11. Holberton School to Authenticate Its Academic Certificates With the Bitcoin Blockchain. URL: http://www.marketwired.com/press-release/holberton-school-authenticate-its-academic-certificates-with-bitcoin-blockchain-2065768.htm

12. *Durant E., Trachy A*. Digital Diploma debuts at MIT. URL: http://news.mit.edu/2017/mit-debuts-secure-digital-diploma-using-bitcoin-blockchain-technology-1017

13. *Sony Global Education Develops Technology Using Blockchain for Open Sharing of Academic Proficiency and Progress Records*. URL: https://www.sony.net/ SonyInfo/News/Press/201602/16-0222E/index.html

14. *Electronic apparatus, method for electronic apparatus and information processing system. URL: https://patents.justia.com/patent/20170346637*

15. *Civic White paper.* URL: https://tokensale.civic.com/CivicTokenSaleWhitePaper.pdf

16. *Nakamoto S*. Bitcoin: A peer-to-peer electronic cash system. 2008.

17. *Evans P*. Thinking outside the blocks: a strategic perspective on block-chain and digital tokens. URL: https://www.bcg.com/blockchain/thinking-outside-the-blocks.html

18. *Valenta M., Sandner P*. Comparison of Ethereum, Hyperledger Fabric and Corda. FSBC Working Paper, 2017.

19. *Buterin V*. et al. A next-generation smart contract and decentralized application platform //white paper. 2014.

20. *Van de Sande A*. How to build serverless applications for Mist. URL: https://blog.ethereum.org/2016/07/12/build-server-less-applications-mist

21. *Benet J*. Ipfs-content addressed, versioned, p2p file system //arXiv preprint arXiv:1407.3561. 2014.

22. Документация API ЕСИА. URL: https://partners.gosuslugi.ru/ catalog/esia

23. *Шайфутдинов А.* Репозиторий на GitHub. URL: https://github.com/paradisensei/CareerTracker

CAREER DIGITAL PASSPORT BASED ON DISTRIBUTED LEDGER TECHNOLOGY

A. I. Shaifutdinov[1], A. F. Khasyanov[2]

Higher School of Information Technologies and Intelligent Systems, Kazan Federal University

[1]aidarshaifutdinov@gmail.com, [2]ak@it.kfu.ru

Abstract

This paper considers problems associated with the documentation of employment process and management of employment records. Today, these tasks are solved through paper contracts and, in the Russian Federation, through «labor books». In this paper a software solution based on distributed ledger technology (blockchain) and smart contracts is proposed to replace the existing paper workflow.

Keywords*: employment contracts, employment records, paper workflow, blockchain, smart contracts, digitalization, decentralized applications, Ethereum, Solidity, IPFS*

REFERENCES

1. *Kadnikova N.* Elektronnyy dokumentooborot: preimuschestva i nedostatki //Kadrovik. 2014. №. 1. P. 179.

2. *Setdarova L.* Electronnaya trudovaya knizhka: Realii i Perspectivy//Gumanitarnye i yuridicheskie issledovaniya. 2013. № 2.

3. Issledovanie WCIOM. URL: https://wciom.ru/index.php?id=236&uid=116622

4. Issledovanie CareerBuilder. URL: http://www.careerbuilder.com/share/aboutus/pressreleasesdetail.aspx?sd=8/7/2014&id=pr837&ed=12/31/2014

5. *Cherkasov A.* Decentralized Professional Platform of the Ethereum blockchain. URL: https://aworker.io/docs/Aworker_Whitepaper.pdf

6. *Chen Z., Zhu Y*. Personal Archive Service System using Blockchain Technology: Case Study, Promising and Challenging //AI & Mobile Services (AIMS), 2017 IEEE International Conference on. IEEE, 2017. P. 93–99.

7. *Pinna A., Ibba S*. A blockchain-based Decentralized System for proper handling of temporary Employment contracts //arXiv preprint arXiv:1711.09758. 2017.

8. *Wang X*. et al. Human resource information management model based on blockchain technology //Service-Oriented System Engineering (SOSE), 2017 IEEE Symposium on. IEEE, 2017. P. 168–173.

9. *Grech A*. *et al.* Blockchain in Education. Joint Research Centre (Seville site), 2017. №. JRC108255.

10. *Sharples M., Domingue J*. The blockchain and kudos: A distributed system for educational record, reputation and reward //European Conference on Technology Enhanced Learning. Springer, Cham, 2016. P. 490–496.

11. *Holberton School to Authenticate Its Academic Certificates with the Bitcoin Blockchain.* URL: http://www.marketwired.com/press-release/holberton-school-authenticate-its-academic-certificates-with-bitcoin-blockchain-2065768.htm

12. *Durant E., Trachy A*. Digital Diploma debuts at MIT. URL: http://news.mit.edu/2017/mit-debuts-secure-digital-diploma-using-bitcoin-blockchain-technology-1017

13. *Sony Global Education Develops Technology Using Blockchain for Open Sharing of Academic Proficiency and Progress Records*. URL: https://www.sony.net/SonyInfo/News/Press/201602/16-0222E/index.html

14. Electronic apparatus, method for electronic apparatus and information processing system. *URL: https://patents.justia.com/patent/20170346637*

15. Civic White paper. URL: https://tokensale.civic.com/CivicTokenSaleWhitePaper.pdf

16. *Nakamoto S*. Bitcoin: A peer-to-peer electronic cash system. 2008.

17. *Evans P*. Thinking outside the blocks: a strategic perspective on block-chain and digital tokens. URL: https://www.bcg.com/blockchain/thinking-outside-the-blocks.html

18. *Valenta M., Sandner P*. Comparison of Ethereum, Hyperledger Fabric and Corda. FSBC Working Paper, 2017.

19. *Buterin V*. et al. A next-generation smart contract and decentralized application platform //white paper. 2014.

20. *Van de Sande A*. How to build serverless applications for Mist. URL: https://blog.ethereum.org/2016/07/12/build-server-less-applications-mist

21. *Benet J*. Ipfs-content addressed, versioned, p2p file system //arXiv preprint arXiv:1407.3561. 2014.

22. Dokumentatsiya *API ESIA*. URL: https://partners.gosuslugi.ru/catalog/esia

23. *Shaifutdinov A. GitHub repository.* URL: https://github.com/paradisensei/CareerTracker.

СВЕДЕНИЯ ОБ АВТОРАХ

ШАЙФУТДИНОВ Айдар Ильдарович – студент Высшей школы информационных технологий и интеллектуальных систем Казанского федерального университета. Сфера научных интересов – технология распределенных реестров (блокчейн), смарт-контракты.

Aidar Ildarovich SHAIFUTDINOV – student of Higher Institute of Information Technologies and Intelligent Systems of Kazan Federal University. Research interests include blockchain, smart contracts.

email: aidarshaifutdinov@gmail.com

ХАСЬЯНОВ Айрат Фаридович – директор Высшей школы информационных технологий и интеллектуальных систем Казанского федерального университета. Имеет степень PhD Боннского университета в области естественных наук. Сфера научных интересов лежит в области применения информационных и интеллектуальных технологий в области образования, различных отраслях информационных технологий и информатики.

Dr. Ayrat KHASYANOV obtained his PhD in Computer Science in 2005 from the University of Bonn in Germany. He serves his duty as the head of the Higher Institute for Information Technologies and Intelligent Systems at Kazan Federal University. His interests lay in the field of application of intelligent and information technology to the fields of teaching and learning, as well as various fields of information technology and computer science.

email: ak@it.kfu.ru

Материал поступил в редакцию 10 июня 2018 года

ЧАСТЬ 2

УДК 004.414.3

АРХИТЕКТУРА ОБУЧАЮЩИХ ПРИЛОЖЕНИЙ С ДОСТОВЕРНОЙ ОЦЕНКОЙ ЗНАНИЙ И ВИЗУАЛЬНЫМ ПРОЕКТИРОВАНИЕМ СЦЕНАРИЕВ ТЕСТИРОВАНИЯ В КОНЦЕПЦИИ MICROLEARNING

М. М. Абрамский[1], А. Р. Москиева[2], Р. Р. Нигматуллина[3]

[1-3] *Высшая школа информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета*

[1]ma@it.kfu.ru, [2]alinamoskieva@gmail.com, [3] RamiliaNigmatullina@gmail.com

Аннотация

Представлен подход к проектированию обучающих приложений в концепции Microlearning. Обсуждена зависимость достоверности оценки знаний от визуального представления вопросов проверочного тестирования. Приведены архитектура системы и принцип работы разработанного авторами инструмента проектирования адаптивных тестов и сценариев тестирования для microlearning-приложений.

***Ключевые слова**: микрообучение, дидактическая единица, достоверная оценка знаний, тесты, адаптивное тестирование, визуальное проектирование теста*

ВВЕДЕНИЕ

Микрообучение или microlearning – это формат обучения, при котором изучается небольшое количество информации за короткий промежуток времени. Микрообучение обладает рядом преимуществ перед стандартными уроками – это модульность, мобильность, небольшое количество времени, затрачиваемое на обучение, и гибкость [1].

Microlearning-приложения для проверки изученного материала используют тестирование – набор вопросов с разными способами ответа на них – выбор вариантов с помощью выпадающего списка или переключателей, поле для ввода информации и др. Визуальные элементы, с помощью которых пользователю дается возможность ответить, будем называть *виджетами*.

Вопрос о создании тестов в концепции Microlearning может рассматриваться отдельно от систем электронного обучения. При этом стоит отметить, что

в электронных системах обучения помимо роли студента есть роль преподавателя, у которого в рамках системы есть возможность создавать образовательный контент, в том числе тесты по изученному материалу. При создании тестов должны учитываться следующие моменты:

- При создании тестового вопроса всегда существует определенная вероятность угадывания варианта ответа по приведенному виджету. Например, в вопросе с выбором ответа вероятность будет равна 1/n, где n – количество вариантов ответа. Таким образом, существует вероятность расхождения оценки за тест с реальным уровнем знаний обучающегося.
- Вопросы в рамках теста могут быть структурированы нелинейно. К примеру, может быть использована модель адаптивного теста [2], где следующие вопросы теста зависят от выбранных вариантов ответа на предыдущие вопросы. Адаптивность теста может быть обеспечена сценарием данного теста, который задается на личное усмотрение преподавателя. Таким образом, испытуемому не придется отвечать на все вопросы, и сложность вопросов будет соответствовать уровню знаний испытуемого.
- При работе с тестом в целом следует рассматривать его не в отрыве от самого обучения, а в рамках него. Например, тест может быть более эффективным, если проходить его с определенной периодичностью. Тогда проектировщику теста в рамках обучающей системы необходимо иметь возможность гибко задавать данную периодичность. Также стоит учитывать возможное забывание материала обучающимся, что также должно быть учтено в оценке за тест (например, материал, тест по которому был пройден успешно несколько раз, может быть отмечен как запомнившийся лучше, чем материал, тест по которому был пройден всего один раз).

Были изучены примеры Microlearning-приложений [3–10], где тесты являются неотъемлемой частью системы, а также рассмотрены инструменты создания тестов Stepik [11], Moodle [12], AVELife TestGold Agent 5 [13], SunRav TestOfficePro tMarker [14], Google Forms [15]. Сделаны следующие выводы:

- большинство приложений предполагает, что изучение нового материала будет проходить ежедневно;

- ни в одной системе не реализована зависимость оценки за тест от визуального представления вариантов ответа в тесте;
- большинство приложений реализовало адаптивное тестирование и забывание, но возможность создавать гибкие тесты пользователям в роли преподавателей гибко отсутствовала в системах [11–15].

В связи с этим было принято решение разработать подход и реализующее его приложение-прототип в концепции Microlearning, где будет возможно организовывать адаптивные тесты с возможностью задавать периодичность теста, измерять знание по изучаемому модулю с учетом вероятности угадывания ответа, текущей степени освоения и забывания ранее изученного материала.

Статья построена следующим образом. В первом разделе описаны выбранная модель знаний и типы заданий для ее проверки, во втором разделе представлены формула зависимости достоверности оценки от проверяемого виджета, а также реализованный механизм забывания. В третьем разделе рассказано о подходе к проектированию адаптивного тестирования с помощью визуального проектирования теста, а также о создании сценариев тестирования.

1. ХРАНЕНИЕ ЗНАНИЙ И ТИПЫ ЗАДАНИЙ ДЛЯ ИХ ПРОВЕРКИ

Модель знания проектируемой системы определяет уровень знаний учащегося по предметной области. Предметная область разбивается на темы, подтемы и дидактические единицы.

Модель позволяет пользователю в каждый момент времени знать процентное значение степени усвоения дидактической единицы. При подсчете учитываются текущий ответ, представление ответа или тот виджет, который используется в качестве формы ответа в тесте на текущем вопросе, значение степени освоения на предыдущем шаге и забывание. Такое представление модели знания способствует развитию мелкой компетенции.

Процесс проверки знаний происходит путем отправления ответа пользователя через форму, представленную на странице, которая называется виджетом. Виджет [16] – это небольшой независимый программный модуль, работающий в некоторой среде, который исполняет одну определенную функцию.

Для отправки решения задачи можно подобрать большое количество виджетов. Все они могут отличаться не только внешним видом, но и действием. Чтобы определить наличие связи виджета и факта освоения, а также его удоб-

ства был проведен опрос, в котором пользователям предлагалось пройти тест с использованием разных виджетов. Для опроса были выбраны следующие виджеты:

1. Необходимо записать ответ в поле ввода;
2. Необходимо записать ответ с помощью поля, где можно увеличивать текущее значение;
3. Необходимо кликнуть верный ответ при помощи клавиатуры, представленной в виде кнопок;
4. Необходимо записать числовой ответ прописью;
5. Необходимо записать номер верного варианта ответа;
6. Необходимо кликнуть на номер верного ответа;
7. Необходимо выбрать верный ответ из выпадающего списка;
8. Необходимо поставить галочку около верного ответа;
9. Необходимо кликнуть на кнопки «да» или «нет»;
10. Необходимо написать ответ прописью да или нет.
11. Необходимо выбрать ответ «да» или «нет» из выпадающего списка;
12. Необходимо выбрать верный ответ, используя радиокнопки.

Результаты опроса представлены ниже (Рис. 1): сделан вывод, что виджет не влияет на факт освоения.

Для приложения важно, в каком виде приходит ответ от пользователя. Для выявления типов заданий по типу ответа было принято решение проанализировать предметную область и выделить возможные типы. В качестве тестовой предметной области была выбрана дискретная математика. После анализа задачников [17] и [18] выделены следующие типы:

1. Задания с числовым ответом;
2. Задания с ответом в виде рисунка;
3. Задания с выбором ответа;
4. Задания с развернутым ответом;
5. Задания с ответом да/нет.

Для каждого типа заданий можно выделить вероятность того, что пользователь имеет знания, проверяемые вопросом в тесте (Табл. 1).

Название виджета	Количество оценок	Средняя оценка	Процент верных ответов
Виджет #1	50	3.77	0.7333
Виджет #2	50	2.62	0.6667
Виджет #3	50	3.0	0.7143
Виджет #4	50	1.99	0.4643
Виджет #5	50	2.95	0.8571
Виджет #6	50	3.87	0.7931
Виджет #7	50	3.48	0.7037
Виджет #8	50	3.26	0.6538
Виджет #9	50	4.19	0.8846
Виджет #10	50	2.38	0.6923
Виджет #11	50	3.07	0.8077
Виджет #12	50	3.9	0.1538

Рис. 1. Результаты опроса пользователей о удобстве использования виджетов

Таблица 1. Таблица коэффициентов, с которыми пользователь может угадать ответ на вопрос

Тип задания	Вероятность владения проверяемым знанием с учетом представления
Числовой ответ	$\frac{1}{n}$, где n – число всевозможных ответов
Ответ в виде рисунка	$\frac{1}{n}$, где n – число всевозможных вариантов рисунков
Выбор ответа	$\frac{1}{n}$, где n – число всевозможных вариантов ответа
Развернутый ответ	1
Ответом «да» или «нет»	0.5

2. СТЕПЕНЬ УСВОЕНИЯ ДИДАКТИЧЕСКОЙ ЕДИНИЦЫ И ЗАБЫВАНИЕ ЗНАНИЙ

Степень освоения дидактической единицы – это численный показатель обладания пользователя проверяемым знанием, измеряемый в процентном соотношении или имеющий численное значение, находящееся в интервале [0, 100]. Для расчета такой степени было сформулировано правило:

$$KUL' = KUL + increment,$$

где KUL’ (knowledge unit level) – это новое значение коэффициента, KUL – значение, полученное на предыдущем шаге; $increment = f(KUL, p)$ – это значение, которое зависит от предыдущего значения коэффициента усвоения дидактической единицы, объема еще не изученного материала и вероятности того, что пользователь обладает проверяемым в вопросе знанием с учетом представления задания и вероятностью угадывания ответа;

$$increment = \frac{1 - KUL}{n} = \left[\frac{1}{n} = p\right] = p\,(1 - KUL),$$

где *p* – вероятность того, что пользователь обладает знанием по проверяемой в вопросе дидактической единице с учетом представления задания и вероятности угадывания ответа.

По результатам исследований памяти Германа Эббингауза [19] были сформулированы правила, по которым уменьшается значение степени усвоения дидактической единицы:

- после одного дня без повторений значение степени освоения должно уменьшится на 34%;
- после двух дней без повторений значение степени освоения должно уменьшится на 28%;
- после шести дней без повторений значение степени освоения должно уменьшится на 25%;
- после 31 дня без повторений значение степени освоения должно уменьшится на 21%.

3. ИНСТРУМЕНТ ВИЗУАЛЬНОГО ПРОЕКТИРОВАНИЯ ТЕСТОВ

В результате проведенной работы было разработано приложение для создания тестов и «программ тренировок» – запланированных периодических прохождений определенных тестов.

В основе тестов лежит сценарий – последовательность вопросов в зависимости от правильности ответов.

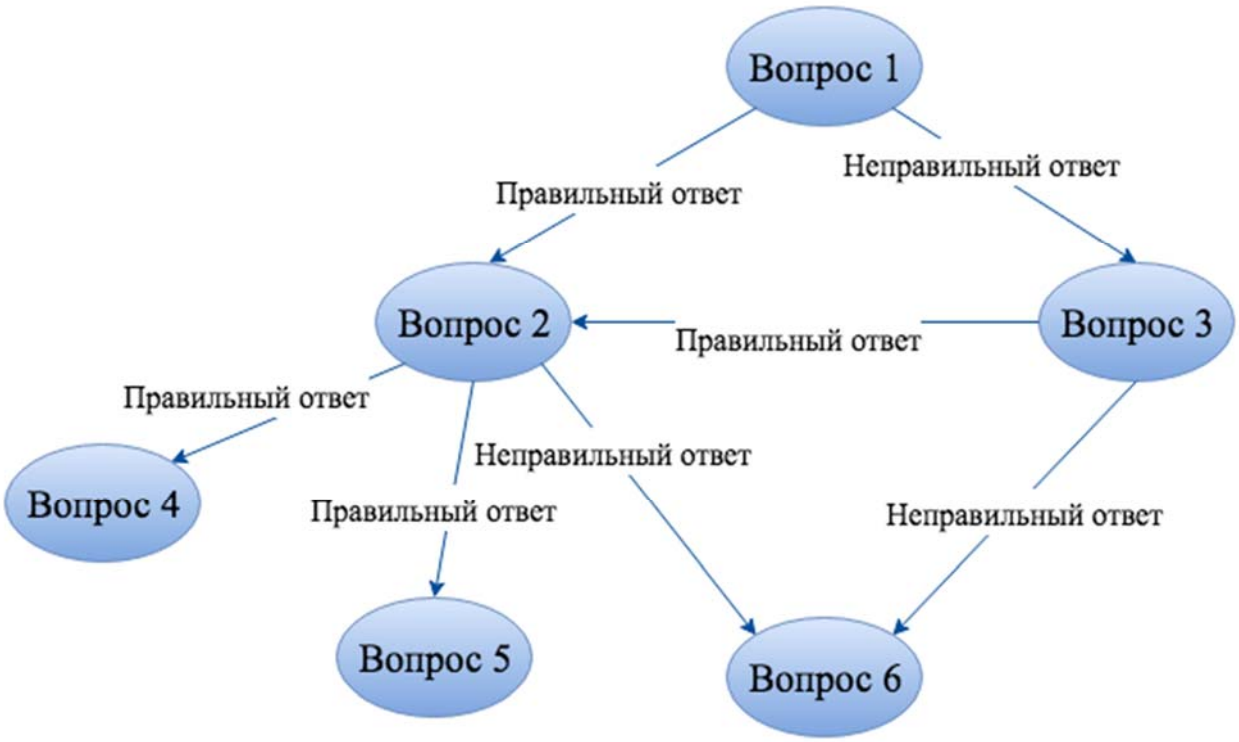


Рис. 2. Сценарий теста

С помощью библиотеки Directed Graph Editor была спроектирована возможность создания сценария. Сценарий создается преподавателем. Для создания сценария ему необходимо добавить вопросы в сценарий и определить верные и неверные пути. Пример использования Directed Graph Editor приведен на рисунках 3–5. Номера в кружочках — это номера вопросов из списка, отображаемого преподавателю.

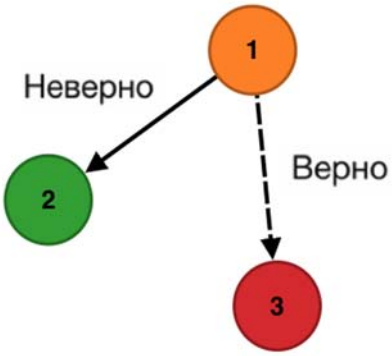


Рис. 3. Создание сценария теста

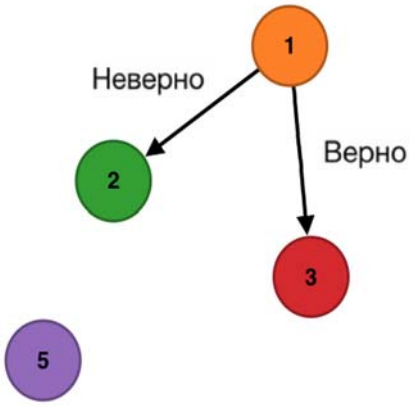


Рис. 4. Создание сценария теста: добавление пятого вопроса в сценарий.

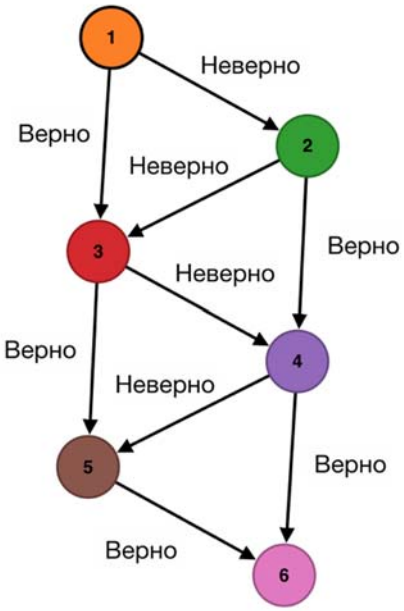


Рис. 5. Готовый сценарий теста

Для лучшего усвоения материала преподаватели могут задавать разные настройки теста: выбирать виджеты, с помощью которых должны быть представлены вопросы теста, указывать частоту повторений теста. Например, можно задать, что тест должен повторяться 1 раз в неделю в первый месяц изучения и 1 раз в месяц – в последующие 3 месяца. Также преподаватели могут создавать программы тренировок, которые позволяют объединять тесты в группы и задавать определенную их последовательность. С помощью программ тренировок преподаватели смогут проводить систематическое тестирование. Данные программы могут быть направлены на достижение определенных целей, например, подготовки к контрольной работе.

ЗАКЛЮЧЕНИЕ

В ходе выполнения исследования были выявлены факторы, которые необходимо учитывать при проверке ответов учащихся. Описан процесс проверки, при котором рассчитывается степень усвоения материала, изучаемого в рамках дидактической единицы, учитывающая достоверность оценки и забывания. Предложен подход к визуальному проектированию тестов и сценариев тестирования.

В дальнейших исследованиях планируются изучение особенностей тестирования в различных предметных областях, а также развитие интерактивного подхода к проектированию тестов.

СПИСОК ЛИТЕРАТУРЫ

1. *Peter A. Bruck*. Mobile Learning with Micro-content: A Framework and Evaluation, 2012.

2. *Сметанюк Л.В., Кравцов Г.М.* К теории и практике использования адаптивных тестов, 2008. 150 с. [Электронный ресурс]. URL: http://www.ite.kspu.edu/webfm_send/509/1

3. Duolingo [Электронный ресурс]. URL: https://www.duolingo.com/

4. LinguaLeo [Электронный ресурс]. URL: https://lingualeo.com/ru

5. Инструмент для создания тестов AVELife TestGold Agent 5 [Электронный ресурс]. URL: http://avelife.ru/products/testgold/agent.htm

6. Инструмент для создания тестов SunRav TestOfficePro tMarker. URL: https://www.sunrav.ru/docs/testofficepro/tmaker/100.html

7. Инструмент для создания тестов Quizlet [Электронный ресурс]. URL: https://quizlet.com/ru

8. Semper: Учись легко [Электронный ресурс] / Сайт Google play. URL: https://play.google.com/store/apps/details?id=co.unlockyourbrain

9. Drops [Электронный ресурс]. URL: https://languagedrops.com/

10. Word of the day [Электронный ресурс]. URL: http://wordwordapp.com/

11. Онлайн-курсы от ведущих вузов и компаний страны [Электронный ресурс]. URL: https://welcome.stepik.org/ru

12. Портал дистанционного обучения СПбГАСУ [Электронный ресурс]. URL: https://moodle.spbgasu.ru

13. Инструмент для создания тестов AVELife TestGold Agent 5 [Электронный ресурс]. URL: http://avelife.ru/products/testgold/agent.htm

14. Инструмент для создания тестов SunRav TestOfficePro tMarker [Электронный ресурс]. URL: https://www.sunrav.ru/docs/testofficepro/tmaker/100.html

15. Get started with Forms [Электронный ресурс]. URL: https://gsuite.google.com/ learning-center/products/forms/get-started/

16. Достоверность работы компьютерных систем [Электронный ресурс]. URL: //www.immsp.kiev.ua/publications/articles/2016/2016_4/04_2016_Cespedes.pdf

17. *Гаврилов Г.П., Сапоженко А.А.* Задачи и упражнения по дискретной математике: Учеб. пособие. 3-е изд., перераб. М.: ФИЗМАТЛИТ, 2005. 416 с. ISBN 5-9221-0477-2.

18. *Альпин Ю.А., Ильин С.Н.* Задачи по дискретной математике: Учебно-методическое пособие. Казань: Казанский федеральный университет, 2013. 26 с.

19. О скорости забывания [Электронный ресурс]. URL: http://www.voppsy.ru/issues/1983/834/834142.html

MICROLEARNING APPS ARCHITECTURE WITH RELIABLE KNOWLEDGE EVALUATION AND VISUAL DESIGN OF TESTING SCENARIOS

M. M. Abramskiy[1], A. R. Moskieva, R. R. Nigmatullina[3]

[1-3] *Higher School of Information Technologies and Intelligent Systems of Kazan (Volga Region) Federal University*

[1]ma@it.kfu.ru, [2]alinamoskieva@gmail.com, [3]RamiliaNigmatullina@gmail.com

Abstract

An approach for designing Microlearning Applications is presented. The dependency of reliability of knowledge evaluation from visual representation of testing questions. And architecture of system and working principle of developed instrument for designing adaptive tests and testing scenarios is shown.

Keywords*: Microlearning, didactic unit, reliable knowledge evaluation, tests, adaptive testing, visual design of tests*

REFERENCES

1. *Peter A. Bruck.* Mobile Learning with Micro-content: A Framework and Evaluation, 2012

2. *Smetanyuk L.V., Kravtsov G.M. K* teorii i praktike ispol'zovaniya adaptivnykh testov. 2008. 150 s. URL: http://www.ite.kspu.edu/webfm_send/509/1

3. Duolingo. URL: https://www.duolingo.com/

4. LinguaLeo. URL: https://lingualeo.com/ru

5. Instrument dlya sozdaniya testov AVELife TestGold Agent 5. URL: http://avelife.ru/products/testgold/agent.htm

6. Instrument dlya sozdaniya testov SunRav TestOfficePro tMarker. URL: https://www.sunrav.ru/docs/testofficepro/tmaker/100.html

7. Instrument dlya sozdaniya testov Quizlet. URL: https://quizlet.com/ru

8. Semper: Uchis' legko / Sajt Google play. URL: https://play.google.com/store/apps/details?id=co.unlockyourbrain

9. Drops. URL: https://languagedrops.com/

10. Word of the day. URL: http://wordwordapp.com/

11. Onlajn-kursy ot vedushhikh vuzov i kompanij strany. URL: https://welcome.stepik.org/ru

12. Portal distantsionnogo obucheniya SPbGASU. URL: https://moodle.spbgasu.ru

13. Instrument dlya sozdaniya testov AVELife TestGold Agent 5. URL: http://avelife.ru/products/testgold/agent.htm

14. Instrument dlya sozdaniya testov SunRav TestOfficePro tMarker. URL: https://www.sunrav.ru/docs/testofficepro/tmaker/100.html

15. Get started with Forms. URL: https://gsuite.google.com/learning-center/products/forms/get-started/

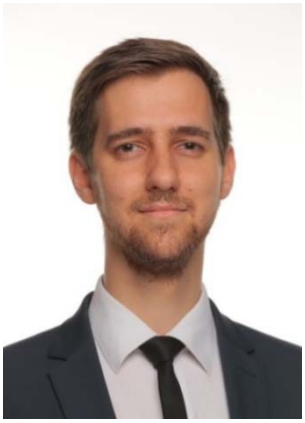
16. Dostovernost' raboty komp'yuternykh system. URL: http://www.immsp.kiev.ua/publications/articles/2016/2016_4/04_2016_Cespedes.pdf.

17. *Gavrilov G.P., Sapozhenko A.A.* Zadachi i uprazhneniya po diskretnoj matematike: Ucheb. posobie. 3-e izd., pererab. M.: FIZMATLIT, 2005. 416 s. ISBN 5-9221-0477-2.

18. *Al'pin Yu.A., Il'in S.N.* Zadachi po diskretnoj matematike: Uchebno-metodicheskoe posobie. Kazan': Kazanskij Federal'nyj Universitet, 2013. 26 s.

19. O skorosti zabyvaniya. URL: http://www.voppsy.ru/issues/1983/834/834142.html, svobodnyj.

СВЕДЕНИЯ ОБ АВТОРАХ



АБРАМСКИЙ Михаил Михайлович – старший преподаватель кафедры программной инженерии Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета.

Mikhail Mikhailovich ABRAMSKIY – senior lecturer at Department of Software Engineering of Higher School of ITIS KFU.

email: ma@it.kfu.ru

МОСКИЕВА Алина Рустемовна – студентка Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета.

Alina Rustemovna MOSKIEVA – student of Higher School of ITIS KFU.

email: alinamoskieva@gmail.com

НИГМАТУЛЛИНА Рамиля Радиковна – студентка Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета.

Ramilia Radikovna NIGMATULLINA – student of Higher School of ITIS KFU.

email: RamiliaNigmatullina@gmail.com

Материал поступил в редакцию 1 августа 2018 года

УДК 004.387

ПРОГРАММИРОВАНИЕ ЗАПАХОВ ДЛЯ ВИРТУАЛЬНОГО ОСМОТРА МЕСТА ПРОИСШЕСТВИЯ

И. О. Антонов[1], К. В. Зезегова[2], В. В. Кугуракова[3], Е. Н. Лазарев[4], М. Р. Хафизов[5]

[1]*Юридический факультет Казанского (Приволжского) федерального университета;*

[2-5]*Высшая школа информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета*

[1]igolant@mail.ru, [2]hig154era@gmail.com, [3]vlada.kugurakova@gmail.com, [4]evgenln11401@gmail.com, [5]murkorp@gmail.com

Аннотация

Проанализированы существующие программы виртуального осмотра места происшествия и выделены основные важные критерии, которые могут понадобиться для разработки обучающего приложения с использованием виртуальной реальности. Для повышения погружения в иммерсивную виртуальную среду, воссоздающую процесс осмотра места происшествия, изучены устройства, генерирующие запахи, и выбрано оптимальное. Разработан метод использования ароматов в виртуальном осмотре места происшествия. Этот метод может быть использован и в других сферах, требующих при вынесении решений знания о составе воздуха.

Ключевые слова: *виртуальная реальность, VR, иммерсивность, виртуальный запах, цифровой запах, криминалистика, обучение криминалистов, виртуальные симуляции*

ВВЕДЕНИЕ

В настоящее время виртуальная реальность развивается очень интенсивно, она применяется во многих сферах, в том числе в обучении. Например, виртуальная реальность используется в изучении биомедицинских технологий: используются виртуальные лаборатории, в которых симулируются различные процессы [1].

На сегодняшний день развитие дошло и до виртуализации запахов, что очень важно при погружении в виртуальную реальность [2]. Данный аспект также важен для развития методик обучения в областях, где главным является не только зрительный, но и обонятельный фактор.

С использованием технологий виртуального запаха улучшится процесс обучения, ведь теперь каждый студент или школьник сможет самостоятельно, без возможного ущерба здоровью, побывать в смоделированных ситуациях, близких к реальности. Это увеличит наглядность изучения специальности и, соответственно, качество получаемых знаний.

Следователь – это как раз та специальность, где очень важен практический опыт, в том числе и знакомство с потенциально возможной запаховой картиной [3] на месте происшествия. Виртуальная реальность, использующая еще и устройства, задействующие органы обоняния, способствует созданию эффекта полного погружения [4] в изучаемую предметную область, что позволит улучшить уровень практического понимания многих аспектов профессии и способно дополнительно заинтересовать обучающихся. Здесь же могут использоваться средства тактильного взаимодействия, для более широкого охвата используемых теоретических знаний [5].

Данные технологии помогают решить проблему получения специфического профессионального опыта. Также с использованием именно цифровых технологий появляется возможность исследования обстановки реального места преступления без риска потерять важные доказательства по делу (вариант подготовки к осмотру места происшествия для следователя, ведущего производство по уголовному делу, позволяющий подобрать и отработать оптимальную тактику производства данного следственного действия).

Одним из преимуществ цифровых технологий является то, что ими можно воспользоваться в любой момент. Это, безусловно, повышает доступность обучения и помогает улучшить его качество. Хорошо освоенные, благодаря цифровым технологиям, тактические приемы подготовки и производства осмотра места происшествия способны оказать существенную помощь в оценке следователем сложившейся следственной ситуации, а также помочь ему лучше сориентироваться на практике в реальных ситуациях расследования уголовного дела и,

соответственно, минимизировать риски своих потенциальных тактических ошибок.

ТЕХНОЛОГИИ ВИРТУАЛИЗАЦИИ ЗАПАХА

На данный момент существуют две технологии виртуализации запаха, обе находятся в стадии разработки. Одна из них – это стимуляция запахов с помощью генерации электрического тока [6]. В нос помещаются электроды, которые симулируют обонятельные рецепторы, и у пользователя устройства возникает ощущение, что он чувствует запах. Данная технология очень перспективна, но она еще не доработана, и использовать ее для данного проекта невозможно.

Мы остановились на устройствах, которые генерируют реальный запах [7], а не его чувство. У них есть несколько недостатков – это сложность получения чистого аромата, так как пока не выветриться предыдущий, он будет смешиваться с каждым последующим запахом. Также такие устройства поддерживают ограниченное количество картриджей, которые требуют регулярной смены. Но несмотря на все эти минусы, они уже производятся и используются. Для нашей цели данные устройства также подойдут, так как при создании эффекта погружения для будущего криминалиста большого набора запахов не потребуется.

УСТРОЙСТВА ДЛЯ ГЕНЕРАЦИИ ЗАПАХА

Существует множество устройств для генерации запахов, мы рассмотрели несколько из них и выбрали подходящее.

Olorama – это очки виртуальной реальности, включающие в себя 10 беспроводных ароматизирующих устройств. Но это достаточно дорогое устройство, которое вряд ли подойдет студентам [8].

FEELREAL VR Mask представляет собой маску, которая подсоединяется через Bluetooth-интерфейс к VR-очкам. Конструкция массивна и не очень удобна [9].

OhRoma – данное устройство выглядит как респиратор, который крепится к очкам виртуальной реальности и источает крошечные порции ароматических веществ прямо в нос пользователя.

Мы выбрали VAQSO VR [10], данный генератор запахов выглядит как небольшая коробочка, закрепляемая внизу любого VR-шлема и распыляющая

ароматы перед носом пользователя (Рис. 1). Немаловажно, что стоит это устройство недорого, соответственно, оно подойдет для студентов и школьников.

Рис. 1. VAQSO

ОСМОТР МЕСТА ПРОИСШЕСТВИЯ

Для создания эффекта погружения пользователя в окружающую среду была проведена работа по исследованию растительности и почвы, возможных на конкретной территории осмотра места происшествия.

Рис. 2. Скриншот виртуального места происшествия

Были созданы и подобраны соответствующие ассеты[3], таким образом, получилось создать реалистичную картину окружающей среды кладбища, на котором проходит виртуальный осмотр места происшествия (Рис. 2). Также было проработано графическое окружение (Рис. 3): могильные камни, ограждения, лавки и урны, соответствующие тематике кладбища, выбранного для создания виртуальной среды региона.

[3] Ассет, от англ. asset – ресурс, используемый при разработке игрового приложения: графический, логический, звуковой или др.

Рис. 3. Скриншот виртуального кладбища

Для увеличения иммерсивности на данных о птицах и животных, которые теоретически могут обитать в регионе местонахождения кладбища, было создано звуковое сопровождение приложения.

При первом попадании в описываемую среду пользователь должен почувствовать сопутствующие запахи, такие, как запах травы, земли и так далее. Это поможет ему наиболее полно погрузится в происходящие события, для лучшего усвоения материала.

Далее по ходу развития событий запаховая картина будет сообразно изменяться (количество источников запаха и их интенсивность). Она напрямую обусловлена ситуацией моделируемого осмотра места происшествия и будет иметь обучающий характер.

Трупный запах, а именно, сероводород, метилмеркаптан и этилмеркаптан, генерируется при приближении к трупу. Данные запахи сопровождают процесс гниения и являются одним из факторов, по которому можно определить стадию разложения [11].

Рис. 4. Скриншот трупа с пулевыми отверстиями, покрытыми кровью

Запах пороха также является важным показателем, который может помочь рассчитать время выстрела. Запахи крови (Рис. 4) и пороха смешиваются с естественными запахами растительности и выкопанной земли и дают яркую ароматическую картину, без которой погружение в виртуальную среду было бы неполным.

МЕТОДИКА РАБОТЫ ЗАПАХА В МОДУЛЕ

Запахи генерируются по сигналу от модуля к устройству при совершении определенных действий, прописанных в контроллере. В результате в воздух происходит выброс аромата одного из картриджей.

Контроллер проверяет, подходит ли действие пользователя под какой-либо шаблон и в соответствии с этим решает, какой запах выпустить:

- если положение пользователя – рядом с могилой, то выпустить запах разложения тканей трупа;
- если положение пользователя – рядом со вторым трупом, то выпустить запах пороха и крови;
- если положение пользователя – не рядом с могилой и не рядом со вторым трупом, то выпустить запах травы и свежевскопанной земли.

Запаховый модуль необходимо реализовывать в абстракциях используемых запахов для возможности вызова его при других ситуациях. Упрощенная схема работы модуля представлена на Рис. 5.

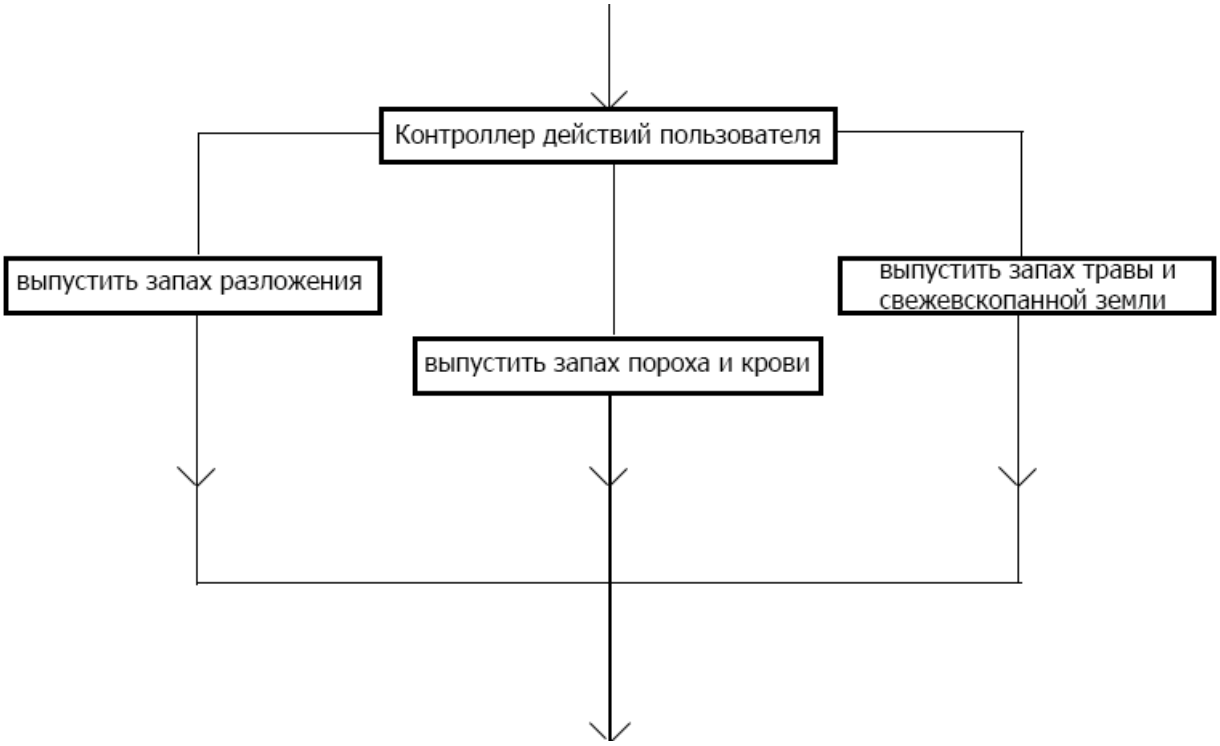


Рис. 5. Упрощенная схема работы модуля

ПЕРСПЕКТИВА ИСПОЛЬЗОВАНИЯ ЦИФРОВОГО ЗАПАХА ДЛЯ ВИРТУАЛЬНОГО ОСМОТРА МЕСТА ПРОИСШЕСТВИЯ

Существующие в настоящее время виртуальные криминалистические места происшествия не в полной мере обеспечивают эффект полного погружения и, соответственно, в результате их применения могут быть упущены важные моменты при обучении. В этой связи перспектива использования цифрового запаха для виртуального осмотра места происшествия выглядит очень широкой. К тому же данные технологии позволяют применять их не только при обучении студентов (курсантов), но и в качестве поддержания и развития профессиональных навыков у действующих следователей. В дальнейшем возможно расширение круга проходящих обучение с использованием данного модуля (оперативники, специалисты, участвующих в производстве следственных действий).

Также данный метод можно использовать в других сферах, например, при разработке обучающих симуляторов в медицине, химической промышленности, фармацевтике, биологии, экологии и т. д. или применять при оценке характеристик различных промышленных продуктов [12].

После доработки система [13] сможет реализовать естественный интерфейс на основе жестов и мультисенсорные стимулы, включая визуальные эффекты, звук, запахи и климатические эффекты. Тогда модуль будет актуален и для виртуальных туров, где человек, сможет полностью погрузиться в виртуальную реальность и пережить практически реальное туристическое путешествие.

Данный метод, наравне с другими методами усиления погружения в виртуальную реальность, увеличивает эмоциональную восприимчивость [14], значит, пользователь данной системы намного лучше запомнит информацию, что в свою очередь может помочь при обучении в любой сфере, где возможно использование виртуальной реальности.

ЗАКЛЮЧЕНИЕ

Программирование запахов для виртуального осмотра мест происшествий способствует более полному погружению в процесс производства расследования уголовного дела, возможности применить и отточить на практике изученные в теории приемы, которым в реальной жизни сложно найти место применения, если не совершено преступление.

Разработанный метод будет в дальнейшем развиваться, результаты его применения будут изучены и проанализированы.

СПИСОК ЛИТЕРАТУРЫ

1. *Абрамов В.Д., Кугуракова В.В., Ризванов А.А., Абрамский М.М., Манахов Н.Р., Евстафьев М.Е.* Виртуальные лаборатории как средство обучения биомедицинским технологиям // Электронные библиотеки. 2016. Т. 19. №3. С. 129–148.

2. *Kugurakova V.V., Elizarov A.M., Khafizov M.R., Lushnikov A.Yu., Nizamutdinov A.R. Towards* the Immersive VR: Measuring and Assessing Realism of User Experience // The 2018 Int. Conf. on Artificial ALife and Robotics. AROB 23nd Anniversary (ICAROB 2018).

3. Криминалистическая одорология. URL: http://crimlib.info/Одорология

4. *Serrano B., Baños R.M. and Botella C.* Virtual reality and stimulation of touch and smell for inducing relaxation: A randomized controlled trial// Computers in Human Behavior. 2016. Vol. 55. P. 1–8.

5. *Kugurakova V., Khafizov M., Akhmetsharipov R., Lushnikov A., Galimova D., Abramov V., Correa O.* Virtual Surgery System with Realistic Visual Effects and Haptic Interaction // The 2017 Int. Conf. on Artificial ALife and Robotics. AROB 22nd Anniversary (ICAROB 2017). Japan, Miyazaki, January 19–22, 2017.

6. *Surina Hariri, Nur Ain Mustafa, Kasun Karunanayaka and Adrian David Cheok.* Electrical Stimulation of Olfactory Receptors for Digitizing Smell // MVAR '16 Proc. of the 2016 Workshop on Multimodal Virtual and Augmented Reality. Tokyo, Japan. November 16, 2016.

7. *Daniel Harley, Alexander Verni, Mackenzie Willis, Ashley Ng, Lucas Bozzo and Ali Mazalek.* Sensory VR: Smelling, Touching, and Eating Virtual Reality // TEI '18 Proc. of the Twelfth Int. Conf. on Tangible, Embedded, and Embodied Interaction. Stockholm, Sweden, March 18–21, 2018. P. 386–397.

8. Olorama. URL: http://www.olorama.com/virtual-and-augmented-reality-with-smells

9. FEELREAL VR Mask. URL: https://feelreal.com/

10. OhRoma. URL: https://www.camsoda.com/products/ohroma/

11. VAQSO. VR. URL: https://vaqso.com/

12. *Лелиовская А.А.* Судебная медицина. Учебное пособие. М.: Изд-во Юрайт, 2010. 250 с.

13. *Carulli M., Bordegoni M., Cugini U. and Weibin D.* A Methodology for the Analysis of the Influence of Odours on the Users' Evaluation of Industrial Products. ICoRD'15 – Research into Design Across Boundaries. 2015. Vol. 35. P. 397–407.

14. *Manghisi V.M., Fiorentino M., Gattullo M., Boccaccio A., Bevilacqua V., Cascella G.L., Dassisti M. and Uva A.E.* Experiencing the Sights, Smells, Sounds, and Climate of Southern Italy in VR// IEEE Computer Graphics and Applications. 2017. Vol. 37, No. 6. P. 19–25.

15. *Rodríguez A., Rey B. and Alcañiz M.* Immersive Virtual Environments for Emotional Engineering: Description and Preliminary Results// Annual Review of CyberTherapy and Telemedicine. 2011. V. 9, No. 1. P. 161–164.

SMELLS' PROGRAMMING FOR A VIRTUAL SURVEY OF A CRIME SCENE

I. Antonov[1], K. Zezegova[2], V. Kugurakova[3], E. Lazarev[4], M. Khafizov[5]

Kazan (Volga Region) Federal University

[1]igolant@mail.ru, [2]hig154era@gmail.com, [3]vlada.kugurakova@gmail.com, [4]evgenln11401@gmail.com, [5]murkorp@gmail.com

Abstract

Virtual odor is a new technology that allows the user to plunge the user completely into virtual reality. In the paper, we propose a method of using virtual odor in learning systems for criminalists. We also describe the scheme of this method.

Key words: *virtual relation, VR, immersive, virtual odor, virtual smell, digital odor, digital smell, criminology, learning criminologists, virtual simulation*

REFERENCES

1. *Abramov V., Kugurakova V., Rizvanov A., Abramskiy M., Manahov N., Evstafiev M.* Virtual laboratories for biomedical professional education// Russian Digital Libraries Journal. 2016. V. 19. No 3. P. 129–148.

2. *Kugurakova V.V., Elizarov A.M., Khafizov M.R., Lushnikov A.Yu., Nizamutdinov A.R. Towards* the Immersive VR: Measuring and Assessing Realism of User Experience // The 2018 Int. Conf. on Artificial ALife and Robotics. AROB 23nd Anniversary (ICAROB 2018).

3. Criminalistic odoralogy. URL: http://crimlib.info/

4. *Serrano B., Baños R.M. and Botella C.* Virtual reality and stimulation of touch and smell for inducing relaxation: A randomized controlled trial// Computers in Human Behavior. 2016. Vol. 55. P. 1–8.

5. *Kugurakova V., Khafizov M., Akhmetsharipov R., Lushnikov A., Galimova D., Abramov V., Correa O.* Virtual Surgery System with Realistic Visual Effects and Haptic Interaction // The 2017 Int. Conf. on Artificial ALife and Robotics. AROB 22nd Anniversary (ICAROB 2017). Japan, Miyazaki, January 19–22, 2017.

6. *Surina Hariri, Nur Ain Mustafa, Kasun Karunanayaka and Adrian David Cheok.* Electrical Stimulation of Olfactory Receptors for Digitizing Smell // MVAR '16 Proc. of the 2016 Workshop on Multimodal Virtual and Augmented Reality. Tokyo, Japan. November 16, 2016.

7. *Daniel Harley, Alexander Verni, Mackenzie Willis, Ashley Ng, Lucas Bozzo and Ali Mazalek.* Sensory VR: Smelling, Touching, and Eating Virtual Reality // TEI '18 Proc. of the Twelfth Int. Conf. on Tangible, Embedded, and Embodied Interaction. Stockholm, Sweden, March 18–21, 2018. P. 386–397.

8. Olorama. URL: http://www.olorama.com/virtual-and-augmented-reality-with-smells

9. FEELREAL VR Mask. URL: https://feelreal.com/

10. OhRoma. URL: https://www.camsoda.com/products/ohroma/

11. VAQSO. VR. URL: https://vaqso.com/

12. *Leliovskaya A.A.* Forensic medicine. Tutorial M.: Yurait, 2010. 250 s.

13. *Carulli M., Bordegoni M., Cugini U. and Weibin D.* A Methodology for the Analysis of the Influence of Odours on the Users' Evaluation of Industrial Products. ICoRD'15 – Research into Design Across Boundaries. 2015. Vol. 35. P. 397–407.

14. *Manghisi V.M., Fiorentino M., Gattullo M., Boccaccio A., Bevilacqua V., Cascella G.L., Dassisti M. and Uva A.E.* Experiencing the Sights, Smells, Sounds, and Climate of Southern Italy in VR// IEEE Computer Graphics and Applications. 2017. Vol. 37, No. 6. P. 19–25.

15. *Rodríguez A., Rey B. and Alcañiz M.* Immersive Virtual Environments for Emotional Engineering: Description and Preliminary Results// Annual Review of CyberTherapy and Telemedicine. 2011. V. 9, No. 1. P. 161–164.

СВЕДЕНИЯ ОБ АВТОРАХ

АНТОНОВ Игорь Олегович – кандидат юридических наук, доцент, заведующий кафедрой уголовного процесса и криминалистики Юридического факультета Казанского (Приволжского) федерального университета.

Igor Olegovich ANTONOV – PhD in Juridical sciences, Associate Professor, Head of the Department of Criminal Procedure and Criminalistics, Faculty of Law, Kazan (Privolzhsky) Federal University.

email: igolant@mail.ru

ЗЕЗЕГОВА Ксения Васильевна – бакалавр, закончила Высшую школу информационных технологий и интеллектуальных систем в 2017 году. Сфера интересов: программирование иммерсивных виртуальных сред.

Ksenia Vasilievna ZEZEGOVA – Bachelor, graduated from the Higher School of Information Technology and Intelligent Systems in 2017. Sphere of interests: programming immersive virtual environments.

email: hig154era@gmail.com

КУГУРАКОВА Влада Владимировна – старший преподаватель Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета, руководитель Научно-исследовательской лаборатории «Виртуальные и симуляционные технологии в биомедицине».

Vlada Vladimirovna KUGURAKOVA, senior Lecturer of Higher School of Information Technology and Intelligent Systems, Head of Laboratory "Virtual and simulational technologies in biomedicine".

email: vlada.kugurakova@gmail.com

ЛАЗАРЕВ Евгений Николаевич – лаборант-исследователь Научно-исследовательской лаборатории «Виртуальные и симуляционные технологии в биомедицине» Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета. Сфера интересов: проектирование левел-дизайна, новые подходы к созданию иммерсивных виртуальных сред.

Evgeniy Nikolaevich LAZAREV – researcher of Scientific Laboratory "Virtual and simulational technologies in biomedicine.

email: evgenln11401@gmail.com

ХАФИЗОВ Мурад Рустэмович – ведущий технический специалист в направлении разработки цифровых визуализаций и виртуальных систем в научно-исследовательской лаборатории «Виртуальные и симуляционные технологии в биомедицине» Высшей школы информационных технологий и информационных систем Казанского (Приволжского) федерального университета.

Murad Rustemovich KHAFIZOV, lead software engineer in digital visualizations and virtual systems development of Laboratory "Virtual and simulational technologies in biomedicine" of Kazan Federal University.

email: murkorp@gmail.com

Материал поступил в редакцию 1 мая 2018 года

УДК 004.436.4+004.415.2+004.434:004.94

КОНФИГУРИРОВАНИЕ ВЕБ-ПРИЛОЖЕНИЙ НА ОСНОВЕ ДИАГРАММ СОСТОЯНИЙ UML

И. А. Габидуллин[1], А. А. Марченко [2]

[1,2] *Высшая школа информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета*

[1]ibragim0795@yandex.ru, [2]marchenko@it.kfu.ru

Аннотация

Описан способ использования UML-диаграмм для конфигурирования поведения веб-приложений: при помощи конфигурации определяются поведение системы, переходы между экранами, а также логика обработки данных. Изучены и сравнены разного рода UML-диаграммы на предмет их возможностей. Разработан веб-фреймворк для ASP.NET Core, который использует UML-диаграмму для формирования конфигурации в формате файлов XML или JSON, на основе которых выстраивается поведение веб-приложения. Рассмотрены дальнейшие шаги в использовании и развитии полученного веб-фреймворка.

***Ключевые слова**: UML-диаграммы, веб-сайт, конфигурация, веб-фреймворк*

ВВЕДЕНИЕ

Современное программное обеспечение в виде веб-сайтов или приложений для операционных систем становится всё сложнее. Большинство веб-приложений наполнено разнообразным функционалом, и его становится только больше как в корпоративных системах, так и в широко распространённых веб-сайтах для пользователей интернета. Чем шире функционал, тем в большее количество состояний может перейти программное обеспечение.

С другой стороны, на начальной стадии разработки программного обеспечения удобно использовать такой инструмент, как UML (Unified Modeling Language) [1]. Этим языком моделирования могут быть описаны разные характеристики (начиная от модели данных, заканчивая поведением системы) разрабатываемого программного обеспечения в виде диаграмм. На основании некото-

рых UML-диаграмм, например, структурных (то есть модели данных), можно при помощи специальных UML-редакторов создавать шаблоны исходного кода, которые в дальнейшем можно преобразовать в рабочий программный код. Эти возможности поддерживаются некоторыми редакторами UML [2, 3]. Сейчас ведутся работы в направлении того, чтобы на основе поведенческих UML-диаграмм генерировать уже готовый исходный код программного продукта [4, 5], но очевидно, что при такой генерации получается неоптимальный, возможно нечитабельный и неочевидный программный код. Работы в данном направлении до сих пор продолжаются.

В статье предложен другой подход: использовать поведенческую UML-диаграмму для конфигурирования приложения. Для этого разработан веб-фреймворк, который использует UML-диаграмму, описывающую поведение веб-приложения для конфигурирования разрабатываемого веб-приложения. Таким образом, диаграмма состояний UML будет преобразована в конфигурацию системы. Для разработки были выбраны язык программирования C# и ASP.NET Core, потому что, во-первых, C# занимает в индексе Tiobe высокую позицию [6], а, во-вторых, ASP.NET Core используется как в корпоративной, так и индивидуальной разработках.

1. ДИАГРАММА СОСТОЯНИЙ UML

Пример такой диаграммы можно увидеть на Рис. 1. Оригинальное название – UML State Machine или UML Statechart.

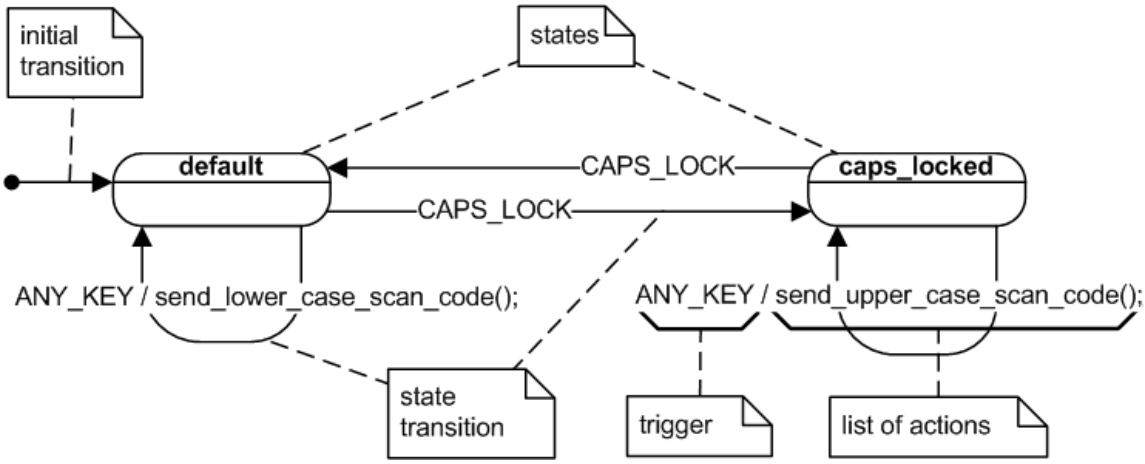


Рис. 1. Пример работы клавиатуры, описанный при помощи UML State Chart

Диаграмма состояний UML является развитием идей диаграммы состояний из теории автоматов. Диаграмма состояний – это направленный граф, вершины которого представляют состояние системы, а рёбра – это переходы между состояниями. Ключевой особенностью является то, что здесь применяется событийно-ориентированный подход. Изменение состояния системы происходит под действием каких-либо событий. Если рассматривать систему как веб-сайт, то это, например, клик мыши на какую-нибудь кнопку. Помимо этого, достоинством диаграммы состояний является то, что каждое состояние определяет специфическое поведение системы. Состоянием описывается бизнес-логика приложения. Таким образом, с помощью данной диаграммы можно спроектировать сложную систему, в которой будут строго организованные переходы и строго организованная обработка данных. Использование такой структуры позволяет добиться чёткости и явного поведения системы, когда на каждое событие происходит какое-либо действие.

UML-редакторы для хранения диаграмм используют специальный язык разметки XML (eXtensible Markup Language) [7]. Для UML существует специальный подкласс XML – XMI (XML Metadata Interchange) [8]. Данный язык является очень сложным и покрывает все возможные UML-диаграммы. Большинство UML редакторов поддерживает XMI [2, 3]. Но XMI не подходит под наши задачи именно из-за его сложности; XMI, который генерируется редакторами UML, хранит в себе помимо того, что важно для разработчика, ещё и то, что необходимо редакторам для отображения UML. В итоге получается очень большой файл, который содержит много ненужной информации.

С другой стороны, существует SCXML. Данный язык создан для поддержки как раз UML State Chart. Он может быть сгенерирован некоторыми UML-редакторами [2]. Язык отличается хорошей читаемостью. В то время как XMI разработан для программ, SCXML разработан скорее для передачи информации между людьми. Его может читать и понимать практически не подготовленный специалист. SCXML-файлы генерируются на основе UML State Chart.

Существует конвертер, который генерирует SCXML-файл на основе XMI [9]. Так же, как и XMI, SCXML-конвертер создан на основе XSLT (XSL Transformations) – язык преобразований XML-документов. Значит, можно путем преобразований

получить из XMI-файла SCXML-файл. В итоге для дальнейшей работы нами был выбран SCXML-язык. В разработке языка конфигурации системы мы отталкивались именно от SCXML.

2. ЯЗЫК КОНФИГУРАЦИИ СИСТЕМЫ

SCXML, несмотря на все достоинства, имеет громоздкую структуру с большим количество вложенности. Поэтому был разработан язык конфигурации. Также причиной новой разработки является оптимизация: исключены все лишние элементы, что освобождает память при работе с файлом конфигурации.

Были разработаны два вида языка конфигураций: на основе JSON и на основе XML. Соответственно разработчик может описать систему в том виде, в котором ему удобнее. Достоинством JSON является то, что он компактнее, чем XML. XML же, с одной стороны, более понятен для чтения, а с другой, является довольно громоздким языком разметки. В мире существуют разработчики, которые предпочитают, как XML-нотацию, так и JSON-нотацию, поэтому было принято решение поддерживать оба формата. Рассмотрим пример языка на основе JSON.

```
[
  {
    "Id": "1",
    "PreviousState": "1",
    "CurrentState": "FirstStep",
    "NextState": "2",
    "Conditions": {
      "cond1": 2,
      "cond2": 3
    }
  },
  {
    "Id": "2",
    "PreviousState": "1",
    "CurrentState": "SecondStep",
    "NextState": "3"
  },
```

Рис. 2. Пример языка конфигурации на основе JSON

На Рис. 2 мы видим часть поведения системы. Рассмотрим её подробнее.

Мы видим, что состояние «1» имеет предыдущее состояние «0», то есть самое начальное состояние. В свою очередь из состояния «1» по умолчанию происходит переход в состояние «2». Но также есть условные переходы. При определенном условии система перейдет в состояние «2», а при определенных – в состояние «3». Таким образом, описание системы покрывает как последовательные переходы, так и условные переходы, что важно в условиях комплексного программного обеспечения.

Как будет выглядеть данная конфигурация, написанная с использованием XML, можно увидеть на Рис. 3.

```
<?xml version="1.0"?>
<ArrayOfState>
  <State>
    <Id>1</Id>
    <PreviousState>1</PreviousState>
    <CurrentState>FirstStep</CurrentState>
    <NextState>2</NextState>
  </State>
  <State>
    <Id>2</Id>
    <PreviousState>1</PreviousState>
    <CurrentState>SecondStep</CurrentState>
    <NextState>3</NextState>
  </State>
```

Рис. 3. Пример языка конфигурации на основе XML

На данный момент язык конфигурации поддерживает основные возможности для разработки программного обеспечения.

3. ПРОГРАММНОЕ РЕШЕНИЕ ДЛЯ ВЗАИМОДЕЙСТВИЯ С КОНФИГУРАЦИЕЙ

Рассмотрим программное решение. Веб-фреймворк взаимодействует с базовым подходом .NET Core – MVC, то есть реализована схема разделения принципа работы на основании – «Модель – Представление – Контроллер». Разработанное программное обеспечение используется на уровне «Контроллер».

Программное решение представляет собой несколько классов, таких, как:

1. FlowService – является самой важной частью фреймворка, именно этот класс решает, в какое следующее состояние переходит система;
2. JsonWorker или XmlWorker – реализуют интерфейс IFlowWorker; эти классы обеспечивают работу с файлами конфигурации, написанными соответственно на языках JSON и XML;
3. FlowData – это модель данных; после считывания классы из второго пункта хранят данные в этом файле.

Класс FlowService регистрируется в конфигурации ASP.NET Core в виде сервиса. После этого в Контроллере веб-приложения вызывается метод данного класса, для перехода в следующее состояние. В свою очередь Контроллер по возвращаемому значению из метода определяет, какие действия будут выполняться в дальнейшем.

FlowService взаимодействуют как с файлом конфигурации, так и с базой данных веб-приложения. В ней хранятся история пользователя (на каком именно состоянии остановилось приложение), а также история переходов между состояниями (для возврата, при необходимости, в предыдущие состояния).

Одним из достоинств такой системы является то, что можно добавлять новый функционал прямо из системы, таким образом, мы можем создать приложение, которое может конфигурировать само себя. Данный функционал важен для корпоративных систем, в которых существует необходимость настроить приложение для каждой из ролей.

На текущий момент похожих разработок для ASP.NET Core обнаружено не было. Такого рода разработки есть для языка программирования Java Script [10, 11]. Их недостатком является то, что программа будет выполняться на стороне пользователя, значит, существует возможность для ее модификации пользователем, а это может повлиять на безопасность веб-приложения. Также существует похожий по идеям фреймворк Spring Web Flow для языка программирования Java [12], в нем также используется конфигурация в виде XML-файла. Недостатком является то, что этот фреймворк разработан в 2008 году и не поддерживает современные возможности веб-приложений. В свою очередь, так как мы используем современный веб-фреймворк ASP.NET Core, это обеспечивает нам поддержку современных возможностей веб-приложений.

ЗАКЛЮЧЕНИЕ

На данный момент времени решение, которое существует, позволяет создавать веб-приложения, которые включают в себя большое количество переходов между состояниями системы. Данный веб-фреймворк подходит для таких приложений, в которых есть чёткая структура переходов между состояниями системы, например, какой-нибудь тест, экзамен или же такие действия, как покупка билетов или бронирование отеля, где на каждом отдельном шаге необходимо вводить какие-либо данные.

В дальнейшем планируется развитие как языка конфигурации, так и самого веб-фреймворка. Будет продолжено изучение разметки SCXML на предмет её возможностей, а также на предмет того, какой еще функционал можно будет привнести. Планируется развитие веб-фреймворка в сторону создания веб-приложений, которые могут конфигурировать сами себя. Помимо этого, планируется разработать веб-приложение для апробации фреймворка, представленного выше.

СПИСОК ЛИТЕРАТУРЫ

1. Unified Modeling Language. URL: http://www.uml.org/
2. Enterprise Architect. URL: http://www.sparxsystems.com/products/ea/
3. Visual Paradigm. URL: http://www.visual-paradigm.com/
4. *Yusufu M., Zhang H.J., Yusufu G., Liu Z.D., Cheng P., Dilisati D.* Modeling and Analysis of Complex System with UML: A Case Study // Applied Mechanics and Materials, January 2014. V. 513–517. P. 1346–1351.
5. *Pham V.C., Radermacher A., Gerard S., Li S.* Complete code generation from UML state machine // 5th Int. Conf. on Model-Driven Engineering and Software Development, Porto, Portugal, 19–21 Feb. 2017. P. 208–219.
6. TIOBE Index. URL: http://tiobe.com/tiobe-index/
7. Extensible Markup Language (XML). URL: https://www.w3.org/XML/
8. XML Metadata Interchange (XMI). URL: https://www.omg.org/spec/XMI/
9. XMI to SCXML Converter. URL: http://github.com/apache/commons-scxml/blob/master/extras/xmi2scxml.xsl/
10. Next-gen state management based on Harel Statechart and SCXML. URL: http://github.com/aksonov/statem/

11. State Machine Cat. URL: http://github.com/sverweij/state-machine-cat/

12. *Vervaet E.* The Definitive Guide to Spring Web Flow. Berkeley: Apress, 2008. 380 p.

CONFIGURING WEB APPLICATIONS BASED ON UML STATE MACHINE DIAGRAM

I. A. Gabidullin[1], A. A. Marchenko[2]

[1,2] *Higher School of Information Technologies and Intelligent Systems at Kazan (Volga region) Federal University*

[1]ibragim0795@yandex.ru, [2]marchenko@it.kfu.ru

Abstract

In this paper, we describe a way to use UML diagrams for configuring web-application's behavior. That is, using the configuration will determine the behavior of the system, the transitions between the screens, as well as the business logic of application. We examine and compare various UML diagrams for their possible features. A web framework for ASP.NET Core is developed, which uses the UML diagram to form a configuration in the XML or JSON file formats. Configuration will determine the behavior of the system. In addition, we describe further steps in using and developing the web-framework.

Keywords*: UML diagram, web site, configuration, web-framework*

REFERENCES

1. Unified Modeling Language. URL: http://www.uml.org/
2. Enterprise Architect. URL: http://www.sparxsystems.com/products/ea/
3. Visual Paradigm. URL: http://www.visual-paradigm.com/
4. *Yusufu M., Zhang H.J., Yusufu G., Liu Z.D., Cheng P., Dilisati D.* Modeling and Analysis of Complex System with UML: A Case Study // Applied Mechanics and Materials, January 2014. V. 513–517. P. 1346–1351.

5. *Pham V.C., Radermacher A., Gerard S., Li S.* Complete code generation from UML state machine // 5th Int. Conf. on Model-Driven Engineering and Software Development, Porto, Portugal, 19–21 Feb. 2017. P. 208–219.

6. TIOBE Index. URL: http://tiobe.com/tiobe-index/

7. Extensible Markup Language (XML). URL: https://www.w3.org/XML/

8. XML Metadata Interchange (XMI). URL: https://www.omg.org/spec/XMI/

9. XMI to SCXML Converter. URL: http://github.com/apache/commons-scxml/blob/master/extras/xmi2scxml.xsl/

10. Next-gen state management based on Harel Statechart and SCXML. URL: http://github.com/aksonov/statem/

11. State Machine Cat. URL: http://github.com/sverweij/state-machine-cat/

12. *Vervaet E.* The Definitive Guide to Spring Web Flow. Berkeley: Apress, 2008. 380 p.

СВЕДЕНИЯ ОБ АВТОРАХ

ГАБИДУЛЛИН Ибрагим Анварович – магистрант Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета.

Ibragim Anvarovich GABIDULLIN – has master's degree of the Higher School of Information Technologies and Intelligent Systems at Kazan (Volga region) Federal University.

email: ibragim0795@yandex.ru

МАРЧЕНКО Антон Александрович – ассистент кафедры Программной инженерии Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета.

Anton Aleksandrovich MARCHENKO – assistant of the Department of Program Engineering of the Higher School of Information Technologies and Intelligent Systems at Kazan (Volga region) Federal University.

email: marchenko@it.kfu.ru

Материал поступил в редакцию 28 мая 2018 года

УДК 004.387

СИНХРОНИЗАЦИЯ ДВИЖЕНИЙ ИГРОКА И ВИРТУАЛЬНОГО АВАТАРА

П. Д. Гришков[1], В. В. Кугуракова[2]

[1,2]*Высшая школа информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета*

[1]grishkovpavel@gmail.com, [2]vlada.kugurakova@gmail.com

Аннотация

Представлены математические подходы для реализации методов по синхронизации действий человека и виртуального аватара, с использованием инверсной кинематики. Для создания полноценной системы синхронизации поведения игрока и VR-аватара описана реализация необходимого для этого функционала: позиционирование рук, калибровка их размера, сгибание рук в анатомически приемлемые стороны, анатомическое сгибание позвоночника, приседание и перемещение в пространстве. Реализация наклона и приседания значительно расширяет функционал синхронизации поведения игрока и его аватара, что позволяет создать полный набор визуальных самоощущений пользователя, находящегося в виртуальной среде, чего лишено большинство приложений виртуальной реальности на данный момент.

***Ключевые слова**: виртуальная реальность, VR, VR-аватар, инверсная кинематика*

ВВЕДЕНИЕ

Очки виртуальной реальности ограничивают поле зрения пользователя. Одним из побочных эффектов этого факта является то, что пользователь не может видеть собственное тело. Люди на протяжении всей своей жизни видят собственное тело и не задумываются о важности этого факта. Оказавшись в виртуальной реальности, человек в большинстве случаев может наблюдать только кисти рук, в худшем случае – лишь изображение контроллеров. Из-за этого пропадает ощущение реальности происходящего, и пользователь сразу осознаёт, что находится в искусственно созданном пространстве, что может плохо отразиться

на целях, которые преследует определённое приложение. Например, если человек, проходя обучение на виртуальном симуляторе ремеслу, опасному для человеческой жизни, будет без должной оценки ситуации производить критические действия, впоследствии он может повторять подобное в рамках своей работы, что может привести к трагедии. Таким образом, для систем виртуальной реальности, например [1, 2], немаловажным фактором является представление собственного тела при взгляде от первого лица.

Иммерсивность в виртуальной реальности – коэффициент погружения человека в виртуальный мир – показывает, насколько реалистично человек воспринимает искусственный мир. Достижение высокой иммерсивности в виртуальной реальности достигаются с помощью использования передовых технологий изображения, звука, осязания и др., например, чувство осязания в VR достигается с помощью имитации тактильных ощущений через специальные перчатки [3]. Возможность механически воздействовать на виртуальные объекты дополняют картину нового мира пользователя.

Визуальное представление собственного тела было требованием систем виртуальной реальности почти с момента их создания. Еще в конце 1980-х годов VPL Research экспериментировала с нереалистичными аватарами [4]. Потенциальную важность визуализации аватаров для ощущения эффекта присутствия была отмечена в ранних моделях Слейтера и Уилбера [5]. Слейтер продемонстрировал, что виртуальное тело оказывает значительное влияние на самовосстановление эффекта присутствия при перемещениях [6]. Лин [7] показал, что аватар, который был правильно откалиброван от пола до высоты участника эксперимента, увеличил точность оценок высоты. Люди, правильно оценивающие пропорции мира, с меньшей вероятностью упадут с уступа. Таким образом, полезность аватаров достаточно широко изучена.

Технология виртуальной реальности в ближайшем будущем откроет новые границы в общении людей на расстоянии. Появляется всё большее количество приложений, позволяющих видеть и слышать друг друга не просто на видеозаписи, а уже в трехмерном пространстве. Применение виртуальных аватаров является неотъемлемой частью подобных систем, ведь в социальных ситуациях пользователи могут эффективно их использовать для обмена информацией. Мимика, жесты и положение тела используются как средства общения, в лю-

бых прямых взаимодействиях между людьми. Например, Доддс с соавторами [8] показал, что аватар – полезный ресурс в общении с другими людьми в многопользовательских системах виртуальной реальности.

Подготовка к работе в труднодоступных или опасных для жизни человека местах, к работе с узкоспециализированным оборудованием [1] или со сложными технологическими процессами, отработка моторных навыков для обучения проведению хирургическим операциям [2] возможна с помощью VR-тренажеров. В настоящее время всё большее количество крупных компаний занимается разработкой подобных систем для обучения и повышения квалификации персонала. Для решения подобных задач необходима повышенная способность пользователя выполнять определенные когнитивные задачи.

ПОСТАНОВКА ПРОБЛЕМЫ

В настоящее время присутствует ряд проблем, связанных с отражением пользователя и синхронизацией его действий в виртуальной среде:

- сложно сделать аватар похожим на настоящее тело пользователя;
- проблема uncanny valley – аватары распознаются по несвойственным человеку мимике и моторике, даже если 3D-модель полностью совпадает внешне с реальным человеком, прототипом аватара [9];
- отсутствует система полноценных анимаций для убедительного повторения движений человека;
- существуют риски в неправильной калибровке аватара; это приведёт к расхождениям между действиями пользователя и его виртуального аватара; пользователю будет некомфортно взаимодействовать с миром, и в связи с этим теряется эффект присутствия.

Основной задачей данной работы является реализация методов по синхронизации действий человека и виртуального аватара.

АНАЛИЗ СУЩЕСТВУЮЩИХ РЕШЕНИЙ

Все очки (шлемы) виртуальной реальности связывает факт наличия акселерометра, гироскопа и магнитометра. При помощи этих аппаратных решений разработчику можно определить угол поворота очков на голове человека, век-

тор перемещения очков в пространстве и перемещать игрового персонажа, исходя из этих данных.

Контроллеры для очков виртуальной реальности, так же, как и очки, оснащены акселерометром, гироскопом и магнитометром. В дополнение к этим трём датчикам трекинг контроллеров в пространстве происходит с помощью станций глубинного захвата и магнитных систем.

Существует несколько решений перчаток для виртуальности, таких, как CaptoGlove, Manus VR, Gloveone, Control VR, различные решения от CyberGlove Systems [10]. Средства захвата движения, основанные на перчатках, стали популярными в конце 1980-х как способ взаимодействия людей с виртуальными средами, позволяя воспроизводить жесты всей руки. MIT-LED была одной из первых пар перчаток, специально предназначенных для отслеживания движения рук для записи анимаций [11].

Kinect – это бесконтактный игровой контроллер, способный посредством двух сенсоров глубины и цветной видеокамеры распознавать полное трехмерное движение тела человека в пространстве. Слабо подходит для VR, так как не способен распознавать развороты человека на 180 градусов и имеет очень большие погрешности в трекинге конечностей [12].

Leap Motion – технология, предназначенная для захвата мелкой моторики человека. Критичными проблемами для использования Leap Motion в VR являются необходимость держать руки непосредственно перед датчиком в достаточно ограниченной области и неидеальные алгоритмы определения кистей рук – под некоторыми углами руки фактически перестают определяться.

Motion Capture – технология захвата движений. Маркерная система использует костюм с датчиками, надеваемый на человека, и множество камер, которые фиксируют данные с датчиков. Безмаркерная система не требует специального костюма с датчиками, распознаёт движения с помощью технологий компьютерного зрения и распознавания образов, однако предоставляет очень зашумлённую информацию, в связи с чем плохо применима в рамках виртуальной реальности.

Другие решения в синхронизации движений для аватара: Proteus VR [13] – работа Монреальской школы общественного здравоохранения – предлагает захват предметов в руку и перемещение по удочке, но пользователи видят только

кисти рук; и IKinema Runtime [14] – программное обеспечение, представляющее собой универсальный алгоритм инверсной кинематики для всего тела, который используется для создания реалистичного движения персонажей предлагает большой набор готовых инструментов для различных видов поведения персонажа – лазание, размещение ног, передвижение.

НАШ ПОДХОД

Наш подход основан на использовании методов инверсной кинематики (IK), как называют математический процесс восстановления движений объекта из других данных.

Анимированный персонаж моделируется скелетом из жестких сегментов, соединенных шарнирами, называемым кинематической цепью. Кинематические уравнения персонажа определяют связь между углами соединений скелета и его позой или конфигурацией. Задача прямой кинематической анимации использует уравнения кинематики для определения позы с учетом совместных углов. Обратная кинематическая задача вычисляет совместные углы для желаемой позы персонажа.

В игровом движке с открытым исходным кодом Unreal Engine [15], разрабатываемом компанией Epic Games с 1998 года, есть несколько способов использовать IK, например, Two Bone IK и FABRIC.

Первым подходом к созданию VR-аватара можно назвать использование стандартного VR-шаблона, содержащего базовые классы для работы с VR-шлемами, в котором реализованы возможность перемещения с помощью удочки и захват предметов. Тело пользователя ограничено кистями рук. Второй и описанный в этой работе подход – парадигма полного тела аватара с использованием IK. Для создания полноценной системы синхронизации поведения игрока и VR-аватара необходима реализация функционала позиционирования рук, калибровки их размера, их сгибания или поворотов, наклона туловища (сгибание позвоночника), приседания и перемещения в пространстве.

Позиционирование рук VR-аватара

Посредством контроллеров в игровой движок передаётся информация о перемещениях рук человека [16]. Чтобы избежать диссонанс между настоящими

руками человека и руками аватара, необходимо правильно позиционировать 3D-модели контроллеров, имеющие схожие с реальными размеры, в кистях аватара.

Размер VR-аватара

Пользователю перед началом работы с виртуальным телом необходимо произвести калибровку в T-pose (поза для универсальности подхода к заданию моделей: в положении, когда ноги соединены вместе, а руки разведены параллельно полу в разные стороны). Реалистичный размер виртуального аватара достигается определением роста и размаха рук игрока. В первую очередь задается размер 3D-модели, исходя из пропорций роста человека к росту модели. После этого изменяются размеры рук аватара. В зависимости от пропорции размаха рук человека к размаху рук 3D-модели увеличиваются или уменьшаются в равных размерах плечи и предплечья скелета модели.

Сгибание рук VR-аватара

Простейшим методом инверсной кинематики является “Two Bone IK”, он позволяет аналитически, с помощью тригонометрических тождеств вычислить вращения для двух костей (см. Рис. 1). Для работы в 3D пространстве, как и в случае VR, существует бесконечное число возможных решений для установки вращения костей.

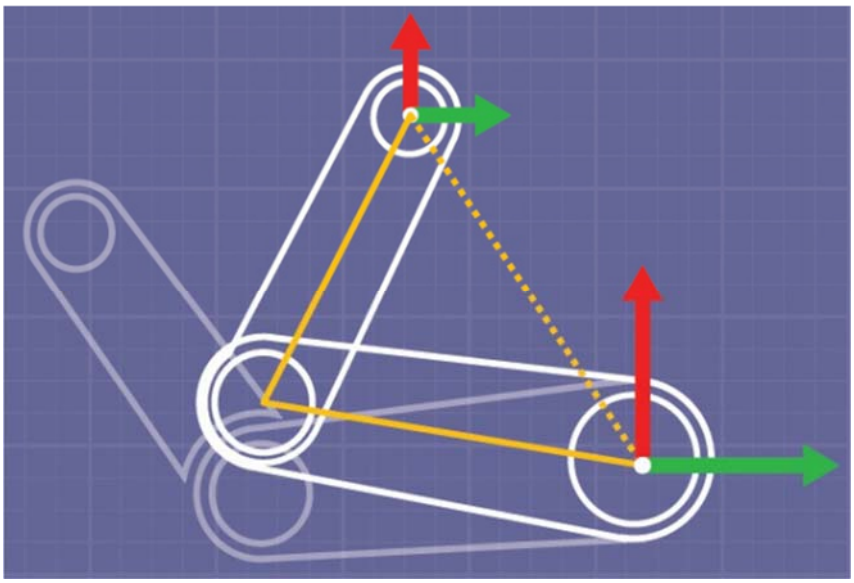


Рис. 1. Two Bone IK

Для упрощения этой задачи вводится дополнительная точка в пространстве, которая вместе с направлением от корня кости к целевой точке IK создаёт плоскость – нормаль к плоскости является векторным произведением этих двух направлений. Эта плоскость затем используется для упрощения задачи IK вплоть до двух измерений.

Сгибание позвоночника VR-аватара

Другой способ использования инверсной кинематики – Forward and Backward Reaching Inverse Kinematics (FABRIK). В отличие от реализации Two Bone IK, FABRIK не ограничивает количество костей, которые могут существовать в обратной кинематической цепи. Чтобы достичь этого, FABRIK использует несколько иной подход: вместо того, чтобы быть аналитическим, FABRIK итеративным путем прохождения вверх и вниз по цепочке костей рассчитывает повороты (Рис. 2) каждой кости.

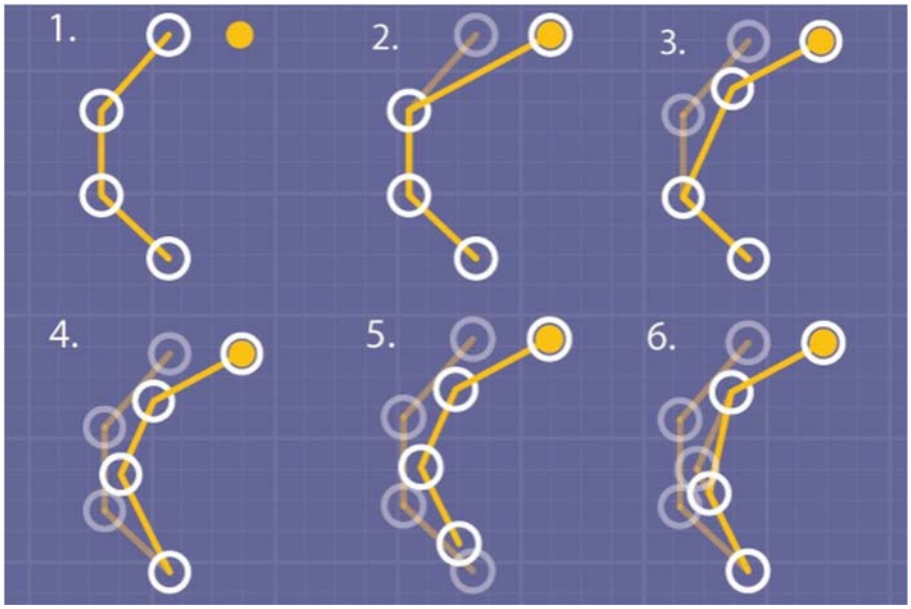


Рис. 2. Forward and Backward Reaching Inverse Kinematics

Эта реализация позволяет добиться синхронизации действий игрока с анимацией его виртуального аватара (см. Рис. 3).

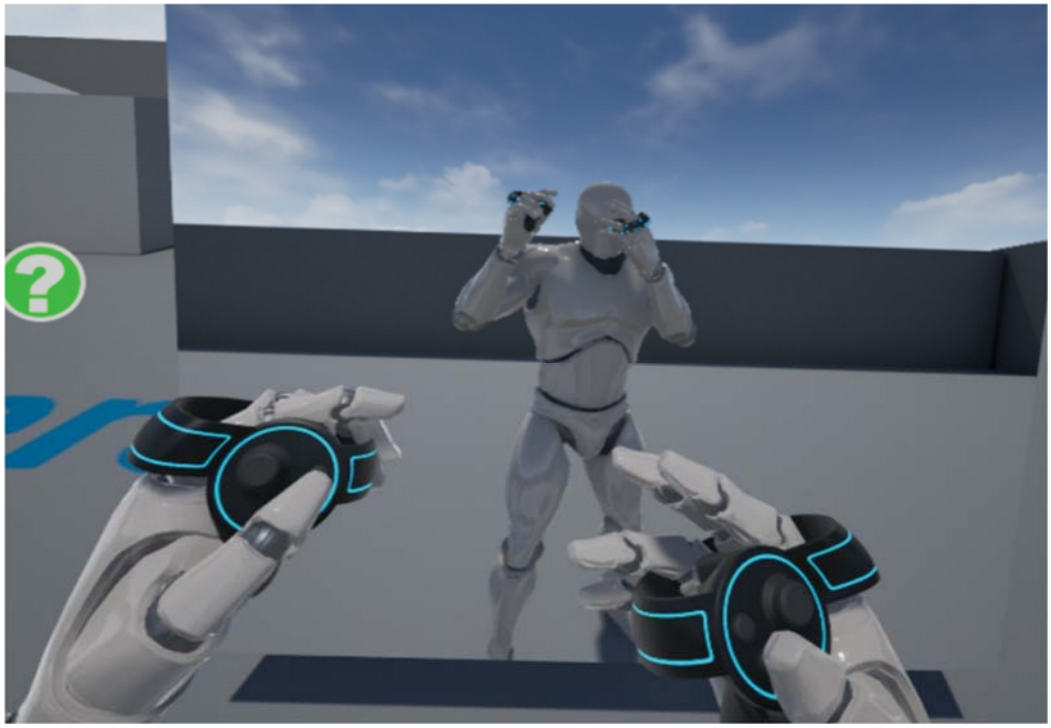


Рис. 3. IK верхней части туловища

Приседание VR-аватара

Помимо наклонов туловища, человек способен приседать (Crouching Recognition). Основываясь только на положении головы человека в пространстве, довольно сложно корректно определить, присел человек или наклонился, ведь мы не знаем, где находится его таз. Для решения этой проблемы был создан метод разграничения анимаций в зависимости от перемещений головы игрока, графически представленный на Рис. 4. На этом рисунке представлен боковой разрез, где ось X направлена по направлению таза человека, ось Z – от пола вверх. Красной точкой обозначена заранее откалиброванная высота игрока. Зелёной точкой обозначена высота таза игрока, которая равна половине высоты человека. Если в текущий момент времени голова человека находится в голубой области, будет проигрываться анимация приседания, если в красном круге – анимация наклона, если выходит за пределы круга – человек сделал шаг вперёд или назад. Область приседания ограничена формулой (параболы) $z=-x2+x+h$, где h – высота человека.

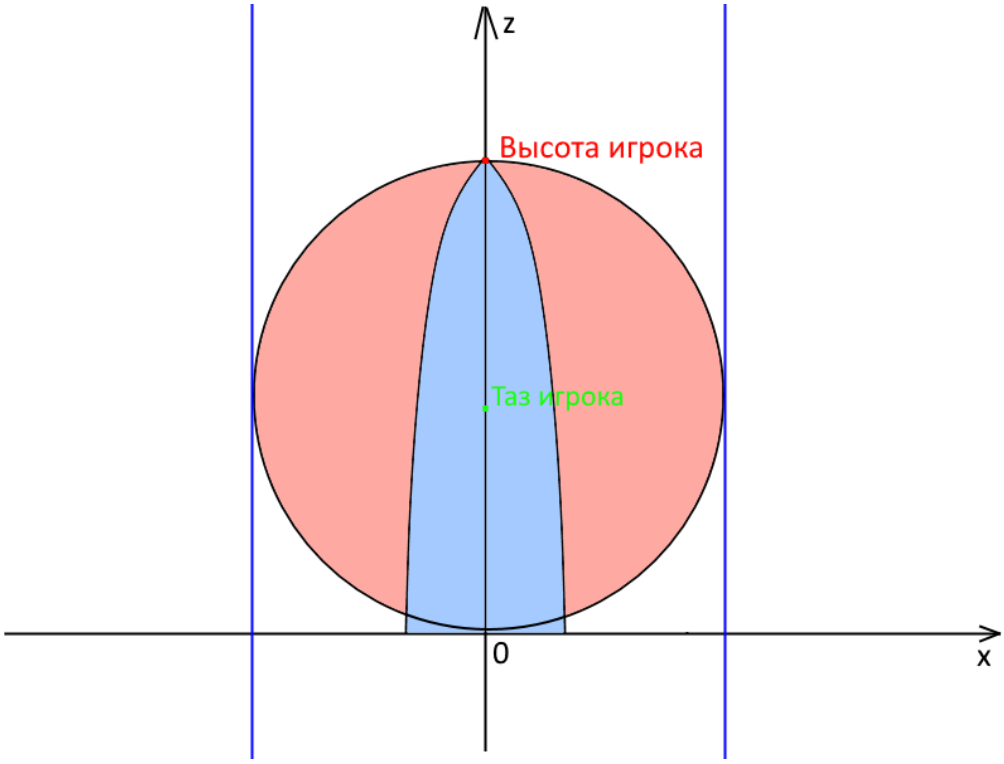


Рис. 4. IK верхней части туловища

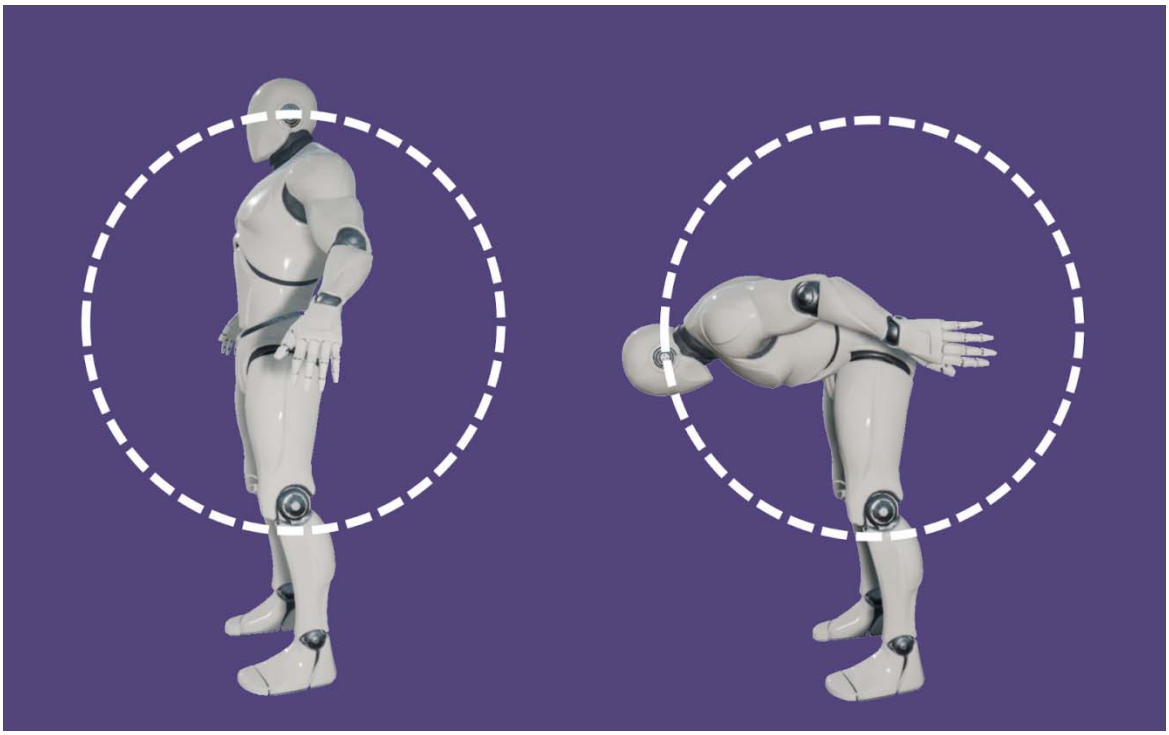


Рис. 5. Траектория максимального сгибания человека

Траектория максимального сгибания человека (Рис. 5) выводится по формуле (окружности) x2+(z-h2)2=(h2)2, где h – высота человека.

Перемещение в пространстве

Перемещение в пространстве реализовано двумя способами: телепортация или ходьба. Телепортация с помощью удочки – проверенное и популярное решение среди VR-приложений. При нажатии на джойстик на контроллере из виртуальной кисти пользователя появляется сплайн, графически отображающий кривую и точку, куда будет совершено перемещение. При вращении джойстика пользователь может выбрать направление, куда он будет смотреть после перемещения.

Ходьба используется, когда человек вышел за траекторию максимального сгибания (Рис. 5). Ноги аватара двигаются в рамках анимаций по StateMachine.

Направление таза аватара зависит от направления взгляда, положения рук относительно взгляда и тела (Рис. 6). В реалтайме рассчитывается предположительное направление таза: если оно больше, чем на 15 градусов отличается от текущего, то значения интерполируются и таз разворачивается на правильный угол. Предположительное направление равняется сумме векторов взгляда, вектора от головы до левой руки, умноженного на вес данной руки, и аналогичного вектора правой руки.

Вес руки рассчитывается в зависимости от положения руки: чем ближе она к тазу, тем он меньше. Если рука заходит за таз, её вес равен 0.

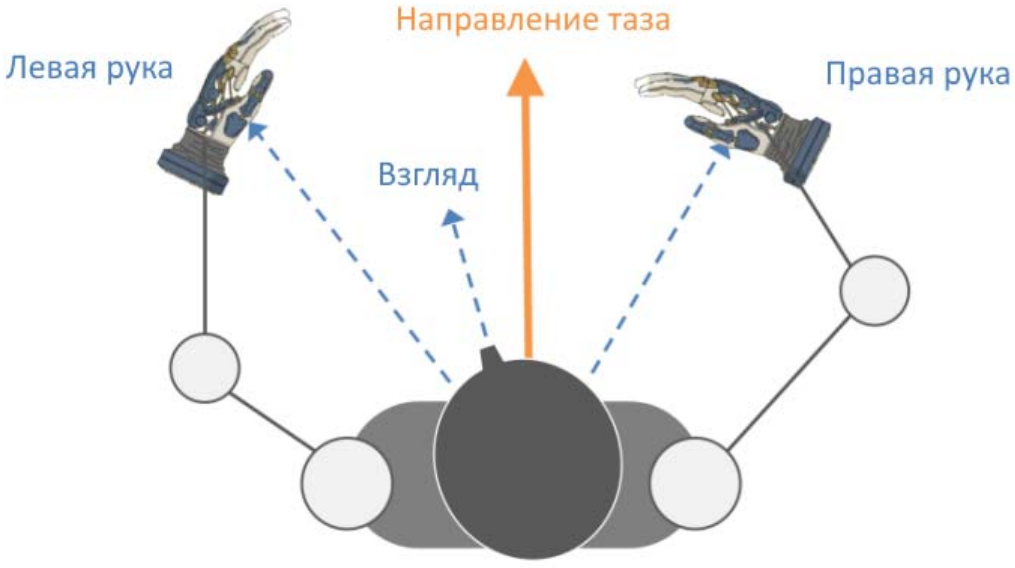


Рис. 6. Схема зависимостей при вращении аватара на месте

РЕЗУЛЬТАТЫ

Описаны математические подходы для реализации собственных методов по синхронизации действий человека и виртуального аватара:

- распознавание приседания и наклона человека;
- синхронизация нижней части туловища;
- приведены подходы правильного позиционирования рук относительно контроллеров;
- выведены зависимости анимаций от роста и размаха рук человека.

Все разработанные методы реализованы в плагине для Unreal Engine 4 [15]. Реализация наклона и приседания значительно расширяет функционал для синхронизации поведения игрока и его аватара, предлагая, таким образом, полный набор визуальных самоощущений пользователя, находящегося в виртуальной среде, чего лишено большинство приложений виртуальной реальности на данный момент. Точное позиционирование рук разработано в IKinema [14], для игрового движка Unity доступен плагин реализации перемещения в пространстве, существуют другие неполные решения. Однако такого полного комплекса методов на данный момент нет ни в одном из решений, представленных в мире.

ЗАКЛЮЧЕНИЕ

Актуальность реализованного плагина достигается тем, что всё большее количество крупных компаний занимается разработкой тренажеров виртуальной реальности для обучения и повышения квалификации персонала. Созданный виртуальный аватар способен снижать умственную нагрузку на определенные виды задач и повысить производительность при их решении [17].

Работа подчеркивает важность использования аватаров как в однопользовательских, так и в многопользовательских системах виртуальной реальности для повышения коэффициента погружения в виртуальную среду за счет визуального представления своего тела.

БЛАГОДАРНОСТИ

Работа поддержана Программой повышения конкурентоспособности КФУ и финансировалась за счет субсидии, выделенной Казанскому федеральному

университету на выполнение госзадания в области научной деятельности 0211/02.11.10122.001.

СПИСОК ЛИТЕРАТУРЫ

1. *Abramov V.D., Kugurakova V.V., Rizvanov A.A., Abramskiy M.M., Manakhov N., Evstafiev M.E., Ivanov D.S.* Virtual biotechnological labs development // BioNanoScience. 2017. Vol. 7. Iss. 2. P. 363–365.

2. *Kugurakova, V., Khafizov M., Akhmetsharipov R.* Virtual surgery system with realistic visual effects and haptic interaction // Proc. of The International Conference On Artificial Life And Robotics. 2017. P. P86–P89.

3. *Shigapov M., Kugurakov, V., Zykov E.* Design of digital gloves with feedback for VR // Proc. of IEE EWDTS. 2018.

4. *Won A.S., Bailenson J., Lee J., Lanier J.* Homuncular Flexibility in Virtual Reality // Journal of Computer-Mediated Communication. 2015. Vol. 20. No. 3. P. 241–259.

5. *Slater M., Wilbur S.* A framework for immersive virtual environments (five): Speculations on the role of presence in virtual environments // Presence: Teleoperators and virtual environments. 1997. Vol. 6. No. 6. P. 603–616.

6. *Martin Usoh M.S., Steed, A.* Taking Steps: The Influence of a Walking Technique on Presence in Virtual Reality // ACM Transactions on Computer-Human Interaction. 1995. Vol. 2. No. 3. P. 201–219.

7. *Lin Q., Rieser J.J., Bodenheimer B.* Stepping off a ledge in an HMD-based immersive virtual environment // Proc. of ACM Symposium on Applied Perception. 2013. P. 107.

8. *Dodds T.J., Mohler B.J. & Bülthoff H.H.* Talk to the virtual hands: Self-animated avatars improve communication in head-mounted display virtual environments // PLoS ONE. 2011. Vol. 6. No. 10.

9. *Kugurakova V., Talanov M., Manakhov N.* Anthropomorphic artificial social agent with simulated emotions and its implementation // 6th Annual Int. Conference on Biologically Inspired Cognitive Architecture. 2015. Vol.71. P. 112–118.

10. CyberGlove Systems. URL: http://www.cyberglovesystems.com/

11. *Wheatland N., Wan Y., Song H., Neff M., Zordan V. & Jörg S.* State of the Art in Hand and Finger Modeling and Animation // Computer Graphics Forum. 2015. Vol. 34. No. 2. P. 735–760.

12. Microsoft Kinect. URL: https://www.xbox.com/ru-RU/xbox-one/accessories/kinect

13. Proteus VR. URL: https://www.proteus-vr.com

14. IKinema. URL: https://www.ikinema.com

15. Unreal Engine. URL: http://unrealengine.com

16. *Copenhaver J.* VR Animation and Locomotion Systems in Lone Echo. URL: https://readyatdawn.sharefile.com/share/view/s80d4725de7045259

17. *Steed A., Pan Y., Zisch F. & Steptoe W.* The impact of a self-avatar on cognitive load in immersive virtual reality // Proc. of IEEE Virtual Reality. 2016. P. 67.

SYNCHRONIZATION OF PLAYER AND VIRTUAL AVATAR MOVEMENTS

P. D. Grishkov[1], V. V. Kugurakova[2]

[1,2]*Higher School of Information Technologies and Intelligent Systems. Kazan Federal University*

[1]grishkovpavel@gmail.com, [2]vlada.kugurakova@gmail.com

Abstract

The paper presents mathematical approaches for implementing methods for synchronizing human actions and virtual avatar movements, using inverse kinematics. To create a complete system for synchronizing the player's behavior and VR-avatar, the implementation of the necessary functionality is described: hand positioning, calibration of their size, bending of hands into anatomically acceptable sides, anatomical flexion of the spine, squatting and moving in space. The implementation of tilt and squat significantly extends the functionality of synchronization of the player's behavior and avatar, which allows creating a complete set of visual sensations of the user in a virtual environment, which is deprived of most of the applications of virtual reality at the moment.

Keywords*: Virtual reality, Unreal Engine, inverse kinematic, avatar, crouching recognition*

REFERENCES

1. *Abramov V.D., Kugurakova V.V., Rizvanov A.A., Abramskiy M.M., Manakhov N., Evstafiev M.E., Ivanov D.S.* Virtual biotechnological labs development // BioNanoScience. 2017. Vol. 7. Iss. 2. P. 363–365.

2. *Kugurakova, V., Khafizov M., Akhmetsharipov R.* Virtual surgery system with realistic visual effects and haptic interaction // Proc. of The International Conference on Artificial Life And Robotics. 2017. P. P86–P89.

3. *Shigapov M., Kugurakov, V., Zykov E.* Design of digital gloves with feedback for VR // Proc. of IEE EWDTS. 2018.

4. *Won A.S., Bailenson J., Lee J., Lanier J.* Homuncular Flexibility in Virtual Reality // J. of Computer-Mediated Communication. 2015. Vol. 20. No. 3. P. 241–259.

5. *Slater M., Wilbur S.* A framework for immersive virtual environments (five): Speculations on the role of presence in virtual environments // Presence: Teleoperators and virtual environments. 1997. Vol. 6. No. 6. P. 603–616.

6. *Martin Usoh M.S., Steed, A.* Taking Steps: The Influence of a Walking Technique on Presence in Virtual Reality // ACM Transactions on Computer-Human Interaction. 1995. Vol. 2. No. 3. P. 201–219.

7. *Lin Q., Rieser J.J., Bodenheimer B.* Stepping off a ledge in an HMD-based immersive virtual environment // Proc. of ACM Symposium on Applied Perception. 2013. P. 107.

8. *Dodds T.J., Mohler B.J. & Bülthoff H.H.* Talk to the virtual hands: Self-animated avatars improve communication in head-mounted display virtual environments // PLoS ONE. 2011. Vol. 6. No. 10.

9. *Kugurakova V., Talanov M., Manakhov N.* Anthropomorphic artificial social agent with simulated emotions and its implementation // 6th Annual Int. Conference on Biologically Inspired Cognitive Architecture. 2015. Vol.71. P. 112–118.

10. CyberGlove Systems. URL: http://www.cyberglovesystems.com/

11. *Wheatland N., Wan Y., Song H., Neff M., Zordan V. & Jörg S.* State of the Art in Hand and Finger Modeling and Animation // Computer Graphics Forum. 2015. Vol. 34. No. 2. P. 735–760.

12. Microsoft Kinect. URL: https://www.xbox.com/ru-RU/xbox-one/accessories/kinect

13. Proteus VR. URL: https://www.proteus-vr.com

14. IKinema. URL: https://www.ikinema.com

15. Unreal Engine. URL: http://unrealengine.com

16. *Copenhaver J.* VR Animation and Locomotion Systems in Lone Echo. URL: https://readyatdawn.sharefile.com/share/view/s80d4725de7045259

17. *Steed A., Pan Y., Zisch F. & Steptoe W.* The impact of a self-avatar on cognitive load in immersive virtual reality // Proc. of IEEE Virtual Reality. 2016. P. 67.

СВЕДЕНИЯ ОБ АВТОРАХ

ГРИШКОВ Павел Дмитриевич – бакалавр Высшей школы информационных технологий и интеллектуальных систем (ВШ ИТИС) Казанского (Приволжского) федерального университета (КФУ). Сфера интересов: разработка приложений на Unreal Engine.

GRISHKOV Pavel Dmitrievich – Bachelor of Higher School ITIS. Sphere of interests: Unreal Engine development.

email: grishkovpavel@gmail.com

КУГУРАКОВА Влада Владимировна – старший преподаватель ВШ ИТИС КФУ, руководитель лаборатории «SIM». Сфера интересов: разработка игр, иммерсивность виртуальных сред, интерпретация биосигналов человека, погруженного в виртуальную среду, методы трансформации нарративной идентичности.

KUGURAKOVA Vlada Vladimirovna – Senior Lecturer of Higher School of Information Technology and Intelligent Systems, Head of Laboratory «SIM». Sphere of interests: game development, immersivity of virtual environment, narrative identity.

email: vlada.kugurakova@gmail.com

Материал поступил в редакцию 1 июля 2018 года.

УДК 004.42+004.75

МЕДИЦИНСКИЙ ЦИФРОВОЙ ПАСПОРТ, ОСНОВАННЫЙ НА ТЕХНОЛОГИИ РАСПРЕДЕЛЕННЫХ РЕЕСТРОВ

А. М. Плискин[1], А. Ф. Хасьянов[2]

Высшая школа информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета

alexnetmore@gmail.com[1], ak@it.kfu.ru[2]

Аннотация

Представлена реализация медицинского цифрового паспорта, использующая технологию распределенного реестра для хранения зашифрованных медицинских данных, цифровых сущностей пациентов и медицинских работников и доступов к данным. Описана система безопасного распределенного хранения высокочувствительных конфиденциальных медицинских данных.

***Ключевые слова**: медицинские записи, электронные данные о здоровье, блокчейн, распределенный реестр, пиринговая система, IPFS, Ehtereum, Bitcoin, смарт-контракт, цифровая сущность, шифрование с открытым ключом, симметричное шифрование*

ВВЕДЕНИЕ

Медицинские данные пациентов –- высокочувствительная конфиденциальная личная информация пациента, которая всегда должна быть доступна по его требованию. При этом пациент должен видеть любое их изменение, и только он может решать, кому предоставлять к ним доступ, а кому нет.

Стандартные решения в виде централизованных узлов и баз данных имеют следующие уязвимости:

- политику доступа пациентов к своим медицинским данным определяет владелец централизованных баз данных; по сути, пациент не владеет своими данными;
- в случае взлома одного узла злоумышленники получают доступ к данным многих пользователей;

- в случае выхода из строя узла медицинские данные всех пациентов теряются и не могут быть восстановлены, если не существует их резервной копии;
- злоумышленник, имея доступ к данным, может внести в них непоправимые и не всегда заметные изменения, в результате чего медицинские данные станут недействительными, и их дальнейшее использование может привести к назначению врачом неверного лечения пациенту, что может навредить его здоровью.

Технология блокчейн [1] предоставляет распределенную, неизменяемую и прозрачную историю всех транзакций, что в свою очередь позволяет решить проблему безопасного распределенного хранения электронных медицинских записей, созданных верифицированными медицинскими работниками.

Для безопасного хранения большого количества медицинских данных, где каждый пациент имеет прямой доступ к своим данным, а сами записи может внести только верифицированный медик, необходимо решение, основанное на технологии распределенных реестров или просто блокчейн и офф-чейн пиринговых систем. Эти технологии позволяют пользователям надежно и открыто хранить любую информацию.

Целью работы является создание системы медицинского цифрового паспорта – системы для хранения конфиденциальных медицинских данных в распределенном реестре на основе технологии блокчейн и пиринговых оф-чейн решений с открытым исходным кодом.

Для реализации решения были поставлены следующие задачи:

- Разработка логики смарт-контракта в блокчейн-сети Ethereum [3] для описания и хранения цифровых сущностей пользователей системы.
- Разработка логики смарт-контракта для хранения и предоставления доступа к чтению и изменению медицинских данных пациентов для медиков.
- Разработка пользовательского приложения для загрузки данных в распределенную файловую систему IPFS [4] и блокчейн-сеть Ethereum [2].
- Разработка функционала пользовательского приложения для шифрования и дешифрования данных, а также для просмотра этой информации пользователями системы.

ОБЗОР СУЩЕСТВУЮЩИХ РЕШЕНИЙ

Следующие решения для хранения медицинских историй используют технологию распределенного реестра:

GemHealth

Эта система построена на блокчейн-платформе Ethereum. Рассматриваются кейсы для хранения информации о рецептах, посещениях и анализа медицинских данных [5]. К недостаткам системы можно отнести следующее: записи хранятся в памяти смарт-контрактов; за это необходимо платить «gas», а размеры медицинских записей могут быть большими, поэтому придется платить много «gas». Также отсутствует информация о том, в каком виде хранятся записи, шифруются ли они. Единственным владельцем медицинских данных должен быть пациент, к которому они относятся, только он может решать, кому предоставлять к ним доступ. Если данные будут находиться в открытом доступе и не будут зашифрованы, то воспользоваться ими сможет любой без разрешения пациента, к которому они относятся.

GemHealth не является проектом с открытым исходным кодом. Техническое описание распределенной системы должно быть доступно любому ее пользователю, который может удостовериться в том, что его данные действительно надежно хранятся в зашифрованном виде и никто не может без его решения получить к ним доступ. Приложение должно иметь полностью открытый исходный код; оно должно работать автономно без внешнего контроля. Приложение может адаптировать свой протокол в ответ на предлагаемые улучшения и отзывы на рынке, но все изменения должны решаться на основе консенсуса его пользователей.

MedRec

Эта система построена на блокчейн-платформе Ethereum и представляет собой журнал медицинских записей пациентов, доступ к которому можно легко дать поставщикам медицинских услуг [6]. К достоинствам системы следует отнести то, что в памяти смарт-контрактов хранятся не сами записи, а указатели на эти файлы, таким образом, снижается стоимость «gas» за хранение этих записей. К недостаткам системы можно отнести следующее: записи хранятся на базе провайдера, который оказывает медицинские услуги. При изменении данных мож-

но будет узнать о том, что они были изменены, но восстановить их в случае потери оригинала уже будет нельзя. Также решение не является полностью распределенным, поскольку при отключении провайдера от сети данные уже нельзя будет получить. Также данное решение не является проектом с открытым исходным кодом.

Blockchain Health co

Информация об этом проекте есть только на официальном сайте [7], на котором сказано, что у каждого участника есть своя цифровая сущность. Пользователи могут делиться информацией с учеными, а те в свою очередь могут использовать ее для своих исследований. Система служит для создания доступного для исследований хранилища медицинских данных. К недостаткам системы можно отнести следующее: неизвестно, где и в каком виде хранятся сами данные, также система не является проектом с открытым исходным кодом.

DokChain

Эта система представляет собой приватную блокчейн-сеть для хранения медицинских записей [8]. Данные о медицинских записях хранятся в пиринговой файловой системе IPFS, что позволяет меньше загружать блокчейн-сеть. Недостатки системы заключаются в том, что в системе используется отдельная самописная блокчейн-сеть. Неизвестны ее возможности и характеристики. Нет информации о том, функционирует ли она на уровне продукта промышленного качества. Система не является проектом с открытым исходным кодом.

e-Health Records – e-Estonia

Это эстонская национальная система для хранения медицинских данных граждан страны, которая хранит данные о 97% граждан Эстонии [9]. Пациенты в любой момент времени могут получить доступ к своим медицинским записям и просмотреть список медиков, у которых есть доступ к данным. Данные, находящиеся в блокчейн-сети, используются для статистических исследований. Главным недостатком системы является то, что она доступна только для граждан Эстонии, так как только у них есть зарегистрированная в ней цифровая сущность. Также отсутствует техническое описания работы системы, и система не является проектом с открытым исходным кодом.

Blockchain For Health Data

Стоит сразу отметить, что описан лишь концепт системы для хранения электронных медицинских карточек пациентов, но реализация отсутствует. Сведения о медицинских данных хранятся в «озере данных» (англ. data lake), а не в блокчейне [10]. Планируется публикация исходных кодов проекта. К недостаткам стоит отнести то, что представлен лишь концепт системы. Также система не является полностью распределенной, так как для хранения статических файлов используется «озеро данных», которое хранится на одном узле или группе узлов, что представляет собой кластер, доступный исключительно его владельцу. Нет информации о том, какая блокчейн-сеть будет использована, поэтому неизвестно, насколько безопасно будут осуществлены хранение цифровых сущностей и доступ к данным внутри системы.

КОНЦЕПТУАЛЬНОЕ ОПИСАНИЕ И АРХИТЕКТУРА СИСТЕМЫ

Система медицинского цифрового паспорта на основе технологии распределенных реестров представляет собой комплекс, в котором данные о пациентах, медицинских работниках и доступ к медицинским записям хранятся в памяти смарт-контракта в блокчейн-сети Ethereum. Сами записи в зашифрованном виде хранятся в распределенной файловой системе IPFS.

Любой пациент может зарегистрироваться в системе и дать разрешение медицинскому работнику на просмотр его медицинских данных и внесение новых. Сущность медицинского работника должна быть подтверждена владельцем смарт-контракта. Медик может проверить, есть ли у него доступ к записям пациента. Если доступ есть, то он может просматривать данные пациента и добавлять новые, в противном случае он не имеет доступа к данным.

Каждая медицинская запись может измеряться произвольным количеством байт. В целях уменьшения стоимости, которую приходится платить за хранение одной записи, в памяти смарт-контракта медицинская запись, предварительно зашифрованная псевдослучайным ключом, сохраняется в распределенной файловой системе IPFS. В блокчейн сохраняется адрес этого файла, который представляет собой его хэш. К самому зашифрованному файлу доступ имеет любой участник сети IPFS, при этом он представляет собой произвольный набор байт, который не несет никакой полезной информации. Благодаря этому сохра-

няется полная конфиденциальность любой медицинской записи, хранящейся в IPFS.

Любая медицинская запись шифруется симметрично алгоритмом AES [14]. В процессе регистрации в системе пользовательское приложение пациента генерирует псевдослучайный ключ, который представляет собой конкатенацию адреса из 64 произвольных битов. Сам ключ, только уже ассиметрично зашифрованный публичным ключом пациента, лежит в профиле пациента в памяти смарт-контракта. Когда пациент дает доктору доступ к своим записям, ключ для шифрования записей достается пользовательской программой пациента, расшифровывается его приватным ключом, после шифруется публичным ключом медицинского работника и сохраняется в блокчейн, добавляя новое значение в список доступов.

Система состоит из трех компонентов (Рис. 1):

1. Пользовательский интерфейс, созданный при помощи javascript-фреймворка React-Redux, библиотеки для работы с блокчейн-сетью Ethereum – web3.js, библиотеки, предоставляющий API для работы с IPFS, и библиотеки bitcore-lib.js для шифрования с открытым ключом.
2. Блокчейн-составляющая – смарт-контракт, написанный на языке Solidity [11].
3. IPFS-демон, развернутый на устройстве пользователя для взаимодействия с сетью IPFS.

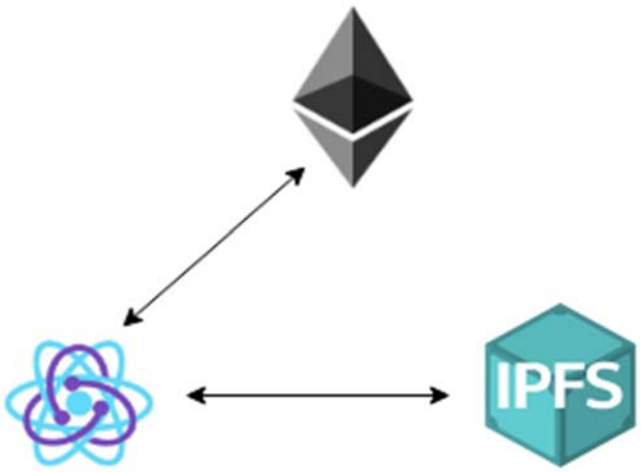


Рис. 1. Архитектура компонентов системы

СТРУКТУРА СМАРТ-КОНТРАКТА

Смарт-контракт написан на языке Solidity версии 0.4.20. Для выделения владельца смарт-контракта наследуется другой смарт-контракт Ownable, являющийся стандартом от Zeppelin [12] для наделения смарт-контракта правами собственности:

```
pragma solidity ^0.4.20;

import './Ownable.sol';

contract MedCard is Ownable {
```

Для привязки цифровой сущности медицинского работника и пациента к конкретному адресу введены словари, в которых каждому адресу соответствует структура, описывающая сущность медика и пациента:

```
mapping (address => Doctor) public doctors;

mapping (address => Patient) public patients;
```

Для составления медицинской истории пациента список адресов (хешей) медицинских записей привязан к конкретному адресу пациента:

```
mapping (address => string[]) private records;
```

Для проверки состояния пользователя в системе используются модификаторы доступа: является или нет пользователь медицинским работником; является или нет пользователь медицинским работником; является или нет пользователь пациентом:

```
modifier isDoctor(address _address) {
  ...
}

modifier isNotDoctor(address _address) {
    ...
}

modifier isPatient(address _address) {
    ...
}

modifier isNotPatient(address _address) {
    ...
}
```

В описании цифровой сущности медицинского работника хранятся хэш его профиля в IPFS, статус и публичный ключ. Статус медицинского работника может находиться в двух состояния (подтвержден и не подтвержден):

```
struct Doctor {
    string profile;
    bool accepted;
    string publicKey;
}
```

В описании цифровой сущности пациента хранятся хэш его зашифрованного профиля IPFS, зашифрованный ключ для работы с записями пациента, хэш файла с доступами для медиков и публичный ключ. В данном случае ключ шифруется публичным ключом пациента, поэтому прочитать его может только сам пациент, используя для дешифрования свой приватный ключ:

```
struct Patient {
    string profile;
    string passphrase;
    string permissions;
    string publicKey;
}
```

Для создания сущности пациента и медицинского работника добавлены соответствующие методы. Методы могут быть вызваны, если в системе пользователь не является ни пациентом, ни медиком:

```
function applyPatient(
    string _profile,
    string _passphrase,
    uint _permissions,
    string _publicKey
)
    public isNotPatient(msg.sender)
{
    patients[msg.sender] = Patient({
        profile: _profile,
        passphrase: _passphrase,
        permissions: _permissions,
```

```
            publicKey: _publicKey
        });
    }

    function applyDoctor(
        string _profile,
        string _publicKey
    )
        public isNotDoctor(msg.sender)
    {
        doctors[msg.sender] = Doctor({
            profile: _profile,
            accepted: false,
            publicKey: _publicKey
        });
    }
```

Для подтверждения статуса медицинского работника добавлен метод, который может вызвать только владелец смарт-контракта:

```
    function approveDoctor(
        address _address
    )
        public isNotDoctor(_address) onlyOwner
    {
        doctors[_address].accepted = true;
    }
```

Метод для добавления новой записи (добавляются не само содержание записи, а ее хэш из сети IPFS). Запись должна соответствовать sha3-хэшу ключа пациента для работы с записями, а добавить ее может только верифицированный доктор:

```
    function addRecord(
        string _hash,
        string _value
    )
        public isDoctor(msg.sender)
    {
        records[_hash].push(_value);
```

```
}
```

Для получения хэшей медицинских записей пациента добавлены два метода. Первый получает количество записей, второй достает хэш по индексу:

```
function getPatientRecordsLength(
    string _hash
)
    public constant returns (uint)
{
    return records[_hash].length;
}


function getPatientRecord(
    string _hash,
    uint _index)
public constant returns (string)
{
    return records[_hash][_index];
}
```

Пациент может дать доступ медикам, обновив список доступов:

```
function updatePermissions (
    string _permissions
)
    public isPatient(_msg.sender)
{
    patients[msg.sender].permissions = _permissions;
}
```

ПОЛЬЗОВАТЕЛЬСКОЕ ПРИЛОЖЕНИЕ

Пользователи системы напрямую не работают с блокчейн-сетью Ethereum. Вместо этого создано пользовательское приложение, которое дает возможность пользоваться всей бизнес-логикой смарт-контракта, шифровать и дешифровать данные о ключах и самих медицинских записях и сохранять их в IPFS.

Сначала пользователю необходимо предоставить системе доступ к своей цифровой сущности, которую определяет приватный ключ. При входе в систему пользователю предлагается выбрать файл, в котором находится приватный ключ. Сам файл должен быть формата json и выглядеть следующим образом:

```
{
    "pkey": "0x9df1b091b86983ebbfaee3613bfabf26aa873d82485c3ece4e95bc39713d8c45"
}
```

По приватному ключу система определяет пользователя и интерфейс для работы с блокчейн-сетью Ethereum.

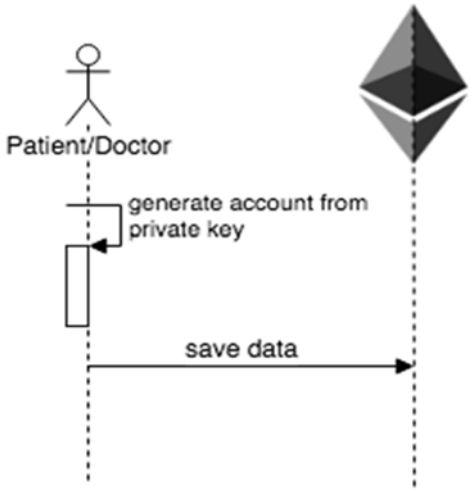


Рис. 2. Схема регистрации в системе

Если пользователь не зарегистрирован в системе, то ему предлагаются две формы для регистрации: как медицинского работника или пациента. После заполнения формы транзакция с данными отправляется в блокчейн-сеть Ethereum и сохраняется в памяти смарт-контракта (Рис. 2). Если регистрируется пациент, то для него генерируется псевдослучайный ключ, который шифруется его приватным ключом. Зашифрованный ключ также отправляется вместе с данными транзакции. После подтверждения транзакции пользователь может войти в систему, предоставив ей файл с приватным ключом. Интерфейс будет зависеть от роли пользователя.

Пациент сможет по адресу найти любого медицинского работника, дать ему доступ к своим данным. Также пациент сможет просмотреть свои медицинские записи.

Если пациент хочет дать доступ медику, то у него локально дешифруется его псевдо-рандомный ключ, который лежит в памяти смарт-контракта. Полученное значение шифруется публичным ключом медицинского работника и отправляется в память смарт-контракта (Рис. 3). В итоге у медика появляется доступ на чтение данных пользователя.

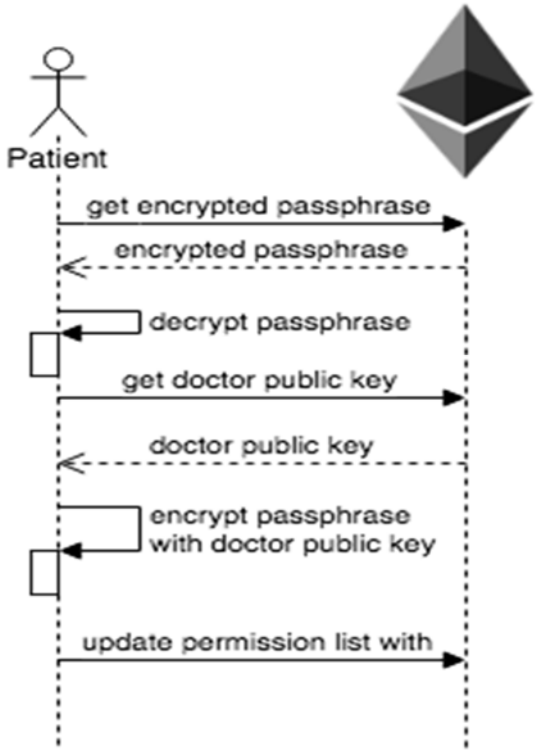


Рис. 3. Схема предоставления доступа к записям

Медицинский работник может по адресу найти любого пациента. Если у медика есть доступ, то он сможет просмотреть все записи пациента (Рис. 4) и добавить новые.

При добавлении новой записи происходит следующее: медик прикрепляет файл с новой медицинской записью; она шифруется ключом, полученным дешифрированием значения ключа конкретного пациента (ключ дешифруется приватным ключом медика) (Рис. 5). После полученный набор байт кладется в пиринговую сеть IPFS, возвращается хэш этой записи и сам хэш уже посылается транзакцией в блокчейн-сеть Ethereum.

На данном этапе разработки приложения владелец смарт-контракта может подтвердить любого медика, используя среду для написания, развёртывания и тестирования смарт-контрактов Remix.

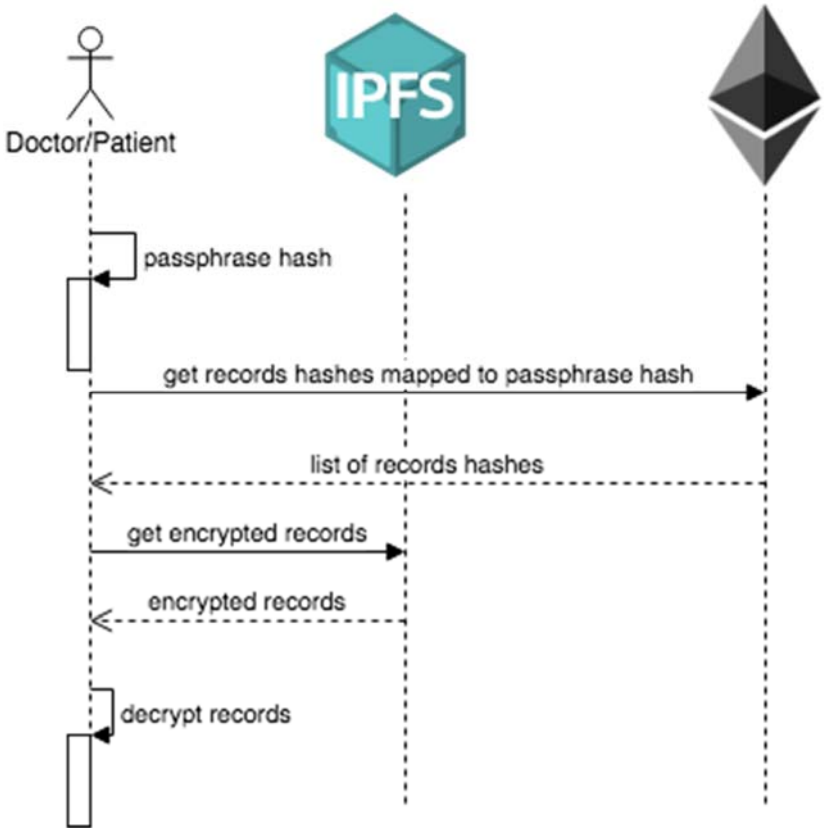


Рис. 4. Схема получения записей

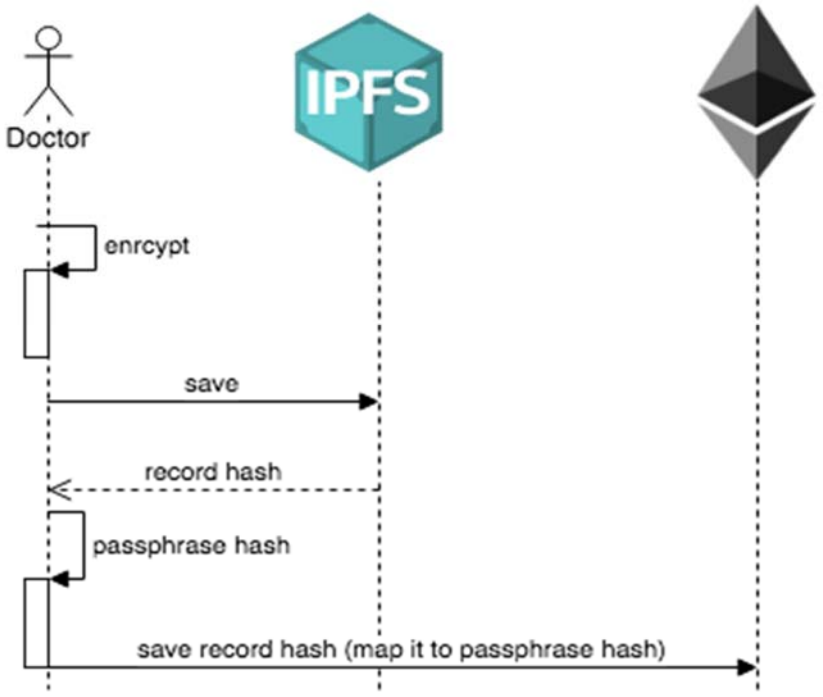


Рис. 5. Схема добавления новых записей

ЗАКЛЮЧЕНИЕ

Спроектирована, разработана и описана система для распределенного безопасного хранения медицинского паспорта пациента и обеспечения безопасного доступа к высокочувствительным медицинским записям с открытым исходным кодом, который доступен в репозитории на гитхабе (англ. github) [13]. Разработанная система имеет следующие достоинства:

- относительно невысока стоимость хранения данных в памяти смарт-контракта за счет хранения контента медицинских записей в бесплатной пиринговой сети IPFS;
- использовано асимметричное шифрование медицинских записей, которое не позволяет злоумышленникам получить доступ к конфиденциальным данным; сложность подбора закрытого ключа несоизмеримо высока и требует полного перебора всех возможных значений;

Дальнейшая работа предполагает:

- исследовать методы для верификации цифровой сущности медицинского работника; рассмотреть проекты, созданные на основе технологии

распределенных реестров, на основе которых можно получить информацию о компетенциях, лицензиях и опыте медиков;

- внести функционал в пользовательское приложение для поиска пациента и медицинского работника по QR-коду;
- добавить возможность для пациента давать доступ только к части своих данных;
- внедрить систему в инфраструктуру медицинской клиники Казанского федерального университета.

СПИСОК ЛИТЕРАТУРЫ

1. *Nakamoto S.* Bitcoin: A peer-to-peer electronic cash system. 2008.
2. *Wood G.* Ethereum: A Secure Decentralised Generalised Transaction Ledger //Ethereum Project Yellow Paper. 2014. V. 151. P. 1–32.
3. *Buterin V.* Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform. URL: https://github.com/ethereum/wiki/wiki/% 5BEnglish% 5D-White-Paper. 2014
4. *Benet J.* Ipfs-content Addressed, Versioned, p2p file system //arXiv preprint arXiv:1407.3561. 2014.
5. *Prisco G.* The Blockchain for Healthcare: Gem Launches Gem Health Network With Philips Blockchain Lab. URL: https://bitcoinmagazine.com/articles/the-blockchain-for-heathcare-gem-launches-gem-health-network-with-philips-blockchain-lab-1461674938
6. *Ekblaw A. et al.* A Case Study for Blockchain in Healthcare: “MedRec” Prototype for Electronic Health Records and Medical Research Data //Proceedings of IEEE Open & Big Data Conference. 2016. V. 13. P. 13.
7. Blockchain Health. URL: https://blockchainhealth.co/
8. Dockchain. URL: https://dokchain.com/download/whitepaper/
9. E-estonia healthcare. URL: https://e-estonia.com/solutions/healthcare/
10. *Linn L. A., Koo M. B.* Blockchain for Health Data and its Potential Use in Health It and Health Care Related Research //ONC/NIST Use of Blockchain for Healthcare and Research Workshop, Gaithersburg, Maryland, United States: ONC/NIST. 2016.
11. *Dannen C.* Introducing Ethereum and Solidity. Apress, 2017.

12. Zeppelin Solidity. URL: http://zeppelin-solidity.readthedocs. io/en/latest/ownable.html

13. *Pliskin Alexander*. MedicalRecords. URL: https://github.com/GoodBoy962/ MedicalCards

14. *Баричев С. Г., Гончаров В. В., Серов Р. Е.* 2.4. 2. Стандарт AES. Алгоритм Rijdael. Основы современной криптографии. М.: Горячая линия–Телеком, 2002. С. 30–35.

MEDICAL DIGITAL PASSPORT BASED ON DISTRIBUTED LEDGER

A. M. Pliskin[1], **A. F. Khasyanov**[2]

Higher School of Information Technologies and Intellectual Systems at Kazan (Volga Region) Federal University

alexnetmore@gmail.com[1], ak@it.kfu.ru[2]

Abstract

The paper presents the implementation of the patient medical digital passport, using distributed ledger for storing encrypted electronic health records, patient and medic digital entities and accesses for data. A system for the secure distributed storage of highly sensitive confidential medical data is described.

Keywords*: medical data, electronic health records, blockchain, distributed ledger, peer-to-peer system, IPFS, Ehtereum, Bitcoin, smart contract, digital identity, public key encryption, symmetric encryption*

REFERENCES

1. *Nakamoto S.* Bitcoin: A peer-to-peer electronic cash system. 2008.
2. Wood G. Ethereum: A Secure Decentralised Generalised Transaction Ledger //Ethereum Project Yellow Paper. 2014. V. 151. P. 1–32.
3. *Buterin V.* Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform. URL: https://github.com/ethereum/wiki/wiki/% 5BEnglish% 5D-White-Paper. 2014
4. *Benet J.* Ipfs-content Addressed, Versioned, p2p file system //arXiv preprint arXiv:1407.3561. 2014.
5. *Prisco G.* The Blockchain for Healthcare: Gem Launches Gem Health Network With Philips Blockchain Lab. URL: https://bitcoinmagazine.com/articles/the-blockchain-for-heathcare-gem-launches-gem-health-network-with-philips-blockchain-lab-1461674938
6. *Ekblaw A. et al.* A Case Study for Blockchain in Healthcare: “MedRec” Prototype for Electronic Health Records and Medical Research Data //Proceedings of IEEE Open & Big Data Conference. 2016. V. 13. P. 13.
7. Blockchain Health. URL: https://blockchainhealth.co/
8. Dockchain. URL: https://dokchain.com/download/whitepaper/

9. E-estonia healthcare. URL: https://e-estonia.com/solutions/healthcare/

10. *Linn L. A., Koo M. B.* Blockchain for Health Data and its Potential Use in Health It and Health Care Related Research //ONC/NIST Use of Blockchain for Healthcare and Research Workshop, Gaithersburg, Maryland, United States: ONC/NIST. 2016.

11. *Dannen C.* Introducing Ethereum and Solidity. Apress, 2017.

12. Zeppelin Solidity. URL: http://zeppelin-solidity.readthedocs.io/en/latest/ownable.html

13. *Pliskin Alexander*. MedicalRecords. URL: https://github.com/GoodBoy962/ MedicalCards

14. *Barichev S. G., Goncharov V. V., Serov R.E.* Standart AES. Algoritm Rijdael. Osnovi sovremennoi cryptografii. M.: Goryachay liniya–Telekom, 2002. S. 30–35.

СВЕДЕНИЯ ОБ АВТОРАХ

ПЛИСКИН Александр Маркович – студент Высшей школы информационных технологий и интеллектуальных систем Казанского федерального университета, разработчик программного обеспечения.

Alexander Markovich PLISKIN – student of the Higher Institute of Information Technologies and Intelligent Systems of Kazan Federal University, software engineer.

email: alexnetmore@gmail.com

ХАСЬЯНОВ Айрат Фаридович – директор Высшей школы информационных технологий и интеллектуальных систем Казанского федерального университета. Имеет степень PhD Боннского университета в области естественных наук. Сфера научных интересов лежит в области применения информационных и интеллектуальных технологий в области образования, различных отраслях информационных технологий и информатики.

Dr. Ayrat KHASYANOV obtained his PhD in Computer Science in 2005 from the University of Bonn in Germany. He serves his duty as the head of the Higher Institute for Information Technologies and Intelligent Systems at Kazan Federal University. His interests lay in the field of application of intelligent and information technology to the fields of teaching and learning, as well as various fields of information technology and computer science.

email: ak@it.kfu.ru

Материал поступил в редакцию 3 июня 2018 года

УДК 004.414.3

ЭФФЕКТИВНАЯ РАЗРАБОТКА ПРИЛОЖЕНИЙ ПРИ МИКРОСЕРВИСНОЙ АРХИТЕКТУРЕ

А. Э. Порфильева[1], Р. Ф. Шайхутдинов[2], Г. А. Нуриева[3], М. Р. Сидиков[4], М. М. Абрамский[5], А. И. Карпов[6], Д. И. Раимов[7], Р. Р. Новиков[8]

[1-8] *Высшая школа информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета*

[1]porfileva.anastasia@gmail.com, [2]rus.shaikhut@gmail.com, [3]nurievag97@gmail.com, [4]sidikov.marsel@gmail.com, [5]ma@it.kfu.ru, [6]artik100313@gmail.com, [7]dinar88raimov@gmail.com, [8]ruslandia996@gmail.com

Аннотация

Рассмотрены особенности внедрения микросервисной архитектуры в процесс разработки. Проиллюстрированы преимущества данного подхода по сравнению с традиционным монолитным подходом. Показана связь использования микросервисной архитектуры с возможностью работы команды по гибким методологиям разработки.

***Ключевые слова**: микросервисы, микросервисная архитектура, эффективная разработка, гибкие методологии*

ВВЕДЕНИЕ

Моделирование архитектуры программного продукта является важной частью процесса разработки. Выбор конкретного архитектурного решения зависит от многих факторов, в частности, назначения разрабатываемой системы. Несмотря на разнообразие подходов к архитектуре приложений, в рамках современных проектов, нуждающихся в легкой масштабируемости, в последнее время предпочтение отдается микросервисам [1].

Термин Microservice Architecture описывает технологию разработки программного обеспечения, при которой приложение представляет собой совокупность слабосвязанных сервисов. Сервисы строятся вокруг бизнес-логики и имеют возможность независимого развертывания. Каждый сервис работает в своем

процессе и взаимодействует с другими сервисами при помощи легковесных механизмов, таких, например, как HTTP [2]. При этом разработка каждого сервиса может вестись независимо от других. Это позволяет организовать процесс разработки гибко и распределенно.

Основная цель данной работы заключается в определении и описании условий, необходимых для осуществления эффективной разработки на основе микросервисной архитектуры.

Статья организована следующим образом: в первом разделе описан подход сервис-ориентированной архитектуры (SOA), во втором разделе дано описание преимуществ микросервисной архитектуры по сравнению с монолитными архитектурами, в третьем разделе описаны предлагаемые принципы организации эффективной разработки в проектной команде, разрабатывающей приложения в микросервисной архитектуре.

1. СЕРВИС-ОРИЕНТИРОВАННАЯ АРХИТЕКТУРА

Сервис-ориентированная архитектура (Service-oriented architecture, SOA) [3] – это подход к разработке приложений, который осуществляет публикацию сервисами своих интерфейсов, позволяя другим сервисам обращаться к ним посредством стандартных интернет-протоколов, которые не зависят от языков программирования и платформ.

Далее будет проведен сравнительный анализ нескольких архитектурных патернов, реализующих принципы SOA.

Общая архитектура брокера объектных запросов (CORBA)

CORBA [4] – это архитектура, предназначенная для интеграции различных систем с помощью вызова удаленных процедур, что позволяет не зависеть от операционных систем, языков программирования и оборудования. Для взаимодействия используется абстрактный протокол GIOP, а интерфейсы описаны на языке IDL [5].

Отметим, что стандарт CORBA имел ряд недостатков, среди которых:

- независимость от местоположения – клиентский код не обладает информацией, локальным или удаленным был вызов;
- использование нестандартных протоколов и портов; корпоративные политики безопасности часто не допускают их использования.

Были необходимы надежный канал связи и простой обмен сообщениями, а также было необходимо уменьшить количество удаленных вызовов для повышения производительности системы. В 1990 г. появились веб-сервисы [6] – это приложение с определенным URL, которое предоставляет свой интерфейс для взаимодействия с помощью сообщений:

- приложения взаимодействуют посредством сообщений, поэтому сократилось число удаленных вызовов;
- сообщения отправляются по HTTP-соединению;
- используются простые спецификации для обмена сообщениями (SOAP, REST, GraphQL);
- используются форматы сообщений, которые не зависят от платформ (XML, JSON и т. д).

Однако и веб-сервисы обладали рядом недостатков:

- возможна перегрузка системы из-за синхронного обмена сообщениями;
- по причине различий форматов сообщений трудно интегрировать одни службы с другими.

Через некоторое время появилась **очередь сообщений** – шаблон проектирования, обеспечивающий независимость компонентов системы, которые используются для обмена информацией с заранее определенным форматом сообщений. Очереди сообщений обеспечивают масштабируемость, отказоустойчивость и гарантированную доставку.

В роли брокеров сообщений используются такие инструменты, как RabbitMQ [7], Kafka [8], Beanstalkd [9], Amazon MQ [10] и т. д.

Одним из недостатков этого шаблона является то, что очереди сообщений создают дополнительную операционную сложность. Каждая очередь должна быть создана, настроена и должна контролироваться. Каждому отправителю и получателю необходимо настроить местоположение и имя каждой очереди.

Данный подход решил проблему синхронного обмена сообщениями, но проблема трудной интеграции различных сервисов осталась. Еще одной попыткой решить проблему стал шаблон проектирования «Сервисная шина предприятия» (ESB), который работает как очередь сообщений, но конвертирует сообщения в

нужный формат, тем самым решая проблему интеграции сервисов, использующих разные форматы сообщений [11]. Недостатки данного шаблона:

- снижается скорость связи из-за конвертации сообщений;
- общая точка отказа;
- высокая сложность конфигурирования.

2. МИКРОСЕРВИСНАЯ АРХИТЕКТУРА

Микросервисная архитектура [2] является современным представлением подхода SOA. В микросервисной архитектуре сервисы имеют меньший размер и обмениваются сообщениями внутри приложения. Программный интерфейс сервиса публикуется через шлюз, который транслирует запросы извне в нужные сервисы. Таким образом, микросервисы могут быть разработаны в широком технологическом стеке.

Сервисы возможно реализовать на различных языках программирования с применение разных баз данных, в зависимости от того, что лучше подходит для решения конкретных задач [12].

Микросервисы обычно противопоставляются монолитному подходу – когда приложение построено как единое целое с общей базой данных. Автор книг и статей по архитектуре программного обеспечения Мартин Фаулер определяет несколько преимуществ монолитной и микросервисной архитектур [2]:

Преимущества монолитной архитектуры:

- *простота;*
- *согласованность;*
- *межмодульный рефакторинг* – при необходимости можно переместить классы из одного модуля в другой.

Преимущества микросервисной архитектуры:

- *частичное развертывание* – возможность обновлять отдельные сервисы приложения независимо от других;
- *доступность* – если какие-то из модулей прекращают свое функционирование, то работа других модулей и приложения в целом не прерывается;
- *сохранение модульности* – исключается наличие общих состояний между несколькими модулями, что дает возможность отключать/изымать одни модули, не нарушая работу других;

- *гетерогенность* – возможность реализации модулей системы на разных языках; гетерогенные информационные системы помогают избегать наличия тесных связей между модулями системы благодаря использованию разных языков программирования;
- *независимая масштабируемость* – способность модулей справляться с увеличением рабочей нагрузки при добавлении ресурсов независимо от других модулей, размещаемых на других серверных узлах [13].

Следует учитывать, что многие приложения начинают разрабатываться с помощью монолитного подхода, в силу своих преимуществ зарекомендовавшего себя для небольших команд. Однако со временем монолитные приложения начинают обладать массивной кодовой базой, что означает необходимость развернуть весь стек технологий и собрать целиком проект при разработке сравнительно небольших новых задач.

3. ОРГАНИЗАЦИЯ ГИБКОЙ РАЗРАБОТКИ

В своем блоге «Coding the Architecture» в 2013 году архитектор программного обеспечения Саймон Браун предположил, как будет выглядеть гибкая архитектура программного обеспечения [14]. Его описание гибкой архитектуры достаточно хорошо ложится на микросервисную архитектуру. По его мнению, предоставить гибкость поможет стиль архитектуры, построенный с использованием небольших слабосвязанных компонентов (сервисов), которые взаимодействуют друг с другом для ответа на запрос пользователя. Сервисы могут быть модифицированы и протестированы изолированно или даже вырваны и заменены в зависимости от изменения требований, новые компоненты могут быть добавлены и масштабированы, если необходимо.

Согласно манифесту гибкой разработки (agile manifesto, [15]), успешные и эффективные команды принимают необходимые решения совместно, а рекомендованный размер команды составляет не менее 3 и не более 9 человек [16]. Таким образом, Agile обеспечивает поддержку эффективной совместной разработки в архитектуре микросервисов.

Ведущие технологические организации, такие, как Amazon, Netflix и Apple, используют микросервисную архитектуру для обеспечения гибкости быстрой адаптации и реагирования на запросы своих клиентов [17]. Согласно ресурсам

NGINX, 68% организаций проводят исследования или внедряют микросервисы [18]. Такой рост значимости микросервисов отражает переход от крупных приложений с кодовым содержанием (монолитов) к более легким приложениям.

Однако, несмотря на то что ведущие компании уже делают переход от монолитной архитектуры к микросервисной, существует ряд сложностей, с которыми можно столкнуться при разработке и поддержке и которые надо учитывать при переходе на микросервисную архитектуру:

1. Необходимо предусматривать автономность каждого микросервиса от других компонентов. Если каждый сервис вызывает несколько удаленных сервисов, стоит продумать способы по уменьшению общего времени задержки. Помимо этого, следует учитывать проблемы с надежностью, так как удаленный вызов может отказать в любой момент, таким образом, с увеличением количества сервисов потенциальных точек отказа становится больше.
2. Требуется хранить единые компоненты системы в единой библиотеке, однако в этом случае при изменении данной библиотеки необходимо перезапускать все связанные микросервисы. Другой вариант — хранить единый код во всех микросервисах, однако такой код сложнее поддерживать.
3. Стратегия тестирования, которая применяется к монолитным приложениям, должна меняться при переходе на микросервисы. Учитывая, что приложения, построенные в архитектуре микросервисов, обеспечивают высокую функциональность и производительность, тестирование должно охватывать каждый уровень сервисов и их взаимодействие. Следовательно, увеличивается время, необходимое для тестирования всего приложения.
4. Для внедрения больших систем микросервисов требуются затраты на квалифицированных специалистов, хорошая инфраструктура и ресурсы, необходимые для организации взаимодействия сервисов.

Можно выделить следующие случаи, когда микросервисная архитектура будет поддерживать модель эффективной разработки.

1. *Микрокоманды.* В большом проекте с небольшими командами разработчиков, каждая из которых реализует собственный функционал, изменения одной группы так или иначе затрагивают другие группы — такой подход в команде замедляет процесс разработки. Наличие разных групп разработ-

чиков с общей кодовой базой также приводит к проблемам – по мере увеличения объема кода становится сложно следить за тем, чтобы все части были организованными и оптимально сочетались друг с другом. С архитектурой микросервисов возможно разделить код так, чтобы различные команды разработчиков могли полностью владеть им. Это позволит командам внедрять новшества намного быстрее, без утомительных процессов проектирования, просмотра и развертывания.

2. *Опытный менеджер проекта.* При использовании микросервисов роль руководителя проекта выходит на первый план, так как над различными частями приложения работает множество разных команд. Чтобы синхронизировать работу микрокоманд и избежать проблем, следует нанимать грамотных проектных менеджеров.
3. *Длительный период жизни и поддержки проекта.* Если проект стремительно развивается в своих масштабах и годы разработки только будут его увеличивать, то есть смысл разделять кодовую базу на отдельные компоненты. Внедрение микросервисов помогает при длительной разработке. Если проект необходимо выполнить в кратчайшие сроки, то внедрение новых технологий будет лишь замедлять процесс разработки.
4. *Знания.* Не следует использовать микросервисную архитектуру в том случае, если в команде отсутствуют специалисты с пониманием технологий микросервисов.

Рассмотрим, какие технические знания потребуются java-разработчику для работы с микросервисами. Микросервисы – это распределенная система, каждый сервис должен решать только одну проблему, чтобы не перекомплементировать систему. Проектная группа должна быть готова к реорганизации программных элементов по мере необходимости. Знание контейнеров, таких, как Docker, может быть полезным. Поскольку микросервисы Java обычно связывают REST через HTTP, требуются также знания REST / HTTP / RAML / Swagger. Упростить работу с микросервисами помогут такие фреймворки, как Spring, Spark, Dropwizard, Play Framework и так далее [19].

ЗАКЛЮЧЕНИЕ

При разработке программного продукта необходимо серьезно подойти к выбору архитектурного решения. Микросервисы представляются подходящим выбором для реализации больших систем, предполагающих длительную разработку. Однако микросервисная архитектура несет в себе достаточно много рисков. В этом случае успех проекта во многом зависит от наличия опытного системного архитектора и проектного менеджера.

Команда, использующая микросервисную архитектуру, должна быть сформирована на основе бизнес-возможностей. Одним из обязательных условий успешного внедрения в проект микросервисной архитектуры является компетентность команды.

СПИСОК ЛИТЕРАТУРЫ

1. *Chris Richardson* (2017). Pattern: Microservice Architecture. URL: http://microservices.io/patterns/microservices.html

2. *James Lewis, Martin Fowler* (25 марта 2014). Microservices. URL: https://martinfowler.com/articles/microservices.html

3. Microsoft – Understanding Service-Oriented Architecture. URL: https://msdn.microsoft.com/en-us/library/aa480021.aspx

4. Сайт Corba. URL: http://www.corba.org/

5. *Steve Vinoski.* CORBA: Integrating Diverse Applications Within Distributed Heterogenerous Environments / Steve Vinoski, 1997.

6. *Алекс Родригес* (16.09.2015). Web-сервисы RESTful: основы. URL: https://www.ibm.com/developerworks/ru/library/ws-restfu/index.html

7. Сайт RabbitMQ. URL: https://www.rabbitmq.com/

8. Сайт Apache Kafka. URL: https://kafka.apache.org/intro

9. Сайт Beanstalkd. URL: http://kr.github.io/beanstalkd/

10. Сайт AmazonMQ. URL: https://aws.amazon.com/ru/amazon-mq/

11. Microsoft Message Bus (2004). URL: https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ff647328(v=pandp.10)

12. *Sam Newman*. Principles of Microservices, 2015. URL: https://vimeo.com/131632250

13. *Michael Hofmann, Erin Schnabel and Katherine Stanley*. Microservices Best Practices for Java / URL: http://www.redbooks.ibm.com/abstracts/sg248357.html

14. *Simon Brown*. Coding the Architecture. URL: http://www. codingthear-chitecture.com/2013/09/03/what_is_agile_software_architecture.html

15. Agile Mnifesto. URL: http://agilemanifesto.org/

16. The Scrum Guide ("The Development Team" chapter) URL: https://www. scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf

17. Leanix. Why Netflix, Amazon, and Apple Care About Microservices. URL: https://blog.leanix.net/en/why-netflix-amazon-and-apple-care-about-microservices

18. NGINX. The Future of Application Development and Delivery Is Now. URL: https://www.nginx.com/resources/library/app-dev-survey/

19. Leanix. Developing Microservices with Java. URL: https://blog.leanix.net/ en/developing-microservices-with-java

EFFECTIVE APPLICATION DEVELOPMENT WITHIN MICROSERVICE ARCHITECTURE

A. E. Porfileva[1], R. F. Shaikhutdinov[2], G. A. Nurieva[3], M. R. Sidikov[4], M. M. Abramskiy[5], A. I. Karpov[6], D. I. Raimov[7], R. R. Novikov[8]

[1-8] *Higher School of Information Technologies and Intelligent Systems, Kazan (Volga Region) Federal University*

[1]porfileva.anastasia@gmail.com, [2]rus.shaikhut@gmail.com, [3]nurievag97@gmail.com, [4]sidikov.marsel@gmail.com, [5]ma@it.kfu.ru, [6]artik100313@gmail.com, [7]dinar88raimov@gmail.com, [8]ruslandia996@gmail.com

Abstract

The paper presents the features of the using the microservice architecture in the development process. The advantages of this approach are illustrated in comparison with the traditional monolithic approach. The connection between the use of the microservice architecture and the ability of the team to work within agile development methodologies is shown.

***Keywords**: microservices, microservice architecture, efficient development, agile methodologies*

REFERENCES

1. *Chris Richardson* (2017). Pattern: Microservice Architecture. URL: http://microservices.io/patterns/microservices.html

2. *James Lewis, Martin Fowler* (25 March 2014). Microservices. URL: https://martinfowler.com/articles/microservices.html

3. Microsoft - Understanding Service-Oriented Architecture. URL: https://msdn. microsoft.com/en-us/library/aa480021.aspx

4. Corba Website. URL: http://www.corba.org/

5. *Steve Vinoski*. CORBA: Integrating Diverse Applications Within Distributed Heterogenerous Environments / Steve Vinoski, 1997

6. *Aleks Rodriges* (16.09.2015). Web services RESTful: basics. URL: https:// www.ibm.com/developerworks/ru/library/ws-restfu/index.html

7. RabbitMQ Website. URL: https://www.rabbitmq.com/

8. Apache Kafka Website. URL: https://kafka.apache.org/intro

9. Beanstalkd Website. URL: http://kr.github.io/beanstalkd/

10. AmazonMQ Website. URL: https://aws.amazon.com/ru/amazon-mq/

11. Microsoft Message Bus (2004). URL: https://docs.microsoft.com/en-us/ previous-versions/msp-n-p/ff647328(v=pandp.10)

12. *Sam Newman*. Principles of Microservices, 2015. URL: https://vimeo.com/ 131632250

13. *Michael Hofmann, Erin Schnabel and Katherine Stanley*. Microservices Best Practices for Java. URL: http://www.redbooks.ibm.com/abstracts/sg248357.html

14. *Simon Brown*. Coding the Architecture blog. URL: http://www. codingthearchitecture.com/2013/09/03/what_is_agile_software_architecture.html

15. Agile Manifesto URL: http://agilemanifesto.org/

16. The Scrum Guide (“The Development Team” chapter) URL: https://www. scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf

17. Leanix, Why Netflix, Amazon, and Apple Care About Microservices. URL: https://blog.leanix.net/en/why-netflix-amazon-and-apple-care-about-microservices

18. NGINX, The Future of Application Development and Delivery Is Now. URL: https://www.nginx.com/resources/library/app-dev-survey/

19. Leanix, Developing Microservices with Java. URL: https://blog.leanix.net/en/ developing-microservices-with-java

СВЕДЕНИЯ ОБ АВТОРАХ

ПОРФИЛЬЕВА Анастасия Эдуардовна – студентка Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета.

Anastasia Eduardovna PORFILEVA – *student of Higher School of ITIS KFU.*

email: porfileva.anastasia@gmail.com

ШАЙХУТДИНОВ Рустем Фаритович – студент Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета.

Rustem Faritovich SHAIKHUTDINOV – student of Higher School of ITIS KFU.

email: rus.shaikhut@gmail.com

НУРИЕВА Гульшат Атласовна – студентка Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета.

Gulshat Atlasovna NURIEVA – student of Higher School of ITIS KFU

email: nurievag97@gmail.com

СИДИКОВ Марсель Рафаэлевич – руководитель лаборатории Java Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета.

Marsel Rafaelevich SIDIKOV – head of Java Lab of ITIS KFU.

email: sidikov.marsel@gmail.com

АБРАМСКИЙ Михаил Михайлович – старший преподаватель кафедры программной инженерии Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета.

Mikhail Mikhailovich ABRAMSKIY – senior lecturer at Department of Software Engineering of Higher School of ITIS KFU

email: ma@it.kfu.ru

КАРПОВ Артур Иванович – студент Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета.

Artur Ivanovich KARPOV – student of Higher School of ITIS KFU

email: artik100313@gmail.com

РАИМОВ Динар Ильдусович – студент Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета.

Dinar Il'dusovich RAIMOV – student of Higher School of ITIS KFU

email: dinar88raimov@gmail.com

НОВИКОВ Руслан Радикович – студент Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета.

NOVIKOV Ruslan Radikovich – student of Higher School of ITIS KFU

email: ruslandia996@gmail.com

Материал поступил в редакцию 28 июля 2018 года

УДК 004.514+004.58

МЕТОДЫ МОДИФИКАЦИИ ВИЗУАЛЬНЫХ ИНТЕРФЕЙСОВ ANDROID-ПРИЛОЖЕНИЙ НА ОСНОВЕ ИНДИВИДУАЛЬНЫХ ПОЛЬЗОВАТЕЛЬСКИХ ХАРАКТЕРИСТИК

А. М. Сарматин[1], И. С. Шахова[2]

Высшая школа информационных технологий и информационных систем Казанского (Приволжского) федерального университета

[1]antonsarmatin@googlemail.com, [2]is@it.kfu.ru

Аннотация

Проанализированы факторы, влияющие на модификацию визуальных интерфейсов. Предложены правила модификации рассмотренных факторов на основе индивидуальных пользовательских характеристик. Разработаны методы модификации визуальных интерфейсов Android-приложений.

***Ключевые слова**: android, UI, user interface, пользовательский интерфейс, визуальный интерфейс, мобильные приложения*

ВВЕДЕНИЕ

В области разработки визуальных интерфейсов мобильных приложений существует множество стандартов и правил проектирования, в которых интерфейсы подстраиваются под усредненного пользователя мобильного приложения. Данные стандарты и правила основаны на многих факторах, включающих в себя физиологические аспекты взаимодействия пользователя с устройством, визуальное восприятие и простоту понимания структуры интерфейса приложения. Несмотря на то, что многие из этих правил построения мобильных интерфейсов являются эталонными, существует проблема применимости правил для пользователей, отличающихся от усредненного портрета, на который эти правила ориентированы.

В настоящее время разработка удобного мобильного интерфейса, учитывающего широкий спектр особенностей взаимодействия пользователя, является крайне сложной задачей. Большинство фундаментальных исследований трудно

применить к современным требованиям, за последние годы изменились и форма устройств, и принцип взаимодействия с ними. В данный момент взаимодействие с пользователем положено на визуальную составляющую – экран и графический интерфейс, все меньше производителей используют физические органы взаимодействия с устройством. Использование экрана как интерфейса взаимодействия с устройством позволяет адаптировать такой интерфейс под определенные факторы взаимодействия.

Целью работы является разработка методов модификации визуальных интерфейсов Android-приложений на основе индивидуальных пользовательских характеристик, выделяемых при помощи факторов влияющих на восприятие интерфейсов.

ИССЛЕДОВАНИЕ ФАКТОРОВ

Комплексная задача исследования факторов, влияющих на модификацию визуальных интерфейсов, состоит их нескольких подзадач:

- исследование влияния внешних независимых факторов;
- исследование влияния внешних зависимых факторов;
- исследование влияния внутренних факторов.

Для формулировки правил модификации визуальных интерфейсов требуется определить факторы, влияющие на взаимодействие с ними, основываясь на их типе и зависимости от других факторов или условий взаимодействия. Для каждой из подзадач требуется провести исследования и выявить правила поведения пользователей и устройств в зависимости от влияния определенных факторов.

Исследование факторов, влияющих на взаимодействие с визуальным интерфейсом приложения, и формулировка правил их модификации позволят разработать методы модификации визуальных интерфейсов мобильных приложений, основываясь на совокупности данных, полученных при исследовании независимых пользователей и их поведенческих характеристик при использовании приложения.

При разработке графического интерфейса мобильного приложения следует придерживаться правил проектирования, основанных на факторах восприятия интерфейса конечным пользователем. Влияющие на восприятие и взаимодей-

ствие факторы можно разделить на обобщенные группы, упростив задачу их исследования. Рассмотрим внешние и внутренние факторы.

Внешние факторы – это такие факторы, которые не могут быть однозначно предсказаны при разработке интерфейса. Однако внешние факторы оказывают самое сильное влияние на взаимодействие с визуальным интерфейсом приложения. Их также можно разделить на несколько подгрупп: зависимые, независимые, полузависимые (как факторы, относящиеся к первым двум подгруппам).

Внутренние факторы – это такие факторы, которые могут быть предсказаны при разработке интерфейса. К таким факторам можно отнести технические и аппаратные ограничения разработки визуального интерфейса и взаимодействия с ним.

Факторы, влияющие на восприятие и взаимодействие с визуальным интерфейсом приложения, напрямую влияют и на правила проектирования и разработки интерфейсов. Выявив и исследовав такие факторы, можно разработать правила модификации готового интерфейса для решения задач, основанных на этих факторах.

Внешние факторы и их поведение не могут быть однозначно предсказаны на этапе проектирования визуального интерфейса мобильного приложения. К внешним факторам относят влияющие на восприятие и взаимодействие пользователя с интерфейсом приложения факторы, неявные по поведению и силе для разработчика. Со стороны пользователя приложения внешние факторы можно разделить по похожему принципу в зависимости от влияния пользователя на данные факторы. Выделим две основные группы факторов: зависимые от пользователя и независимые от пользователя.

Внешние зависимые факторы – зависят от поведения самого пользователя и могут изменятся под его влиянием, но для разработчика их влияние на этапе разработки будет неизвестно. Конечно, есть правила, выработанные с изучением готовых интерфейсов и пользовательским опытом их использования, но их роль статична и не способна адаптироваться под изменяющиеся факторы. Выделим следующие зависимые факторы: положение телефона; удерживающая рука; ошибочные нажатия; ошибочные переходы.

Внешние независимые факторы – не зависят от самого пользователя и способны изменяться под действием окружающей пользователя среды либо его физиологических особенностей. Выделим следующие независимые факторы: время суток; острота зрения *(полузависимый)*; мелкая моторика.

Некоторые факторы способны переходить из одной группы в другую, приобретая или теряя влияние пользователя на них, в данном случае следует предполагать, что такой фактор относится к той группе, где его влияние на восприятие и взаимодействие с интерфейсом является наиболее сильным. Такие факторы можно называть полузависимыми. В качестве примера можно привести независимый фактор «острота зрения» – физиологический и напрямую не зависящий от пользователя, но при желании он может быть им скорректирован.

Внутренние факторы включают в себя технические и аппаратные ограничения, встречаемые на этапе проектирования и разработки визуальных интерфейсов. Такие факторы являются предсказуемыми и могут быть учтены разработчиком без влияния на восприятие готового интерфейса его пользователем. Для примера одного из таких факторов можно рассмотреть разрешение экрана либо соотношение сторон. При разработке интерфейса существует возможность заранее определить разрешение определенного экрана либо группы устройств с различными разрешениями и, учитывая этот фактор, спроектировать такой интерфейс, который будет иметь одинаковый уровень восприятия и качество взаимодействия на данном устройстве либо на всех устройствах определенной группы.

Каждый фактор, связанный с восприятием и использованием определенного визуального интерфейса (интерактивного элемента), непосредственно влияет на возможные правила модификации данного интерфейса. На основе связи факторов, влияющих на пользователя интерфейса, и факторов, влияющих на модификацию данного интерфейса, можно провести прямую зависимость правил модификации от первоначальных факторов, связанных с пользователем.

ПРАВИЛА МОДИФИКАЦИИ ИНТЕРФЕЙСА

Полагаясь на исследование факторов, влияющих на восприятие и взаимодействие визуального интерфейса, ввиду непосредственной связи данных факторов с правилами построения графических интерфейсов можно выстроить правила их модификации на основе индивидуальных пользовательских характери-

стик. Это достигается за счет применения вышеупомянутых правил и собранных данных из мобильных приложений о влиянии факторов восприятия и взаимодействия к портрету пользователя и его поведенческой характеристике. Полученные правила модификации служат основой разработки методов модификации визуальных интерфейсов – создании системы, позволяющей, опираясь на поведение пользователя, модифицировать и адаптировать графический интерфейс под его потребности и специфику взаимодействия.

Используя внешние независимые факторы, можно выделить следующие пары «фактор – правило»:

Таблица 1. Внешние независимые факторы

Фактор	**Правило**
Время суток	Изменение цветовой палитры графического дизайна
Острота зрения *(полузависимый)*	Изменение размеров интерактивных элементов графического дизайна
Мелкая моторика	Изменение размеров интерактивных элементов графического дизайна, а также изменение их расположения

Время суток – изменение цветовой палитры графического дизайна: изменение времени суток приводит к изменению степени освещенности вокруг пользователя приложения, меняя визуальное восприятие интерфейса приложения и сказываясь на взаимодействии с ним. Опираясь на данные геопозиции, предоставляемые Android SDK, для корректировки локального времени устройства и степени освещенности можно выстроить правило адаптации цветовой палитры приложения к времени суток и освещенности [1, 2].

1) Дневное время – яркая палитра;
2) Ночное время – темная палитра;
3) Дневное время, слабая освещенность – яркая палитра, экран приглушен;*
4) Ночное время, яркая освещенности – яркая палитра, экран приглушен.*

** Для пунктов 3) и 4) используется встроенное в ОС Android правило регулировки яркости дисплея.*

Острота зрения – изменение размеров интерактивных элементов: плохое зрение приводит к осложнениям во взаимодействии пользователя с интерактивными и статичными элементами графического дизайна приложения. Данный фактор является полузависимым.

Опираясь на данные ОС Android о включенном режиме увеличения шрифтов либо режиме для близоруких (зависит от версии ОС Android), можно выстроить правило увеличения интерактивных элементов. Также данное правило может быть включено пользователем вручную внутри приложения, если подобное разрешено разработчиком.

Мелкая моторика – изменение размеров и расположения интерактивных элементов: мелкая моторика пальцев рук пользователя влияет на точность и достоверность взаимодействия с графическим интерфейсом приложения. Основываясь на успешных взаимодействиях, можно собрать карты переходов для различных комбинаций, для каждой карты переходов найти ошибочные взаимодействия.

Под картой понимается такая цепочка переходов, в которой пройден определенный путь от начальной точки до конечной, оканчивающийся успешным выполнением какой-либо функции интерфейса приложения.

Для каждой карты нужно составить правило изменения размера негативного или положительного элемента либо их расположение. Негативный элемент – тот элемент, по которому произведено ошибочное взаимодействие в данной карте, соответственно, положительный – успешное взаимодействие для достижения конечной цели данной карты.

Используя внешние зависимые факторы, можно выделить следующие пары «фактор – правило»:

Таблица 2. Внешние зависимые факторы

Фактор	**Правило**
Положение телефона	Изменение ориентации интерфейса
Удерживающая рука	Изменение расположения интерактивных элементов графического дизайна
Ошибочные нажатия	Изменение расположения, размеров интерактивных

	элементов графического дизайна
Ошибочные переходы	Пропуск определенных взаимодействий с интерфейсом, либо его изменение

Положение телефона – изменение ориентации интерфейса: данная связка фактора и правила поведения интерфейса используется повсеместно, где допускается такой метод адаптации интерфейса под положение устройства. Однако не под все задачи такой вариант подходит, запрет на использование данного правила может быть включен как разработчиком, так и самим пользователем, используя встроенную функцию изменения ориентации интерфейса в ОС Android.

Удерживающая рука – изменение расположения интерактивных элементов графического дизайна: на основе данных, получаемых с датчиков устройства, либо ручной настройке работы данного правила изменять расположение графических интерфейсов мобильного приложения. Данный фактор позволяет применять правило в зависимости от расположения телефона на поверхности ладони, то есть определять, какой рукой пользователь взаимодействует с интерфейсом, и модифицировать интерфейс приложения, отталкиваясь от полученных данных.

Для фактора удерживающей руки можно выделить 4 правила модификации интерфейса: *левая рука У*/В**, правая рука У/В, левая рука У/правая рука В, правая рука У/левая рука В (У* – удерживающая рука – рука в которой пользователь держит телефон; В** – взаимодействующая рука – рука которой пользователь взаимодействует с интерфейсом; У/В – одна рука выполняет обе функции).*

Ошибочные переходы – изменение реакции интерфейса на ошибочные взаимодействия: распознавание ошибочных переходов, как фактора влияния на построение правил модификации реакций интерфейса. На основе карт переходов выявить типичные ошибочные переходы. Ошибочный переход возможно распознать как переход на другой элемент интерфейса и мгновенный возврат назад, такой ошибочный переход можно воспринимать как петлю.

На определенных картах переходов с часто встречающимися петлями изменять отклик интерфейса на переход в данную петлю. Примером изменения такого отклика служит задержка перехода на данный элемент, который в дан-

ной карте является петлей. Это позволит пользователю успеть сделать правильное взаимодействие.

Ошибочные нажатия – изменение размеров и расположения интерактивных элементов: распознавание ошибочных взаимодействий с интерфейсом приложения и последующее влияние их как фактора построения правил модификации интерфейса. Ошибочное нажатие – близкий фактор к ошибочному переходу, различием между двумя данными факторами является конечный отклик интерфейса на взаимодействие с ним. В данном случае итоговым откликом будет выполнение определенного действия (конечное взаимодействие), исключающего переход на другие элементы графического интерфейса. Частным примером такого конечного отклика является конечное взаимодействия в картах переходов. Соответственно, для такого фактора, подобного другому, возможно применение правил подобного фактора с поправкой на возможные взаимодействия и отклики интерфейса.

МЕТОДЫ МОДИФИКАЦИИ ИНТЕРФЕЙСА

Построение методов модификации визуальных интерфейсов требует сбора данных о взаимодействии пользователя с интерфейсом приложения. Обозначим следующие основные типы взаимодействий, основываясь на специфике Android приложения:

- Загрузка Activity;
- Загрузка Fragment;
- Переход на другую Activity;
- Переход на другой Fragment;
- Закрытие Activity;
- Закрытие Fragment;
- Нажатия на элементы интерфейса;
- Длительные нажатия на элементы интерфейса;
- Ввод данных с помощью клавиатуры;
- Ввод данных с помощью элементов интерфейса;
- Скроллинг списка;
- Скроллинг экрана.

Собранные данные об активности пользователя требуются для применения правил модификации интерфейса [3].

Рассмотрим пары «Фактор – Правило» из раздела №2, использующие методы модификации, основанные на пользовательском поведении, и построим для них правила, выделенные при помощи собранных данных о пользователе и его взаимодействии с Android приложением.

1) Изменение расположения элементов графического интерфейса в зависимости от руки, удерживающей телефон.

Для данного метода требуется собрать данные с датчиков наклона мобильного устройства. На основе показателей датчиков выявить их соответствие для правой или левой руки. Проведя анализ полученных показателей конкретного пользователя за некоторый промежуток времени, нужно определить характерную руку, удерживающую телефон при взаимодействии с интерфейсом приложения. Учитывая тот факт, что проектирование интерфейса предполагает правую руку как основную, модифицировать расположение наиболее труднодоступных элементов при изменении руки на левую. Важно отметить, что необходима определенная задержка при обработке данных с датчиков, чтобы исключить внешние воздействия на телефон или положение пользователя. При совсем неестественных показателях – отключить данное правило.

Пример: пользователь лежит на боку и использует устройство правой рукой, но при этом правило может считать, что пользователь держит устройство левой из-за угла наклона.

2) Изменение расположения, размеров интерактивных элементов графического дизайна из-за ошибочных нажатий.

Для данного метода требуется собрать данные с элементов интерфейса, которые имеют обработку нажатий. На основе собранных данных выявить петли, при которых пользователь, после нажатия на один из элементов, быстро взаимодействует с расположенным рядом элементом для достижения требуемого результата. Дополнительное влияние на весомость такой петли может влиять размер конечного элемента взаимодействия – более маленький элемент и количество связанных петель явно указывают на возможность ошибочных нажатий. После сбора достаточного количества данных определить элементы, требующие модификации их расположения или размера.

Изменение размера происходит в следующих случаях: а) конечный элемент взаимодействия существенно меньше элемента, на которого приходится ошибочное нажатие; б) элементы имеют равный размер. В случае а) происходят пропорциональное изменение конечного элемента в большую сторону, а ошибочного – в меньшую. В случае б) происходят изменение размера ошибочного элемента в меньшую сторону и увеличение отступа между ними.

Изменение расположения происходит в следующих случаях: а) конечный элемент взаимодействия равен ошибочному и элементы расположены рядом; б) элементы расположены через какой-либо графический элемент. В случае а) происходит увеличение отступа между элементами при сохранении размеров внешнего элемента (блока). В случае б) происходит перемещение одного элемента на место другого при условиях: элементы расположены в одном внешнем элементе, элементы имеют сопоставимые размеры.

Данный метод не может быть применено по умолчанию к элементам динамически обновляемого списка, но может быть применено к содержащихся в них элементам интерфейса.

3) Изменение расположения, размеров интерактивных элементов графического дизайна либо изменение их поведения в случае ошибочных переходов.

Данный метод наследуется от предыдущего пункта, однако оно имеет дополнительные методы модификации элементов интерфейса.

При обнаружении ошибочных петель на элементах, для которых результатом взаимодействия пользователя является переход на другой экран (Activity/Fragment) и частом их повторении. Для данных элементов применяется задержка срабатывания, если в момент этой задержки пользователь переходит или взаимодействует с предполагаемым правильным элементом интерфейса, то обработка срабатывания ошибочного элемента прерывается.

4) Изменение цветовой палитры интерфейса при ошибочных нажатиях и переходах, а также включенном режиме плохого зрения.

Комбинация из двух предыдущих правил, а также включенном на устройстве или в приложении режима увеличения интерфейса. При условии выполнения всех правил элементы, отмеченные определенным образом, могут становиться другого цвета, для увеличения их заметности на фоне других элементов, при этом нужно учитывать требования Android Guidelines, если приложение ис-

пользует интерфейс, основанный на них. Пример: кнопка отправки сообщения в мобильном приложении меняет свой цвет на контрастный к палитре [4, 5].

ЗАКЛЮЧЕНИЕ

Исследованы факторы, влияющие на взаимодействие пользователя, и выявлены правила модификации интерфейса. Результатом стала разработка методов модификации визуальных интерфейсов Android-приложений на основе индивидуальных пользовательских характеристик. На основе результатов данной работы можно реализовать методы модификации визуальных интерфейсов и интегрировать их в процесс разработки Android приложений с адаптивным интерфейсом.

СПИСОК ЛИТЕРАТУРЫ

1. *Lars Vogel.* Android Location API with the Fused Location Provider – Tutorial. URL: http://www.vogella.com/tutorials/AndroidLocationAPI/article.html

2. Sensors Overview. URL: https://developer.android.com/guide/topics/sensors/ sensors_overview.html

3. *Lars Vogel.* Introduction to Background Processing in Android – Tutorial. URL: http://www.vogella.com/tutorials/AndroidBackgroundProcessing/article.html

4. Material.io: Usability – Accessibility. URL: https://material.io/guidelines/usability/accessibility.html

5. *Mads Soegaard*. Adaptive vs. Responsive Design. URL: https://www.interaction-design.org/literature/article/adaptive-vs-responsive-design.org

METHODS OF ANDROID APPLICATIONS' VISUAL INTERFACES MODIFICATIONS BASED ON INDIVIDUAL USER CHARACTERISTICS

A. M. Sarmatin[1], I. S. Shakhova[2]

Higher School of Information Technologies and Intelligent Systems at Kazan (Volga region) Federal University

[1]antonsarmatin@googlemail.com, [2]is@it.kfu.ru

Abstract

The factors that influence the modification of visual interfaces are analyzed. The rules of modification of the considered factors on the basis of individual user characteristics are proposed. Methods for modifying the visual interfaces of Android applications have been developed.

Keywords*: android, UI, user interface, user interface, visual interface, mobile applications*

REFERENCES

1. *Lars Vogel.* Android Location API with the Fused Location Provider – Tutorial. URL: http://www.vogella.com/tutorials/AndroidLocationAPI/ article.html

2. Sensors Overview. URL: https://developer.android.com/guide/topics/sensors/sensors_overview.html

3. *Lars Vogel.* Introduction to Background Processing in Android – Tutorial. URL: http://www.vogella.com/tutorials/AndroidBackgroundProcessing/ article.html

4. Material.io: Usability – Accessibility. URL: https://material.io/guidelines/usability/accessibility.html

5. *Mads Soegaard*. Adaptive vs. Responsive Design. URL: https://www.interaction-design.org/literature/article/adaptive-vs-responsive-design.org

СВЕДЕНИЯ ОБ АВТОРАХ

САРМАТИН Антон Михайлович – студент Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета, андроид-разработчик.

Anton Mikhailovich SARMATIN – student of the Higher School of Information Technologies and Intelligent Systems at Kazan Federal University, Android developer.

email: antonsarmatin@googlemail.com

ШАХОВА Ирина Сергеевна – ассистент кафедры программной инженерии Высшей школы информационных технологий и интеллектуальных систем Казанского федерального университета. Сфера научных интересов – цифровые образовательные системы, индивидуализация образования, мобильное обучение.

Irina Sergeevna SHAKHOVA – teacher of the Higher School of Information Technologies and Intelligent Systems at Kazan Federal University. Research interests include digital educational systems, individualization of education, mobile learning.

email: is@it.kfu.ru

Материал поступил в редакцию 5 июня 2018 года

УДК 004.414.3

О НЕСКОЛЬКИХ МЕТОДАХ И ИНСТРУМЕНТАХ АНАЛИЗА КАЧЕСТВА УЧЕБНОГО ПРОЦЕССА

Е. А. Свинтенок[1], Б. Е. Попов[2], М. М. Абрамский[3]

[1-3] *Высшая школа информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета*

[1]svintenok@gmail.com, [2]ibogdanpopov@gmail.com, [3]ma@it.kfu.ru

Аннотация

Рассмотрены вопросы анализа связи учебного расписания с успеваемостью студентов, а также определения сложности учебного курса. Выделены факторы, которые стоит отслеживать при подобном анализе. Предложены идеи применения этих данных в системах управления образовательным процессом.

***Ключевые слова**: сложность курса, анализ расписания, системы управления образованием, анализ данных*

ВВЕДЕНИЕ

Задача составления удобного и сбалансированного расписания занятий в вузе продолжает в настоящее время оставаться важной и актуальной. Хорошее расписание позволило бы повысить эффективность образовательного процесса для студентов и преподавателей, а также снизить возможные издержки на материальные ресурсы, оборудование и др. [1]. При этом стоит отметить, что зачастую изменение расписания происходит не на этапе его построения, а на этапе его эксплуатации – когда в процессе учебы обнаруживаются нестыковки, неудобство времени/места проведения занятия. Изменения расписания могут вызывать как положительные, так и отрицательные эффекты в процессе обучения. Использование технологий анализа данных показало свою эффективность в системах принятия решений, в том числе в образовательной сфере [2, 3]. В настоящей работе поставлена задача каталогизации критериев влияния отличительных особенностей расписания на результаты обучения студентов.

Также отметим вопрос сложности учебного курса как понятие, которое не может быть идентифицировано по расписанию, но вместе с последним может влиять на результаты освоения дисциплины. Сложность курса может позволить

установить местоположение предмета в расписании таким образом, что суммарная сложность курсов не будет сильно различаться в разные дни, то есть не будет высоконагруженных дней для студента, следовательно, ему будет проще осваивать учебный материал.

Статья построена следующим образом: в первом разделе рассмотрены задачи анализа расписания и выявления связей между ним и успеваемостью студентов, во втором разделе дано определение сложности курса и описаны факторы, влияющие на него, а в третьем приведено описание программ, созданных для сбора данных и нужных для решения названных выше задач.

1. АНАЛИЗ УЧЕБНОГО РАСПИСАНИЯ

В основе идеи данной работы лежит возможность применения анализа данных для выявления связей между расписанием и показателями эффективности управления образовательным процессом, такими, как успеваемость студентов. Полученные в ходе анализа данные могут быть использованы для дальнейшей оптимизации учебного расписания.

В ходе проведегнных исследований был определен высокоуровневый алгоритм работы с данными для выявления влияния расписания на успеваемость студентов (рис. 1).

Рис. 1. Последовательность работы с данными для выявления влияния расписания на успеваемость

Сначала необходимо выделить значимые факторы учебного расписания и показатели эффективности образовательного процесса, между которыми и нужно

исследовать связи. В качестве исходных данных для анализа было взято расписание Высшей школы ИТИС КФУ [4]. Ниже приведен пример выделенных факторов:

1. *Равномерность нагрузки расписания* – среднеквадратическое отклонение по количеству пар группы в день;
2. *Усредненное время пар относительно суток* – сумма индексов пар, деленная на количество пар, где индекс пары – ее порядковый номер (пара в 8:30 – 1, пара в 10:10 – 2 и т. д.);
3. *Усредненное количество пар без перерывов;*
4. *Количество библиотечных дней;*
5. *Равномерность соотношения количества пар по типам* (лекция/практика) – среднеквадратическое отклонение абсолютной величины разности количества лекций и практик у группы в день;
6. *Порядок пар относительно их типов (лекция/практика)* – отношение усредненного времени лекций к усредненному времени практик (расчет усредненного времени каждого типа пар производится по формуле, описанной в факторе №2);
7. *Соотношение профильных и непрофильных предметов.*

Набор факторов, характеризующих расписание учебного курса группы:

1. *Количество часов в неделю на курс;*
2. *«Склеенность» пар курса;*
3. *Расстояние между парами;*
4. *Усредненное время пар курса;*
5. *Преподаватель курса.*

Группа признаков рейтинга студентов:

1. *Средняя общая успеваемость студентов группы;*
2. *Средняя успеваемость студентов группы по курсу.*

Следующими шагами являются сбор и извлечение нужных для анализа данных из входных источников, в приведенном выше случае – расписания и учебного рейтинга студентов. Составляющие результативных данных этих шагов зависят от факторов, выделенных на предыдущем шаге.

Далее необходимо произвести расчет значений факторных и результативных признаков на основе извлеченных данных, формируя выборку по группам студентов. Ниже представлен пример полученных значений факторных признаков,

рассчитанных по данным расписания студентов Высшей школы ИТИС КФУ для нескольких групп третьего (группы 11-4**) и четвертого курсов (группы 11-5**) (рис. 2). Не углубляясь в анализ, по данным таблицы можно увидеть, например, что нагрузка на четвертом курсе в целом распределена по неделе более равномерно, чем на третьем (т. к. у соответствующих групп ниже значения признака days_load_std), а вот «окон» в расписании четвертого курса больше (признак window_counts). Также по показателям признака pair_times_mean можно заметить, что пары на четвертом курсе немного больше смещены в вечернее время, чем пары третьего курса.

	11-406	11-407	11-408	11-501	11-502	11-503
days_load_std	0.942809041582	0.942809041582	1.699673171197	1.771690968789	1.771690968789	1.572330188676
lec_prac_ratio_std	1.374368541872	1.374368541872	1.374368541872	0.763762615825	0.897527467855	0.5
library_days	0.0	0.0	0.0	0.0	0.0	0.0
pair_count_in_row_mean	2.25	2.25	2.0	3.5	3.666666666666	4.4
pair_times_mean	4.423076923076	4.423076923076	4.423076923076	3.72	3.8	3.72
window_counts	3.0	3.0	2.0	1.0	1.0	0.0

Рис. 2. Пример подсчитанных факторных признаков расписания группы

По выборкам, полученным на предыдущем этапе, можно произвести подсчет коэффициентов корреляции между факторными и результативными признаками учебного процесса групп. Имея коэффициенты корреляции, можно произвести анализ влияния учебного расписания на результативные признаки.

2. СЛОЖНОСТЬ КУРСА

Сложность курса – это величина, которая показывает разность между актуальным опытом обучающегося и уровнем предмета. Также она характеризуется временем, выделяемым на изучение материала, количеством и сложностью выполняемых заданий на практических занятиях.

Для определения сложности курса был выбран метод Майера [5], позволяющий вычислить сложность материала курсов и затем собрать данные о предмете:

- *Оценки преподавателя в системе;*
- *Стаж работы преподавателя;*
- *Сколько времени преподается курс;*
- *Количество лекционных часов;*

- *Количество практических часов;*
- *Количество заданий тестовых;*
- *Средняя сложность тестовых заданий (оценка преподавателя от 1 до 10);*
- *Количество творческих заданий;*
- *Средняя сложность творческих заданий;*
- *Количество задач;*
- *Средняя сложность задач;*
- *Форма сдачи;*
- *Среднее количество набранных баллов;*
- *Медианное количество баллов.*

Метод Майера был расширен еще одной метрикой для определения сложности. При оценке сложности использовались Story Points («очки историй») [6] – абстрактные меры сложности, используемые при оценке сложности задач в IT-проектах. Story Points – это общая оценка нескольких экспертов, которые должны в результате обсуждения прийти к согласию.

3. СБОР ДАННЫХ

Вопрос применения идей, описанных выше, тесно связан с проблемой сбора информации о расписании, учебных курсах и показателях успеваемости студентов. Именно поэтому были написаны скрипты извлечения данных из двух источников: Google-таблиц, в которых хранятся расписание занятий и рейтинг успеваемости студентов, а также из рабочих программ учебных дисциплин, которые хранятся в pdf-формате на сайте университета. На выходе указанные программы возвращают извлеченные данные в форматах, удобных для использования в IT-проектах.

Для извлечения данных из Google-таблиц был написан скрипт на языке программирования Python, производящий парсинг данных из исходных таблиц в файлы формата JSON[4]. Сначала скрипт производит экспорт Worksheet Google-таблицы в .xls файл с помощью библиотеки Pygsheet, затем – парсинг данных из xls-формата в набор словарей и массивов с помощью библиотеки xlrd, и в завершении сериализует объект с данными в JSON-формат. Пример формата данных на выходе приведен на рис. 3.

[4] JSON - JavaScript Object Notation

```
{
    "11-601": {
        "понедельник": {
            "08.30-10.00": {
                "text": "Иностранный язык",
                "merged": true
            },
            "10.10-11.40": {
                "text": "Алгебра и геометрия Арсланов М.М. в ауд.109 к.2",
                "merged": true
            },
            "11.50-13.20": {
                "text": "Алгоритмы и структуры данных Абрамский М.М."
            },
            "13.35-15.05": {
                "text": "Физическая культура \nУНИКС \n14.00-15.30",
                "merged": true
            },
            "17.00-18.30": {
                "text": "Soft skills \"Краткий курс молодого бойца \" в 109 к.2",
                "merged": true
            }
        },
```


Рис. 3. Пример получаемых скриптом данных о расписании

В целях извлечения данных о курсах и предметах из рабочих программ учебных дисциплин была написана программа на языке Java, которая использует библиотеки PdfBox и pdftable (код находится в соответствующем пакете программы, взят из https://github.com/rostrovsky/pdf-table). Она производит парсинг требуемых данных об учебной дисциплине, таких, как название, количество часов, темы и т. д.

```
{
  "code" : "09.03.03",
  "year" : "2017",
  "profile" : "не предусмотрено",
  "independent_work_structure" : [ {
    "week_num" : "1-2",
    "num" : "1.",
    "laboriousness" : "8",
    "section" : "Тема 1. Введение в алгоритмизацию",
    "semester" : "1",
    "control_form" : "домашнее задание",
    "type" : "подготовка домашнего задания"
  }, {
    "week_num" : "3-4",
    "num" : "2.",
    "laboriousness" : "8",
    "section" : "Тема 2. Введение в язык Java и среду JDK. Используемые типы данных.",
    "semester" : "1",
    "control_form" : "домашнее задание",
    "type" : "подготовка домашнего задания"
  }, {
```


Рис. 4. Пример получаемых программой данных об учебной дисциплине


```
"language" : "русский",
"reviewers" : "Таланов М.О.",
"qualification" : "бакалавр",
"number" : "689510117",
"competence" : [ {
  "cipher" : "ПК-17 (профессиональные компетенции)",
  "description" : "способностью принимать участие в управлении проектами создания ин
}, {
  "cipher" : "ПК-2 (профессиональные компетенции)",
  "description" : "способностью разрабатывать, внедрять и адаптировать прикладное пр
}, {
  "cipher" : "ПК-8 (профессиональные компетенции)",
  "description" : "способностью программировать приложения и создавать программные п
} ],
"form" : "очное",
"classroom_work_structure" : [ {
  "week_num" : "1-2",
  "lab_hours" : "4",
  "lec_hours" : "2",
  "practice_hours" : "0",
  "num" : "1.",
  "section" : "Тема 1. Введение в алгоритмизацию",
  "semester" : "1",
  "control_form" : "Письменное домашнее задание  "
```


Рис. 5. Пример получаемых программой данных об учебной дисциплине

ЗАКЛЮЧЕНИЕ

Получены классификация и способ вычисления факторов, которые нужно учитывать при анализе зависимости учебного расписания и результатов обучения, также дан подход к определению сложности курса. Полученные результаты планируется применить в системах управления образованием при построении приложений для оптимизации расписания, выбора курсов, изменения учебных планов предметов и в целом для улучшения образовательного процесса.

СПИСОК ЛИТЕРАТУРЫ

1. *Бронникова, Н.* Проблемы составления расписания в вузе // Ректор вуза. 2015. №7. С. 8.

2. *Горлушкина Н.Н.* Задачи и методы интеллектуального анализа образовательных данных для поддержки принятия решений // Образовательные технологии и общество. 2015. Т. 18, № 1. С. 472–482.

3. *Пиотровская К.Р., Тербушева М.В.* Интеллектуальный анализ данных в педагогической аналитике // Образовательные технологии и общество. 2016. Т. 2(96). С. 10–14.

4. Учебное расписание Высшей школы ИТИС КФУ. URL: https://docs.google.com/spreadsheets/d/1DHir9K8KO8a2AX3AfPiE422HXgf_7AKgSOSS-UOMt_A

5. *Майер Р.В.* Контент-анализ школьных учебников по естественно-научным дисциплинам. Глазов: Глазов. гос. пед. ин-т, 2016. 137 с.

6. *Davidson D.* Why do we use Story Points for Estimating // SCRUM.ORG: Home page of Scrum, 2014. URL: https://www.scrum.org/resources/blog/why-do-we-use-story-points-estimating.

ABOUT SEVERAL METHODS AND TOOLS FOR THE EDUCATIONAL PROCESS QUALITY ANALYSIS

E. A. Svintenok[1], B. E. Popov[2], M. M. Abramskiy[3]

[1-3] *Higher School of Information Technologies and Intelligent Systems, Kazan Federal University*

[1]svintenok@gmail.com, [2]ibogdanpopov@gmail.com, [3]ma@it.kfu.ru

Abstract

This article raises questions of the determining course complexity and analyzing schedule and academic performance relations. Also there are emphasized important constituents of the courses and introduced ideas of the use cases of the allotted data.

Keywords*: course complexity, schedule, academic performance, educational management systems, data analysis*

REFERENCES

1. *Bronnikova N.* Problemy sostavleniya raspisaniya v vuze // Rektor vuza. 2015. №7. S. 8.

2. *Gorlushkina N.N.* Zadachi i metody intellektual'nogo analiza obrazovatel'nykh dannykh dlya podderzhki prinyatiya reshenij // Obrazovatel'nye tekhnologii i obshhestvo. 2015. T. 18, № 1. S. 472–482.

3. *Piotrovskaya K.R., Terbusheva M.V.* Intellektual'nyi analiz dannykh v pedagogicheskoi analitike // Obrazovatel'nye tekhnologii i obshchestvo. 2016. V. 2 (96). S. 10–14.

4. Learning Schedule of Higher School of ITIS KFU. URL: https://docs.google.com/spreadsheets/d/1DHir9K8KO8a2AX3AfPiE422HXgf_7AKgSOSS-UOMt_A

5. *Majer R.V.* Kontent-analiz shkol'nykh uchebnikov po estestvenno-nauchnym distsiplinam. Glazov: Glazov. gos. ped. in-t, 2016. 137 s.

6. *Davidson D.* Why do we use Story Points for Estimating // SCRUM.ORG: Home page of Scrum, 2014. URL: https://www.scrum.org/resources/blog/why-do-we-use-story-points-estimating

СВЕДЕНИЯ ОБ АВТОРАХ

СВИНТЕНОК Екатерина Анатольевна – студентка Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета.

Ekaterina Anatolevna SVINTENOK – student of Higher School of ITIS KFU

email: svintenok@gmail.com

ПОПОВ Богдан Евгеньевич – студент Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета.

Bogdan Evgenievich POPOV – student of Higher School ITIS KFU

email: ibogdanpopov@gmail.com

АБРАМСКИЙ Михаил Михайлович – старший преподаватель кафедры программной инженерии Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета.

Mikhail Mikhailovich ABRAMSKIY – senior lecturer at Department of Software Engineering of Higher School of ITIS KFU

email: ma@it.kfu.ru

Материал поступил в редакцию 2 августа 2018 года

УДК 004.912+004.021+004.023

ИЗВЛЕЧЕНИЕ ЗАГОЛОВКОВ ИЗ PDF-ДОКУМЕНТОВ НАУЧНОЙ ТЕМАТИКИ

Д. С. Филиппов

Высшая школа информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета
dmitriyfil1995@gmail.com

Аннотация

Актуальность представленного исследования обусловлена бедностью существующих подходов к извлечению заголовков из PDF-документов, предложенных в более ранних исследованиях, которые используют либо машинное обучение, либо простые эвристики. Цель настоящего исследования – предоставить более проработанные подходы к общей задаче извлечения заголовка документа и предложить лучший алгоритм выделения его из документов научной тематики. Основная методика, использованная нами при выборе решения, – рассмотреть, как можно большее количество различных ситуаций относительно форматирования заголовка, возникающих в разных документах, и предложить решение для каждой из них, а затем обобщить их в полноценный подход. Результаты выбранного подхода показали его эффективность по сравнению с методами других исследователей, если в нашем распоряжении находятся документы с различными вариациями оформления, структурной организации и форматирования. Данное исследование показало, что глубокое исследование задачи – перспективный путь для разработки лучших решений и инструментов. Статья будет полезна исследователям и разработчикам, которые часто встречаются с проблемой извлечения заголовков как одной из подзадач анализа документов.

Ключевые слова*: Pdf processing, title extraction, header extraction, strategy based approach, title heuristic, structural analysis, style information, text analysis, document analysis, information extraction, анализ текстов, автоматическая обработка документов, структурирование информации.*

ВВЕДЕНИЕ

В последние несколько лет активно развивается тенденция к структурированию информации, основную часть которой составляют текстовые документы. Так как подавляющая часть этих документов находится в формате PDF (Portable Document Format), то задача их обработки и анализа является как никогда актуальной. В число распространённых подзадач обработки входит проблема автоматического формирования библиотечных записей (метаданных), например, применительно к электронным библиотекам. Библиотечные записи вмещают в себя такую информацию, как авторы публикации, год, место издания, тип издания и, в частности, заголовок. Структура и содержание библиотечной записи, её взаимосвязь с информацией на титульных страницах публикации являются весьма интересной темой и заслуживают отдельного исследования.

Извлечение заголовка является также необходимым компонентом выделения и анализа структуры документа. В частности, комплексный структурный анализ документа может включать в себя: выделение заголовка, структуризацию оглавления, суммаризацию и структурирование контента с целью извлечения сущностей определённых типов. Например, моделирование и формализация публикаций в области математики, описанные в [7–9], могут применяться в сочетании с выделением заголовка для составления картотеки обработанных публикаций, формирования метаданных, при составлении тренировочных и тестовых корпусов и для иных целей. Активно применяется и обобщённый анализ научных публикаций различной тематики с опорой на корпуса статей известных журналов и конференций, таких, как ACM, IEEE и другие [10–15].

Однако даже задача выделения заголовка является не такой простой, как может показаться на первый взгляд. Имеется множество проблем, которые не только затрудняют анализ PDF-документов в целом, но и усложняют задачу анализа и выделения заголовков. Среди них, например, отсутствие текстового слоя, пользовательская кодировка, разнообразное и иногда весьма специфическое оформление титульных страниц, расположение и форматирование заголовка и другие. Поэтому данная работа посвящена извлечению заголовков из PDF-документов, а также возникающим при этом проблемам и возможными подходами к их разрешению. Для некоторых из проблем, освещённых в настоящей

работе, варианта решения предложено не будет, причиной чему являются технологические ограничения, для преодоления которых требуются отдельные исследования и разработки. В данной статье рассмотрены различные случаи, касающиеся специфического оформления и структурной организации заголовка.

Для извлечения заголовков из документов исследователи предлагают ряд методов. Большинство из них основано на одних и тех же принципах [1–2], [4] или близких техниках [3, 4], [10], [15, 16]. Многие предлагают готовые инструменты в виде приложений или веб-сервисов для множества задач, включая извлечение заголовка [2–6], [10], [14–16].

Большинство предложенных методов опирается на одну единственную эвристику: заголовок расположен на первой странице в блоке текста с самым большим шрифтом. Если таких блоков несколько, выбирается наидлиннейший. Другие же предлагают использовать алгоритмы машинного обучения. Серьёзный недостаток таких подходов состоит в том, что даже небольшие вариации в формате или структурной организации документа с высокой вероятностью приведут к неправильным результатам, извлечению только части заголовка, или же вовсе алгоритмы завершат свою работу неудачно. Нами предложен способ извлечения, преимущественно опирающийся на стилевую информацию текста документа, ищущий заголовки не только на первой странице и, кроме того, более устойчивый к флуктуациям форматирования заголовка. Необходимо отметить, что данный метод применим только к документам научной тематики, множество которых будет рассмотрено ниже в секции «Описание целевых документов», так как другие типы документов могут иметь серьёзные отличия в стилевом оформлении и структуре титульных страниц. Для остальных документов предложен механизм выбора стратегий извлечения заголовков на основе их жанров и типовых особенностей. Также даны рекомендации для некоторых из стратегий. Кроме того, обсуждены особенности обработки документов в других форматах (офисных, HTML и других).

Таким образом, основная цель настоящего исследования – предложить набор более качественных эвристик для извлечения заголовка из PDF-документов научной тематики, таких, как методички, учебники, научно-популярная литература и др., потому что методов, предложенных более ранни-

ми исследованиями, недостаточно для успешной обработки из Всемирной паутины научных книг в формате PDF.

Дальнейшее изложение организовано следующим образом: вначале идут описание и обоснование подхода, основанного на стратегиях, далее – более подробное описание свойств целевых документов, к которым будет применяться основной алгоритм извлечения. Третья секция перечисляет основные технические проблемы, возникающие при обработке и, в частности, извлечении заголовка из PDF-документов. Четвёртая секция предлагает возможные решения для этого, а также подробно описывает метод выделения заголовков из документов научной тематики. Последние секции показывают результаты описываемого алгоритма на тестовом множестве PDF-документов и сравнивают их с результатами, полученными с использованием методик предыдущих исследований. Также в последней секции сделаны замечания о других возможных стратегиях извлечения заголовка из документов в других форматах.

ОБЩИЙ ПОДХОД К ИЗВЛЕЧЕНИЮ

В различных по стилю и тематике документах титульные страницы и заголовки, в частности, оформляются по-разному. Отличия порой настолько велики, что говорить о методе выделения заголовков, общем для всех форматов и типов документов, не приходится. Даже в пределах одной тематики различия в оформлении могут быть критическими. Для решения рассматриваемой задачи в общем виде можно было бы использовать машинное обучение, как это сделано в [6]. Но тогда на выходе получится «тяжёлая» модель, которую будет трудно изменять и сопровождать. При этом каждое изменение повлечёт за собой переобучение модели с возможными регрессиями. Поэтому оптимальным для этой задачи (как и для многих других в области анализа неструктурированных документов) представляется подход, основанный на стратегиях.

В данном контексте будем понимать стратегию как отдельный алгоритм, рассчитанный на успешное выполнение задачи (выдачу правильного результата) для некоторого набора частных случаев. В контексте настоящей статьи частный случай определяется набором следующих параметров: стилистика документа (научная, деловая, художественная, публицистическая), его тип (статья, исследование, учебник, книга, собрание сочинений и другие), свойства оформления,

формат документа (DOCX, PDF, rich text formats, plain text, HTML). Тогда при обнаружении нового частного случая все старые стратегии остаются без изменений, и всё, что нужно, – добавить новую стратегию, учитывающую этот частный случай.

Таким образом, общая задача выделения заголовка (вне зависимости от формата) сводится к следующему: определяется и реализуется набор стратегий для известных частных случаев. Документ обрабатывается каждой из этих стратегий, пока не будет получен ненулевой результат (нулевой результат – значит, заголовок не найден). Возможна также вариация: вначале документ проходит предобработку с целью выделить лишь некоторое подмножество (в идеале одну) стратегий обработки, чтобы минимизировать риск получения лишних, неправильных результатов из других стратегий.

Вышеописанный подход и применен ниже, а именно, определена стратегия для извлечения заголовков из документов, удовлетворяющих условиям, описанным в следующем разделе.

ОПИСАНИЕ ЦЕЛЕВЫХ ДОКУМЕНТОВ

Как уже было сказано ранее, для анализа будут браться только документы научной тематики в формате PDF. Помимо этого, такие документы должны обладать следующими свойствами: иметь одну или несколько выделенных титульных страниц, где размещается заголовок, который, в свою очередь, будет иметь отличительные признаки, по которым его можно однозначно идентифицировать как заголовок. Допустимы документы на английском и русском языках.

ОПИСАНИЕ ПРОБЛЕМНЫХ СИТУАЦИЙ

Теперь, когда множество целевых документов описано, можно подробно рассмотреть стратегию выделения заголовка. Во многих исследованиях, например, [2–4], авторы опираются на единственную эвристику – заголовком является текстовый блок с самым крупным шрифтом на первой странице или, если таких несколько, самый крупный из них. Однако есть ряд существенных и довольно распространённых отклонений от этого простого правила, которые будут рассмотрены в настоящем разделе.

Первая категория проблем связана с форматом. Формат PDF изначально не был предназначен для автоматической обработки. Хотя его последние версии

поддерживают включение метаданных, облегчающих автоматическую обработку, подавляющая часть документов не содержит этих данных, и приходится прилагать существенные усилия, чтобы точно выделить необходимую информацию. Кроме того, есть следующие факторы, препятствующие успешной обработке таких документов:

- Текст, особенно на титульных страницах, может быть представлен в виде иллюстрации, и тогда единственный способ считать его – это OCR (Optical Character Recognition). Документы, для которых данная особенность актуальна, не будут в полной мере учитываться в этой статье. Однако при наличии OCR-технологии, позволяющей извлекать не только чистый текст, но и стилевую информацию, с ограничениями, допустимыми для успешного выполнения задачи, можно также обрабатывать подобные документы, используя алгоритм, описанный ниже.

Следует отметить, что некоторые из документов с титульной страницей, оформленной в виде обложки, можно обработать и без OCR-инструментов. В данном случае необходимо использовать информацию, считываемую с других титульных страниц, о чём будет сказано ниже.

- PDF-документ имеет внутреннюю кодировку, и считывание текста в первоначальном виде невозможно. Такой документ не может быть обработан ввиду отсутствия информации об этой внутренней кодировке в самом документе.
- На титульных страницах есть рисунки с текстовыми подписями, вносящие искажения и лишний текст.
- Искажения при считывании текста, связанные с неточной шириной и расстановкой пробелов, из-за чего одно слово может быть разделено на несколько слов или, наоборот, несколько слов могут быть склеены в одно. Возможны и смешанные случаи. Иногда также встречается ситуация, когда строка разделена по буквам, что характерно в большей степени для многострочных заголовков, состоящих из коротких слов.

Вторая проблема состоит в том, что заголовок не обязательно будет на первой странице. Фактически документы можно разделить на четыре категории: с одной, двумя, тремя титульными страницами и когда у документа есть не-

сколько страниц с вступительным текстом, а уже затем идёт заголовок. Часто первая страница оформлена, как обложка, и текст из неё выделить трудно или практически невозможно ввиду проблем из первой категории, описанной выше. Кроме того, возможны документы, у которых на разных страницах указан различный заголовок (например, в сокращённом варианте на первой странице и в полном варианте – на следующей).

Следует помнить, что указанные категории весьма условны и будут содержать самые разные вариации от документа к документу. К примеру, на иллюстрации 1 приведён документ с двумя заголовками на разных языках, расположенных на одной странице.

Иллюстрация 1. Пример документа с двумя заголовками на одной странице (одна из многочисленных вариаций титульных страниц


БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

АДМИНИСТРАЦИЯ ПАРКА ВЫСОКИХ ТЕХНОЛОГИЙ

Открытые семантические технологии проектирования интеллектуальных систем

OSTIS-2012

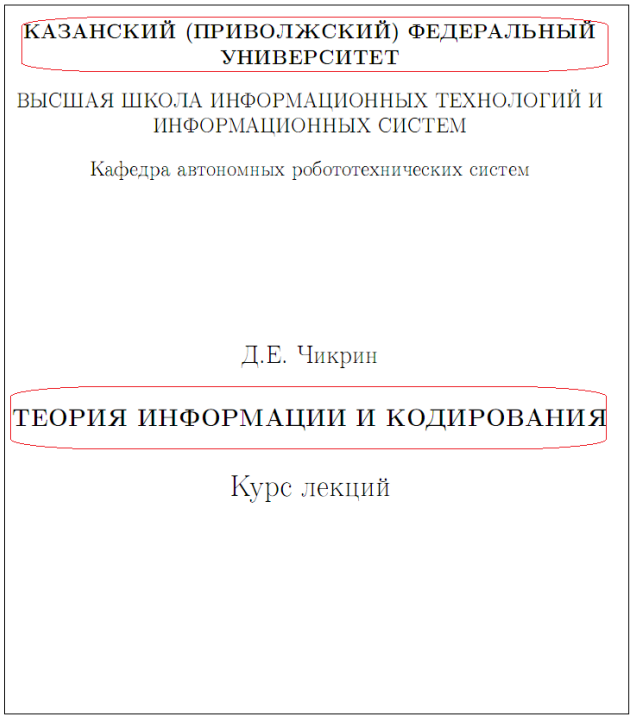
Open Semantic Technologies for Intelligent Systems

МАТЕРИАЛЫ
II МЕЖДУНАРОДНОЙ
НАУЧНО-ТЕХНИЧЕСКОЙ КОНФЕРЕНЦИИ

Следующая трудность в выделении заголовков – наличие шапок, в основном, с названиями учреждений, к которым относится данный труд. Проблема с ними – в том, что они могут иметь такой же по величине шрифт, что и заголовок,

однако почти во всех случаях текстовый блок шапки будет больше блока заголовка по площади. Значит, алгоритм отдаст приоритет шапке (см. иллюстрацию 2). В примере из приложения у заголовка и шапки одинаковый шрифт, как по гарнитуре, так и по стилю и размеру. При этом алгоритм на простой эвристике отдаст предпочтение шапке, как большей по площади.

Иллюстрация 2. Демонстрация коллизий шапок и основного заголовка документа



Следующая проблема возникает в случае, когда заголовок состоит из нескольких частей с разными шрифтами, как в примерах титульных страниц документов, изображённых на иллюстрациях 3 и 4. Пренебрегать второй частью с более мелким шрифтом нельзя, так как выделенный заголовок будет неполным.

Иллюстрация 3. Пример заголовка из двух частей с разным форматированием №1

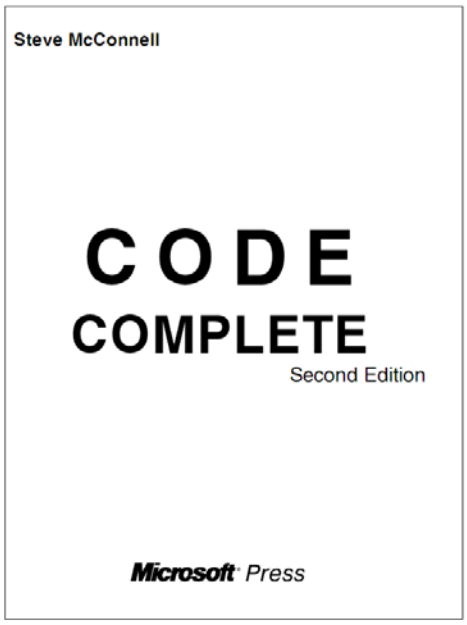
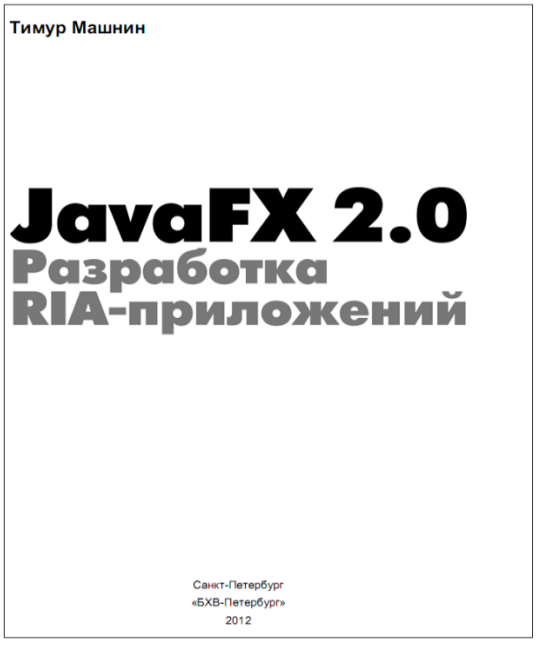


Иллюстрация 4. Пример заголовка из двух частей с разным форматированием №2

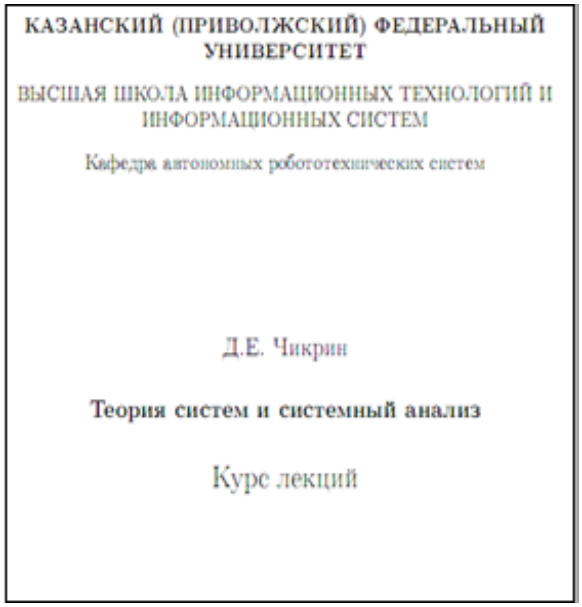


Наконец, ещё одна, самая коварная трудность при выделении заголовка – это то, что заголовки могут не иметь самый большой шрифт.

Рассмотрим пример на иллюстрации 5. Очевидно, заголовок здесь «Теория систем и системный анализ», однако самые большие шрифты здесь – у ша-

пок с названием учреждения и строки «Курс лекций». При этом алгоритм с учётом всех предыдущих проблем отсеет шапки, но разметит «Курс лекций» как заголовок.

Иллюстрация 5. Пример документа с фальш-заголовком


КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ

ВЫСШАЯ ШКОЛА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра автономных робототехнических систем

Д.Е. Чикрин

Теория систем и системный анализ

Курс лекций

Однако можно заметить некоторые особенности, в частности, то, что настоящий заголовок выделен жирным шрифтом. Возникает вопрос об учёте дополнительной подкатегории заголовков, учитывающих эти особенности.

МЕТОДЫ

В предыдущем разделе были рассмотрены некоторые распространённые группы проблем, возникающих при выделении заголовков даже из документов весьма узкой категории. Ниже предлагаются решения некоторых из этих проблем.

Выделение текстов из иллюстраций и исследование внутренних кодировок PDF-документов находятся за рамками этой статьи, однако другие проблемы с форматом можно частично обойти или решить. Ниже предложены возможные варианты:

- Для учёта рисунков с текстовыми подписями рекомендуется предусмотреть наборы условий и параметров, согласно которым текстовый блок

можно классифицировать как подпись (или иного рода запись, не несущую смысла) и исключить из рассмотрения.

- Для исправления искажений при считывании текста, связанных с неточной шириной и расстановкой пробелов, можно предусмотреть словарь «несуществующих» сочетаний; если последовательность символов встречается в этом словаре, значит, это часть слова, которую надо объединить со следующей либо предыдущей частью. В случае склеивания слов нужно смотреть на расстояния между символами в исходном документе. Часто фактические отступы между словами в документе больше отступов между символами, и, выбрав или иным образом определив граничное значение отступа, можно классифицировать последние как символьные или словарные.

Для решения второй проблемы – наличия нескольких титульных страниц и неизвестного местонахождения верного заголовка – необходимо детектировать по возможности все титульные страницы и пытаться выделить заголовок из каждой. В своём программном решении мы использовали отношение суммарной площади символов к площади страницы, отбирая титульные страницы по экспериментально подобранному пороговому значению. При этом при наличии минимум двух титульных страниц, значит, и двух заголовков встаёт проблема выбора одного из них. В идеале алгоритм должен выделять со всех страниц одинаковый заголовок, но гарантировать такой результат нельзя. Так как чаще всего встречаются документы первых трёх категорий (то есть с одно, двумя или тремя титульными страницами), а первая страница обычно оформлена как обложка, из которой трудно выделить текст, то авторы в своём программном решении за некоторыми исключениями отдавали предпочтение заголовку с последней титульной страницы.

Чтобы отсеять шапки из документов, можно определить словарь типичных слов, встречающихся в них: типы учебных учреждений, научных министерств и т. д., и проверять их наличие в текстовых блоках на этапе выделения заголовка либо применить машинное обучение.

Для решения проблемы с заголовком, состоящим из нескольких частей с разным форматированием, потребовалось ввести новое понятие: ***добавочный заголовок (small title)*** – часть заголовка, меньшая его основной части по размеру шрифта (также может отличаться по цвету) и находящаяся рядом с ним. Для

того чтобы понять, как выделять добавочный заголовок, нужно знать отличительные признаки этой сущности. Нами выделены следующие признаки добавочного заголовка:

- шрифт является наибольшим на странице за исключением шрифта основного заголовка;
- шрифт добавочного заголовка совпадает со шрифтом заголовка по стилю, гарнитуре и регистру (uppercase, lowercase or mixed-case);
- находится рядом с заголовком (т. е. расстояние до блока заголовка – наименьшее среди всех расстояний до других блоков);
- комплекс из заголовка и добавочного заголовка значительно отдалён от других текстовых блоков страницы;
- может находиться как сверху, так и снизу основного заголовка;
- расстояние от заголовка до добавочного заголовка не должно превышать 1,5 * max (vertSpacing); если у обоих 0 (оба однострочные), то может быть любым;
- шрифт добавочного заголовка предполагается не более чем в 1,5 раза меньшим шрифта основного заголовка.

Учитывая вышеперечисленные признаки, схема учёта добавочного заголовка при выделении заголовка следующая: так как добавочного заголовок обязан находиться рядом с основным заголовком, то достаточно знать три расстояния – расстояние между заголовком и предполагаемым добавочным заголовком, расстояние от заголовка до ближайшего блока с другой стороны и расстояние от добавочного заголовка до ближайшего блока с противоположной стороны. Если первое расстояние значительно меньше двух последних (хотя бы в 1,5 раза), совпадают гарнитура и стиль шрифта с аналогичными у заголовка, и добавочный заголовок имеет второй наибольший шрифт среди блоков, то этого кандидата можно считать добавочным заголовком и включить его в основную часть.

Наконец, для решения последней из проблем, описанных в предыдущей секции, в данном исследовании введено новое понятие: ***фальшивый заголовок*** – текстовый блок, не подходящий под заголовок по признаку размера шрифта, но имеющий отличительные особенности, которые обычно присущи заголовкам.

Нельзя отрицать, что фальшивый заголовок может состоять более чем из одного блока.

Как и для решения предыдущей подзадачи, необходимо определить свойства фальшивого заголовка:

- у него шрифт не более, чем в 1.5 раза, меньше, чем у основных кандидатов;
- фальш-заголовок оформлен жирным стилем, а основные кандидаты нет и/или фальш-заголовок оформлен uppercase-регистром, а основные кандидаты нет.

Значит, для обнаружения фальшивого заголовка требуется проверять все строки, не попадающие под понятие чистого (primary) заголовка по размеру шрифта, но обладающего вышеперечисленными свойствами. При этом ни один из кандидатов на заголовок не должен быть выделен жирным шрифтом или заглавными буквами, иначе нарушатся эти условия-свойства. Необходимо отметить, что ни один текстовый блок не может являться добавочным заголовком и фальшивым заголовком одновременно. Это легко доказать, сопоставляя свойства-признаки каждого из типов нестандартных заголовков, приведённых выше.

С учётом описанных ранее проблем общая стратегия выделения заголовка из целевых документов научной тематики выглядит следующим образом:

- выделить титульные страницы; вначале проверить первые три страницы: если ни одна из них не титульная, то просматривать весь документ постранично до тех пор, пока не будет найдена первая титульная страница;
- выбрать самый большой по длине из кандидатов на заголовок, отобранных с учётом шапок, добавочных заголовков и фальшивых заголовков.

Если выделено больше одного заголовка, то выбрать один из них.

РЕЗУЛЬТАТЫ

Для тестирования описанного выше алгоритма были взяты 60 PDF-документов научной тематики, удовлетворяющих условиям, приведённым в главе «Описание целевых документов». Для сравнения были также рассмотрены следующие алгоритмы и инструменты: простая эвристики на основании блока с самым крупным шрифтом на первой странице, Docear’s PDF Inspector из [2] и метод на основе машинного обучения из [6]. Первичный запуск всех перечисленных инструментов и методов дал следующие результаты:

Таблица 1. Первичные результаты тестов

Наименование инструмента/ алгоритма	**Общее число документов**	**Число успешно обработанных документов (правильных заголовков)**	**Точность (%)**
Простая эвристика	32	12	37,5
Docear’s PDF Inspector	32	12	37,5
Подход, основанный на машинном обучении	32	20	62,5
Метод, изложенный в данной статье	32	28	87,5

Как видно, у техник, основанных на простых эвристиках (первые два подхода), резко снижается точность, что вызвано наличием исключительных случаев, разобранных в главе «Методы». Однако точность может быть искажена ещё одной группой факторов: искажениями, связанными с пробелами, подписями к рисункам и другими проблемами обработки формата. При этом заголовок может быть распознан правильно, но из-за подобных искажений может не совпасть с ожидаемым результатом. Чтобы исключить влияние неверной расстановки пробелов в заголовках, будем убирать все пробельные символы из них (например, вместо “Genetic algorithm: theory and practice” получится “geneticalgorithm:theoryandpractice”). Таким образом, результаты после удаления пробелов следующие:

Таблица 2. Результаты тестов с применением нормализации

Наименование инструмента/ алгоритма	**Общее число документов**	**Число успешно обработанных документов (правильных заголовков)**	**Точность (%)**
Простая эвристика	32	18	56.25
Docear’s PDF Inspector	32	17	53.13
Подход, основанный на машинном обучении	32	24	75.0
Метод, изложенный в данной статье	32	31	96.88

ОБСУЖДЕНИЕ

Как видно из таблицы 2, влияние неверно распознанных пробелов существенно сказывается на результатах обработки документов. Однако метод, изложенный в статье, показывает существенно лучший результат. Это связано с тем, что простая эвристика и Docear's PDF Inspector не учитывают исключительных случаев, встречающихся в документах. Кроме того, простая эвристика не учитывает того факта, что заголовок не обязательно может быть выделен на первой странице. Подход же, основанный на машинном обучении, показывает довольно высокие результаты на широком круге документов. Однако он не всегда верно ведёт себя в случаях с фальшивыми или составными заголовками.

Следует также отметить, что выделение заголовков человеком отчасти является творческой задачей, и результаты, полученные разными людьми, могут не сойтись, значит, могут быть оспорены и результаты, показанные выше. Это связано с разнообразием информации, представленной на титульных страницах. Помимо заголовков и метаданных на них размещены различные пояснения, дополнения и подзаголовки, которые одними могут включаться в состав заголовка всего документа, другими – нет. Вопрос о том, что же включать в общем случае в состав заголовка, даже в рамках отдельно взятой категории документов, требует серьёзного исследования, выходящего за рамки данной статьи.

ЗАКЛЮЧЕНИЕ

Рассмотрены два основных принципа извлечения заголовков из PDF-документов: подход, основанный на стратегиях для выбора конкретного метода извлечения, и метод с использованием сложных эвристик для выделения заголовков из документов научной тематики. Оба метода были разработаны для улучшения качества выделения заголовка в общем случае, так как предыдущие исследования были ориентированы на простые методы либо машинное обучение. Однако алгоритмы машинного обучения имеют естественные пределы точности и полноты, которые не могут быть преодолены ни посредством выбора большего и лучшего тренировочного корпуса, ни какой-либо комбинацией готовых моделей допустимой точности. Поэтому можно заключить, что предлагаемые подходы имеют широкие перспективы в области автоматической структуризации документов и/или выделения отдельных структурных частей. А именно,

подход, основанный на стратегиях, позволит обрабатывать документы различных типов и жанров, а извлечение заголовков из научных документов описанным выше алгоритмом предоставляет лучшую точность относительно методов предыдущих исследователей.

Следует заметить, что подход, основанный на стратегиях, может быть также успешно использован и при формировании библиотечных записей, в частности, в журнале «Электронные библиотеки», позволяя индексировать не только книжные издания, но и статьи, отчёты и другую документацию. Также анализ стилевой информации и стратегии позволит углубиться в анализ титульных страниц в целом, например, для определения авторов, учреждения, которому принадлежит документ, издательства, типа труда и прочей информации.

Результаты, показанные выше, полностью оправдали ожидания авторов. Предложенный метод будет использован в дальнейших исследованиях (например, при извлечении заголовков секций или обнаружении оглавления документа) и уже может быть применён в различных задачах автоматизированной обработки документов (к примеру, в задачах суммаризации, классификации по темам и других), особенно научных книг и трудов.

СПИСОК ЛИТЕРАТУРЫ

1. *Lipinski M., Yao K., Breitinger C., Beel J., Gipp B.* Evaluation of Header Metadata Extraction Approaches and Tools for Scientific PDF Documents // 13th ACM/IEEE-CS Joint Conf. on Digital Libraries, Indianapolis, USA, 2013. ACM: 2013. P. 385–386.

2. *Beel J., Langer S., Genzmehr M., Müller M.* Docear's PDF Inspector: Title Extraction from PDF Files // 13th ACM/IEEE-CS Joint Conf. on Digital Libraries, Indianapolis, USA, 2013. ACM: 2013. P. 443–444.

3. *Marinai S.* Metadata Extraction from PDF Papers for Digital Library Ingest // 10th Int. Conf. on Document Analysis and Recognition (ICDAR). 2009. P. 251–255.

4. *Васильев А., Самусев С., Шамина О., Козлов Д.* Создание электронной библиотеки русскоязычных научных статей // Сб. работ участников конкурса науч. проектов по информ. поиску под ред. П. И. Браславского, Екатеринбург, Россия, 2007. Изд-во Урал. ун-та, 2007. С. 37–45.

5. *Beel J., Gipp B., Shaker A., Friedrich N.* SciPlore Xtract: Extracting Titles from Scientific PDF Documents by Analyzing Style Information (Font Size) // Research and Advanced Technology for Digital Libraries. 2010. P. 413–416.

6. *Hu Y., Li H., Cao Y., Teng L., Meyerzon D., Zheng Q.* Automatic extraction of titles from general documents using machine learning // 5th ACM/IEEE-CS Joint Conf. on Digital Libraries, New York, USA, 2005. ACM: 2005. P. 145–154.

7. *Elizarov A. M., Kirillovich A. V., Lipachev E. K., Nevzorova O. A., Solovyev V. D., Zhiltsov N. G.* Mathematical knowledge representation: semantic models and formalisms // Lobachevskii Journal of Mathematics. 2014. No 4. P. 348–354.

8. *Elizarov A. M., Lipachev E. K., Nevzorova O. A., Solovyev V. D.* Methods and means for semantic structuring of electronic mathematical documents // Doklady Mathematics. 2014. № 1. P. 521–524.

9. *Solovyev V. D., Zhiltsov N. G.* Logical Structure Analysis of Scientific Publications in Mathematics // Int. Conf. on Web Intelligence, Mining and Semantics, Sogndal, Norway, 2011. ACM: 2011, P. 21:1–21:9.

10. *Han H., Giles C.L., Manavoglu E., Zha H., Zhang Z., Fox E.A.* Automatic document metadata extraction using support vector machines // 3rd ACM/IEEE-CS Joint Conf. on Digital Libraries, Houston, USA, 2003. ACM: 2003. P. 37–48.

11. *Peng F., McCallum A.* Information Extraction from Research Papers Using Conditional Random Fields // Inf. Process. Manage. 2006. No 4. P. 963–979.

12. *Nakagawa K., Nomura A., Suzuki M.* Extraction of logical structure from articles in mathematics // Int. Conf. on Mathematical Knowledge Management, 2004. Springer: 2004. P. 276–289.

13. *Beel J., Gipp B., Langer S., Genzmehr M., Wilde E., Nürnberger A., Pitman J.* Introducing Mr. DLib, a Machine-readable Digital Library // 11th Annual Int. ACM/IEEE Joint Conf. on Digital Libraries, Ottawa, Ontario, Canada, 2011. ACM: 2011, P. 463–464.

14. *Granitzer M., Hristakeva M., Knight R. and Jack K.* A Comparison of Metadata Extraction Techniques for Crowdsourced Bibliographic Metadata Management // 27th Annual ACM Symposium on Applied Computing, Trento, Italy, 2012. ACM: 2012, P. 962–964.

15. *Yilmazel O., Finneran C. M., Liddy E. D.* MetaExtract: an NLP system to automatically assign metadata // 4th ACM/IEEE-CS Joint Conf. on Digital Libraries, Tuscon, USA, 2004. ACM: 2004. P. 241–242.

16. *Mayank S., Barnopriyo B., Priyank P., Manvi G., Sidhartha S.* OCR++: A Robust Framework For Information Extraction from Scholarly Articles // arXiv preprint arXiv:1609.06423. 2016. P. 1–9.

TITLE EXTRACTION FROM ENGLISH SCIENTIFIC BOOKS IN PDF FORMAT

D. S. Filippov

Higher School of Information Technologies and Intelligent Systems at Kazan (Volga region) Federal University

dmitriyfil1995@gmail.com

Abstract

Relevance of the issue under study is due to tenuity of methods proposed by other researchers that use simple heuristics or machine learning algorithms. The purpose of the article is to provide better way to extract titles from scientific PDF documents and offer better and more reasonable approach to title selection generally. The leading approach to the study is regard as many cases and problems appeared during extraction as possible and find an approach to solve all of them. The results showed the efficiency of chosen approach in case of having a document set with all of considered problems. The research highlights that deep analysis of current task problem is a perspective to make the best solutions and tools. The article may be useful for all researchers and developers who often encounter the problem of document structural analysis or title detection as secondary task of a main program workflow.

Keywords*: Pdf processing, title extraction, header extraction, strategy based approach, title heuristic, structural analysis, style information, text analysis, document analysis, information extraction*

REFERENCES

1. *Lipinski M., Yao K., Breitinger C., Beel J., Gipp B.* Evaluation of Header Metadata Extraction Approaches and Tools for Scientific PDF Documents // 13th ACM/IEEE-CS Joint Conf. on Digital Libraries, Indianapolis, USA, 2013. ACM: 2013. P. 385–386.

2. *Beel J., Langer S., Genzmehr M., Müller M.* Docear's PDF Inspector: Title Extraction from PDF Files // 13th ACM/IEEE-CS Joint Conf. on Digital Libraries, Indianapolis, USA, 2013. ACM: 2013. P. 443–444.

3. *Marinai S.* Metadata Extraction from PDF Papers for Digital Library Ingest // 10th Int. Conf. on Document Analysis and Recognition (ICDAR). 2009. P. 251–255.

4. *Vasilyev A., Samusev S., Shamina O., Kozlov D.* Digital library of Russian-language scientific articles creation // coll. of academic papers of information search competetion participants, edited by P. I. Braslavsky, Ekaterinburg, Russia, 2007. Publishing Office of Ural University, 2007. P. 37–45.

5. *Beel J., Gipp B., Shaker A., Friedrich N.* SciPlore Xtract: Extracting Titles from Scientific PDF Documents by Analyzing Style Information (Font Size) // Research and Advanced Technology for Digital Libraries. 2010. P. 413–416.

6. *Hu Y., Li H., Cao Y., Teng L., Meyerzon D., Zheng Q*. Automatic extraction of titles from general documents using machine learning // 5th ACM/IEEE-CS Joint Conf. on Digital Libraries, New York, USA, 2005. ACM: 2005. P. 145–154.

7. *Elizarov A. M., Kirillovich A. V., Lipachev E. K., Nevzorova O. A., Solovyev V. D., Zhiltsov N. G.* Mathematical knowledge representation: semantic models and formalisms // Lobachevskii Journal of Mathematics. 2014. No 4. P. 348–354.

8. *Elizarov A. M., Lipachev E. K., Nevzorova O. A., Solovyev V. D.* Methods and means for semantic structuring of electronic mathematical documents // Doklady Mathematics. 2014. No 1. P. 521–524.

9. *Solovyev V. D., Zhiltsov N. G.* Logical Structure Analysis of Scientific Publications in Mathematics // Int. Conf. on Web Intelligence, Mining and Semantics, Sogndal, Norway, 2011. ACM: 2011. P. 21:1–21:9.

10. *Han H., Giles C.L., Manavoglu E., Zha H., Zhang Z., Fox E. A.* Automatic document metadata extraction using support vector machines // 3rd ACM/IEEE-CS Joint Conf. on Digital Libraries, Houston, USA, 2003. ACM: 2003. P. 37–48.

11. *Peng F., McCallum A.* Information Extraction from Research Papers Using Conditional Random Fields // Inf. Process. Manag. 2006. No 4. P. 963–979.

12. *Nakagawa K., Nomura A., Suzuki M.* Extraction of logical structure from articles in mathematics // Int. Conf. on Mathematical Knowledge Management, 2004. Springer: 2004. P. 276–289.

13. *Beel J., Gipp B., Langer S., Genzmehr M., Wilde E., Nürnberger A., Pitman J.* Introducing Mr. DLib, a Machine-readable Digital Library // 11th Annual International ACM/IEEE Joint Conf. on Digital Libraries, Ottawa, Ontario, Canada, 2011. ACM: 2011. P. 463–464.

14. *Granitzer M., Hristakeva M., Knight R., Jack K.* A Comparison of Metadata Extraction Techniques for Crowdsourced Bibliographic Metadata Management // 27th Annual ACM Symposium on Applied Computing, Trento, Italy, 2012. ACM: 2012. P. 962–964.

15. *Yilmazel O., Finneran C. M., Liddy E. D.* MetaExtract: an NLP system to automatically assign metadata // 4th ACM/IEEE-CS Joint Conf. on Digital Libraries, Tuscon, USA, 2004. ACM: 2004. P. 241–242.

16. *Mayank S., Barnopriyo B., Priyank P., Manvi G., Sidhartha S.* OCR++: A Robust Framework For Information Extraction from Scholarly Articles // arXiv preprint arXiv:1609.06423. 2016. P. 1–9.

СВЕДЕНИЯ ОБ АВТОРЕ

ФИЛИППОВ Дмитрий Сергеевич – бакалавр Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета, студент 1 курса магистратуры.

Dmitriy Sergeevich FILIPPOV – has bachelor's degree of the Higher School of Information Technologies and Intelligent Systems at Kazan (Volga region) Federal University, 1st year graduate student.

e-mail: dmitriyfil1995@gmail.com

Материал поступил в редакцию 4 июня 2018 года